SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Dipartimento di

Informatica, Sistemistica e Comunicazione

Dottorato di Ricerca in: Informatica Ciclo: XXIX

Graph-based Clustering Algorithms for Word Sense Induction

Cognome: Cecchini Nome: Flavio Massimiliano

Matricola: 787772

Tutore: Prof. De Paoli Flavio Maria

Cotutore: Prof. Dr. Biemann Christian

Supervisor: Dott.ssa Fersini Elisabetta

Coordinatore: Prof.ssa Bandini Stefania

ANNO ACCADEMICO 2015/2016

Бугуйл хаях ойроос
нар мандах нь сонин
Бууж үдлэх холоос
айл нь харагдах нь хачин

Б. Явуухулан

Che strano vedere il sole che sorge
e poterlo prendere al cappio
Che strano scorgere quella tenda
e dover cavalcare un pomeriggio per raggiungerla

B. Yavuukhulan

Abstract

This dissertation is about Word Sense Induction (WSI), a branch of Natural Language Processing concerned with the automated, unsupervised detection and listing of the possible senses that a word can assume relative to the different contexts in which it appears. To this end, no external resources like dictionaries or ontologies are used. Among the many existing approaches to WSI, we focus specifically on modelling the context of a word through a graph and on running a clustering algorithm on it: the resulting clusters are interpreted as implicitly describing the possible senses of the word. Fundamental notions of WSI, basic concepts and some WSI approaches selected from literature are presented and examined in the first part of this work.

In the second part, we introduce our threefold contribution. Firstly, we define and explore a weighted (together with an unweighted) Jaccard distance, i.e. a distance on the nodes of a positively weighted undirected graph which we use to obtain second-order relations from the first-order ones modelled by the graph (e.g. co-occurrences). Moreover, we define the related notion of *gangplank edge*, a *separator* edge with weight greater than the mean weights of the edges incident to either of its two ends, and finally a new synthetic interpretation of the curvature on a graph, seen as the difference between weighted and unweighted Jaccard distances between node couples. Our Jaccard distance is at the basis of the second contribution: three novel graph-based clustering algorithms expressly created for the task of WSI, respectively the gangplank clustering algorithm, an aggregative clustering algorithm and a curvature-based clustering algorithm. The third contribution is a novel evaluation framework for graph-based clustering algorithms for WSI, consisting of two word graph data sets (one for co-occurrences and one for semantic similarities) and a new *ad hoc* evaluation measure built around pseudowords. A pseudoword is the artificial conflation of two existing words, used as an ambiguous word whose (pseudo)senses are perfectly known. This enables to evaluate WSI

algorithms on an easily creatable and expandable data set. We carry out a pseudoword-based evaluation for a number of graph-based clustering algorithms, including our three proposed systems.

The investigation of how the parameters of a pseudoword affect an algorithm's outcomes, the comparison of the scores obtained by different evaluation metrics together with the detection of their biases, the size of the clusterings and the trends put in evidence by the hyperclustering step, the influence of the type of a word graph (based on semantic similarities or co-occurrences) on the output of an algorithm - all these factors, preceded by the comprehensive description of the task and the definition of novel concepts and instruments to tackle it, concur to give a deeper insight into the functioning and pitfalls of graph-based Word Sense Induction. We highlight and isolate the elements that determine how the results of an algorithm look like, discuss their properties and behaviours in relation to the word graph features and establish the pro and contra of each algorithm. Our analysis provides an experimental compass that helps pinpoint the right characteristics required by a clustering algorithm for the task of Word Sense Induction, and that helps orient the construction of a word graph. In particular, we have put in evidence the different syntagmatic versus paradigmatic contrast inherent to word graphs based on co-occurrences and semantic similarities.

Contents

List of Figures

List of Tables

Introduction

This dissertation is part of the wide field of Natural Language Processing (NLP), and deals in particular with the task of (textual) Word Sense Induction and Discrimination (WSI and WSD). These can be seen as the unsupervised counterparts of Word Sense Disambiguation, which is backed by a long and rich history of research, starting at least from [Weaver, 1949]. Only relatively recent developments (from around the 2000s onwards) have brought increasing attention to unsupervised approaches, whose challenges arise from the attempt to use as little information about the world as possible, instead exploiting patterns that naturally appear in the natural language. The passage from Word Sense Disambiguation to Word Sense Induction or Discrimination thus shifts the focus from a classification problem to a clustering problem.

As common to all approaches to text-oriented NLP, the task of WSI requires that a text be first modelled. Two of the dominating ways to do it are vector space models and graph models; in our work, we adopted the latter approach, so that we speak of graph-based WSI. Word graphs, usually in the form of undirected, weighted graphs, are intuitive and powerful mathematical structures that allow a meaningful manipulation and representation of the relationships between words, linking topological properties and weighting schemes directly to syntactical and lexical phenomena. More in detail, the goal of Word Sense Induction is:

> Given a word in a corpus, to infer all the possible senses that such word can assume in the corpus judging from its context alone.

In our graph-based setting, this goal is achieved by clustering the node set (the "vocabulary") of a graph that models the local context of a given word and implicitly identifying the obtained clusters with the different meanings assumed by that word in the corpus. Numerous graph-based clustering algorithms have been proposed in literature. We focus especially on the analysis of three of them, namely the Markov cluster algorithm [van Dongen, 2000], Chinese Whispers [Biemann, 2006] and MaxMax [Hope

and Keller, 2013]. While the first two rely on the concept of information flow simulation, the second one takes advantage of the weighted structure of a word graph to isolate connected subgraphs.

One of the major contributions of this dissertation, outspringing from the observation of already existing graph-based clustering algorithms, is the introduction of three novel approaches, revolving around different and new logics: the *gangplank* clustering algorithm, an *aggregative* clustering algorithm and a *curvature-based* clustering algorithm, all based on the definition of a *weighted Jaccard distance* between nodes of a graph (or equivalently, words in a vocabulary), which is also one of our major contributions. In the next paragraphs, we will briefly describe these new concepts.

The weighted Jaccard distance is directly inspired by the Jaccard index

$$\frac{|A \cap B|}{|A \cup B|}$$

for any two finite sets A and B, which defines a bounded distance between sets

$$1 - \frac{|A \cap B|}{|A \cup B|}.$$

We naturally extend this distance to the case of multisets (sets where each element has a multiplicity) and adapt it to the node neighbourhoods of a word graph. The resulting *weighted Jaccard distance* (and its parallel unweighted counterpart) has many interesting and desirable properties and allows considering the node set as a metric space. We deem it to be a simple but promising instrument for Word Sense Induction.

The *aggregative clustering algorithm* is the most direct application of the weighted Jaccard distance for WSI, reinterpreting the k-medoids algorithm but providing it with a medoid-initialization step. It is the only one of our three algorithms making use of a parameter.

The *curvature-based clustering* algorithm originates from the interaction of the unweighted Jaccard distance with its weighted version. Loosely related to other synthetic definitions of curvature on graphs (e.g. through the clustering coefficient), it is used to isolate subregions of a word graph whose elements are brought nearer to each other by the weighted structure of the graph (i.e. connected by positive curvature), than what appears from its bare topological structure.

Finally, the *gangplank clustering algorithm* is the more articulated approach of the three. It involves the weighted Jaccard distance to derive a second-order distance graph from an original word graph, and on this it applies the definition of *gangplank*: an edge that denotes a local weak connection between two nodes (i.e. between words). Gangplank edges are

then used as delimiters of semantically autonomous regions (i.e. clusters) of the word graph, that are identified through cyclical expansion and reduction steps around cluster seeds. This algorithm has been developed with the aim to require as less text pre-processing as possible: the combination of passage to the second order and local weighted topology helps deal with the small-world nature of word graphs and the noisy nature of text word co-occurrences.

The second major contribution of this work lies in the pseudoword evaluation framework that we present here. Pseudowords, first independently sketched in [Gale et al., 1992b, Schütze, 1992], are artificial conflations of two real words that are used to simulate homonymy and polysemy. The advantage herein is that their artificial nature permits us to tune the magnitude of their ambiguity as desired for our evaluation goals. We compute a fairly large double pseudoword data set, each one consisting of 1225 respectively semantic-similarity and co-occurrence-based ego graphs, arising from each possible combination of 50 monosemous terms. It is the largest such data set that we are aware of. On it, we perform both an analysis of the structure and properties of our pseudowords and an accurate and comprehensive evaluation of the three aforementioned existing and of our three novel clustering algorithms. We examine their trends according to different parameters of the pseudowords, making many noteworthy considerations and drawing conclusions about the nature of the functioning of the clustering algorithms, the influence of the word graph's type and the biases of some evaluation measures. To circumvent the latter, we define a new evaluation score expressly for the task of homonymy detection.

Reassuming the previous introduction, the goal of this dissertation is thus at least threefold: we introduce and expound novel mathematical instruments that can prove themselves useful for graph-based WSI; we introduce three novel graph-based clustering algorithms for WSI which exploit these new techniques and having a different nature than existing ones; and finally, we introduce two parallel novel pseudoword data set and a related evaluation framework, on which we perform a thorough comparison of some existing and our new clustering algorithms, gaining deeper insights into their functioning, the structure of word graphs and the considered evaluation measures, thereby defining a new evaluation measure.

This work is subdivided into two main parts, structured as follows.

The first part is concerned with the foundations of WSI and the concepts and instruments at the basis of the second part. Specifically, Chapter 1 introduces the topics and issues of Word Sense Disambiguation, Induction and Discrimination and sets the background of our work. Chapter 2

provides the reader with an overview and a brief insight into some basic notions concerning word graphs, graph theoretical concepts and significance measures that will be used in the second part of the work. Chapter 3 deals with the in-depth description and discussion of some of the existing clustering algorithms for WSI that form the inspiration of our work, presenting some practical issues of graph-based WSI.

The second part regards the novelties introduced in this dissertation. In Chapter 4, we describe the theoretical framework that underlies the novel mathematical instruments and clustering algorithms for the task of Word Sense Induction presented in this dissertation, as well as describing their implementation. Finally, Chapter 5 is dedicated to our novel pseudoword data set and the pseudoword evaluation framework, and examines the performances of various clustering algorithms evaluated on it. Chapter 6 closes this work with some final remarks. An appendix presents minor additions in the form of a small analysis of collapsed pseudowords and charts about the pseudoword evaluation.

Part I

Foundations of Word Sense Induction

Chapter 1

Overview and state of the art

1.1 Description and motivation

In Natural Language Processing, Word Sense Disambiguation (WSD) can be described as the task that tries to reproduce, at least partially, the way humans understand natural language: The goal is to assign to each word or group of words in a discourse its particular meaning in that context [Manning and Schütze, 1999, Martin and Jurafsky, 2000]. We remark that here and thereafter the term *word* will indicate its most shallow manifestation, that is, the mere complex of phonetical or graphical signs which can be isolated from a text and carry some *meaning*. The obvious obstacle to this task is that machines do not have any inherent knowledge of natural languages, and that they actually do not possess knowledge at all; they are just means to store data and perform operations on them. For this reason, for example, when the following three sentences are typed in a computer:

1. Let us *count* to ten!
2. We are going to visit the *count*'s manor tonight.
3. The first *count* accused him of having breached the contract.

the word *count* will be represented in each case as exactly the same sequence of bits in memory. By contrast, as humans we know that *count* has three radically different meanings here:

1. the verb with the meaning *to enumerate*,
2. the noun used as a title for a nobleman, and
3. the noun used as the legal term meaning a charge in a criminal action.

A reader will use all the knowledge he has previously collected prior to seeing these sentences to distinguish all three intended different meanings. On one side, he will use his native linguistic competence to differentiate the grammatical roles of each of the three occurrences of *count*. In the first sentence, *count* is a verb; this can be deduced from the syntactical pattern `/let us _/` of which it is part. In the second and third sentence, the word *count* has the role of a noun, which can be inferred both from the syntactical pattern `/the [adj.] _/` and, in the second sentence, from the morphological marking `/'s/` of the genitive case. Being verbs and nouns two different parts of speech, they also carry different functions and meanings, or at least express different facets of the same scene. We could say that on the linguistic level differences are drawn primarily through the recognition of some kinds of patterns or through the mechanical application of some rules. From a symmetrical point of view, this replicates the processes that we first have to pick up when we are learning a new language[1]. Even if such processes can be algorithmically simulated by tools like part-of-speech taggers or syntactic parsers [Martin and Jurafsky, 2000, Chapters 5,13], thus automatically discriminating the first occurrence of *count* from the other two, we still need another means to clarify the different meanings of the two otherwise grammatically identical nouns. Therefore, on the other side the reader will put to use the contextual information at his disposal. In the first sentence, since *ten* is a number, *count* will most probably refer to the act of enumerating. In the second sentence, *manor* refers to a big estate or a lord's residence, so that we can be quite sure that *count* here has the meaning of *nobleman*. Also, the terms *visit* and *tonight* immediately bring forth to our mind very peculiar scenes that we can link to books or movies; further, we could even think of a particular count. Similarly, in the third sentence, the verb *accuse* and the noun *contract* are revealing of a legal context like a courtroom. In the end, the process of disambiguation taps from all our previous experiences.
So, we can affirm that the disambiguation of words, that we constantly perform for any communication in which we take part, is mostly an interplay between a straightforward, linguistic recognition process and a more vague exploitation of previous knowledge about the world (this point will become more important when discussing about supervised and unsupervised WSD). Besides these two main poles, our comprehension of text or speech also depends on many other contextual and pragmatic circumstances. All such factors have to be emulated or supplied somehow to an automated system, and the way they are modelled is at the basis of the differences between the many existing approaches. In this regard, we

[1]Some of the processes mentioned in this section are studied in Psycholinguistics, for which [Clark and Clark, 1977] is an introduction.

already want to remark that this work deals with the lexical ambiguity among words belonging to the same word class, specifically that of *nouns*, more challenging than cross-class syntactic ambiguity.

The motivation for Word Sense Disambiguation is that natural languages are inevitably and intrinsically *ambiguous* and *vague* [De Saussure, 1916]. On the contrary, formal languages (like Java or Python) tend to be constructed so as to avoid any possible ambiguity, lest the interpreter perform not intended actions. At any one time, every natural language possesses only a finite vocabulary, made of words that themselves have to obey to phonotactical[2] and morphological rules, further reducing their possible forms. Still, the possible range of referents and concepts that can be expressed is unlimited. From this contrast originates the tension in language between the need of precise communication, up to the extreme of one word per concept/referent, and the preservation of economy of communication, since time and memory are limited, up to the extreme of one single word for every possible concept/referent (cf. [Lyons, 1968]). Since neither of these two drastic solutions is viable or satisfactory, by staying in the middle ground the phenomena of polysemy and homonymy will arise[3]. A speaker will eventually have to use words with different meanings but sharing the same identical form for some reason (homonymy), or, conversely, will resort to using the same word with different meanings in different contexts (polysemy). This is made possible by the fact that comprehension results not only from merely grammatical or syntactical features, but is also often mediated by context and other circumstances, as already discussed above.
More in detail, polysemy designates the fact that the occurrence of a word can carry different meanings which, even if differentiated or perceived as independent, come from a common etymological root and were derived by associations. An example of a polysemous word in English is *wood*: as a noun, it can either refer to a forest, *enter the wood*, to the material, *a door made out of wood*, or to an object made out of wood, like an instrument, *he plays a wood*. We just mention that the same word *wood* used as a verb also presents a certain degree of polysemy. Homonymy, on the contrary, denotes the occurrence of two words spelled (homography) or pronounced (homophony) the same way, but with different, unrelated meanings. Homonymy is the casual convergence of non etymologically related words. An example in English is *bow*, in its senses of *weapon for shooting arrows*, derived from the Proto-Germanic root **bugan*, *to bend*, and

[2]Phonotactics is the name of the restrictions of sound combinations that govern the creation of correct words in a given language. For example, in English it is not possible to have a word beginning with `rk-`, whereas e.g. in Georgian this combination is allowed.

[3]A study on how polysemy arises and is mentally processed can be found in [Klepousniotou, 2002].

front of a ship, derived from *bough*, itself coming from a different Proto-Germanic root **bogaz*, *shoulder*[4]. In general, we could claim that the senses of homonymous words are more independent from each other than those of a polysemous word.
The categories of polysemy and homonymy are best suited for common nouns. Proper nouns, though they share many grammatical characteristics with common nouns, can show a very different semantic and pragmatic behaviour[5], distinguished by a higher degree of arbitrariness in their usage. Similarly to homonymous nouns, many proper nouns have the same form as a result of casual convergence, sometimes even across languages, as they tend to undergo only small alterations (like e.g. the case of *Milan* as a city in Italy or a personal name in Serbia), and most of the time, being mere linguistic signs, there is no direct correlation between a referent and its noun. The peculiarity of a proper noun, though, is that it refers each time only to one specific instance between its possible referents. For example, *Milan* can refer to either one of the cities with that name in Italy, the United States, Iran, and so on, but never to the whole group of cities called *Milan*. The disambiguation process should then differentiate not only between different categories of referents like *person*, *location*, and so on (this is the task of Named Entity Recognition and Classification; see [Nadeau and Sekine, 2007] for a survey), but also between different instances in the same category. Finally, we notice that nearly every word in a language might be used as a proper noun without particular restrictions, and for this reason even native speakers may need extra details to point them to the intended referent. In the two sentences:

- Yesterday I went to *Milan*'s birthday.
- I like shopping in *Milan*.

a reader understands that the first *Milan* is a person, but may not know him. At the same time, the reader will understand that the second *Milan* is probably a city, but without further context it might not be clear if the city in Italy or in Tennessee, USA, is intended. Given how much more intricate the approach to proper nouns is with respect to common nouns, from a working point of view we could treat nouns as divided in two distinct lexical classes. Our main aim is to deal with *common nouns* (more on this topic in Section 2.1.8 and Section 4.3.1).
Since we have been considering the case of written texts, we would like to briefly remark that written language behaves quite differently from speech. Writing systems don't always specify different pronunciations

[4]See the etymological dictionary for English [Klein, 1971].

[5]A discussion on what sets nouns apart from other lexical categories can be found in [Baker, 2003].

of the same string of symbols, like e.g. *lead* (rhyming with *need*), meaning *to show the way* and *lead* (rhyming with *bed*), as the metal (a case of heterophonic homography), or conversely use different spellings for the same pronunciation, like e.g. *meat* and *meet* (so-called heterographic homophones).

We can formulate the task of Word Sense Disambiguation as follows:

Task 1. *Given a word with more than one possible meanings and its context in a corpus, to determine a definite sense of that word in that context, possibly using some kind of external knowledge.*

Whether (structured) external knowledge is exploited is the fundamental divide between the two main paradigms of the *supervised* and the *unsupervised* approach, which will be discussed in Sections 1.2 and 1.3. The former is called Word Sense Disambiguation proper, and the latter Word Sense Induction (WSI) or Word Sense Discrimination (again WSD), with slightly different nuances.

Word Sense Disambiguation is a very complex task, but it is fundamental for the development of systems that make use of language-based interaction, like machine translation, web search engines, question answering, text and speech generation, and many other applications; see e.g. [Hung et al., 2005, Carpuat and Wu, 2007, Bhogal et al., 2007, Navigli and Crisafulli, 2010, Plaza et al., 2011].

1.1.1 Word senses: granularity and distribution

The nature of the *sense* or *meaning* of words has been at the center of passionate philosophical and linguistic debates since ancient times, and is at the basis of the branch of *semantics* in modern linguistics [Carnap, 1948]. The theoretical definition of *word sense* is beyond the scope of this work; instead, in the previous section 1.1 we just referred implicitly to the common everyday notion of "sense", and focused on the phenomena of polysemy and homonymy, which, from a point of view of word usage, are of more immediate concern to the task of Word Sense Disambiguation. In the same spirit, in this section we will present some issues with respect to the representation of word senses in the field of WSD. Ours is a practical point of view, but we acknowledge the importance of a more theoretical investigation to better define and direct the efforts and the foundations in WSD research and more generally in NLP.

When using some kind of existing knowledge to learn to disambiguate word senses, e.g. from a corpus where words are tagged with their appropriate sense, as is often the case for supervised approaches (see Section 1.2), computational linguists always make use of an explicit or implicit underlying *sense inventory*. This choice determines how senses are organized and when they are considered distinct. Taking the word *peach* as an example, a WSD algorithm will behave in different ways whether we decide that the possible senses of *tree* and *plant* are distinct or not; in the latter case, in the sentence *In my garden, I grow both peaches and tomatoes* both *peaches* and *tomatoes* will belong to the same class, to whose computational definition both words' shared contexts will contribute. Further, the fact that being a *tree* implies being a *plant*, a *eukaryote*, a *living being*, and so on, and, conversely, that a *tree* might be of genus *Prunus* or *Juglans*, raises the problem of *sense granularity*. The preference for a fine or a coarse sense inventory, i.e. how much we are willing to differentiate between word senses, influences the outcome and the scalability of a WSD algorithm, and is not obviously solved [Palmer et al., 2007]. Additionally, the arbitrariness in the choice of granularity, and more generally the concept of a sense inventory as a partition over all possible senses, and of word sense in general, has been criticized in literature [Kilgarriff, 1997, Hanks, 2000, Kilgarriff, 2007], and graded representations, in contrast with the binary nature of a partition, have been proposed [Erk and McCarthy, 2009], and has also been one of the motivating factors in the development of unsupervised approaches (see Section 1.3). The issue of choosing an adequate sense inventory has its importance also in the field of Named Entity Recognition (NER) [Nadeau and Sekine, 2007], especially for its specific tasks of named Entity Linking and Coreference Resolution, which actually have many elements in common with WSD [Moro et al., 2014], though they may be inspired by different theoretical and logical assumptions.

We could regard a sense partition detailed as above as a choice made on a "vertical" sense axis, because of the hierarchy that in some settings can be associated to some relations, like"being a *tree* implies being a *plant*". Besides it, we also envision a "horizontal", more lexical kind of granularity that stems from polysemy and homonymy, as discussed in Section 1.1. For example, consulting the lexical resource WordNet[6] [Miller, 1995] for the term *shelter*, we will find at least three meanings, very close to each other: a structure that provides protection, a protective covering, or the condition of being protected. The claim that these three senses are truly perceived as distinct by a speaker, or that they can be distinguished based on the context of the word *shelter*, could be and has been argued, and, in the specific case of WordNet, there have been proposals to extract coarser polysemic relations from it [Jiamjitvanich and Yatskevich, 2009]. In any

[6] `https://wordnet.princeton.edu/`

case, the subtlety that we choose to apply to such discriminations is again very influential on the training and on the efficacy of a WSD algorithm, and directly depends on the goals and on the domain of the disambiguation task at hand. Actually, we could assert that the issue of granularity is ingrained in the same definition of the task of Word Sense Disambiguation and can not be separated from it: each formulation of the task gives different coordinates to the problem. Unsupervised approaches (Section 1.3) are more flexible than supervised ones in this regard. In a sense, supervised approaches are top-down whereas unsupervised approaches are bottom-down techniques with respect to granularity.

The choice for more or less granularity is also influenced by the skewed distribution of senses of a given word. The most common case for a polysemous word is to have one or two primary senses that are considered as the most typical ones and overshadow the others, which only appear in more specific and narrower contexts. For semantically related senses, as those of *shelter* cited above (if we agree to treat them as distinct), this can be explained by the fact that each sense is just a particular declination of a generic concept (in this case of *protection*). Instead, when talking about pure homonymy we can acknowledge the tendency of human language not to overload a single term with too many possible different meanings[7], so that a sense might become predominant over another one over time. In any case, the skewed distribution of word senses makes it more difficult for a Word Sense Disambiguation system to detect less-used meanings of a term, and even more so if its output has a coarse granularity, since the appearances of the secondary meanings might be more occasional and fragmented. Again, these argumentations lend themselves to a critique of the generic concept of *word sense*.

In the end, the natural skewness of word sense distribution and the choice of a granularity for the sense inventory are among the major difficulties encountered in WSD, and every approach to it directly or indirectly tries to find a compromise in their interplay.

1.2 Supervised approaches

In a supervised approach, the goal is to exploit some already existing information, knowledge or resource to disambiguate a word by explicitly stating the entity to which it refers. The parallel that first comes to mind is a dictionary: the entry of a word lists its possible senses, so that, to disambiguate it, we just have to learn how to link a particular sense to

[7] Even languages with an extremely high rate of homophones, like Chinese, regularly develop strategies to solve such ambiguities [Yip, 2000].

the word's occurrence that we are considering. In this regard, supervised WSD can be considered similar to a classification problem (see [Agirre and Edmonds, 2007, Chapter 7] for a more complete tractation on the subject). A tagged corpus acts as a dictionary of sorts: each term is labelled with its correct sense, and machine learning techniques can be used to put labels in correlation with a given set of context features. The set of labels forms a ground truth that comes from the knowledge of the annotators and the general consensus between them. A labelling implicitly defines a sense inventory or taxonomy, with its respective granularity (as discussed in Section 1.1.1). There are many manually compiled lexical resources that already give a hierarchically structured ontology, of which one of the most used is *WordNet*[8] [Miller, 1995], defining senses through sets of synonyms, hypernyms and hyponyms. More encyclopedic resources like Wikipedia[9] or DBpedia[10] are also widely used, along with BabelNet[11] [Navigli and Ponzetto, 2012], which is a semantic network that draws from all the aforementioned resources.
Historically, rule-based approaches like decision lists [Yarowsky, 2000] have constituted the earliest attempts at Word Sense Disambiguation. However, now most of the supervised WSD algorithms make use of probabilistic language models. Naive Bayes classifiers [Manning and Schütze, 1999, Chapter 7] are very simple and still popular among many other generative classifiers. Following the introduction of conditional random fields [Lafferty et al., 2001], some Word Sense Disambiguation systems have made use of this approach (e.g. [Hatori et al., 2008, Li, 2013]). Another successful method which has achieved very high performances is that of support vector machines (SVM) [Vapnik, 1995], where words are represented in a vectorial space and separated by a hyperplane. Originally developed for binary classification, it can be adapted to more than two senses if we consider each sense to be opposed to all others. Some systems using SVM are [Murata et al., 2001, Lee et al., 2004], along with a survey in [Joshi et al., 2005]. A vectorial word representation, induced by some predefined features, is also at the basis of methods like the k-nearest neighbours algorithm (on this and other related topics, see [Duda et al., 2001, Chapter 10]) (where k is a parameter that has to be set experimentally), which has proved to be quite solid and has been used in some works on supervised WSD (e.g. [Rezapour et al., 2011]). An interesting approach to Word Sense Disambiguation is that of neural networks [McCulloch and Pitts, 1943], where artificial neurons are trained to specialize in the recognition of hidden patterns, like in [Towell and Voorhees, 1998]. Recently, there has been a resurgence of interest in this field with the advent of the so-called

[8] `https://wordnet.princeton.edu/`
[9] `www.wikipedia.org`
[10] `http://wiki.dbpedia.org/`
[11] `www.babelnet.org`

deep learning and word embeddings [Bengio, 2009, Mikolov et al., 2013b]; some attempts have been made to use these techniques for Word Sense Disambiguation (as in [Wiriyathammabhum et al., 2012]), and they can actually be used for unsupervised Word Sense Induction, too, as mentioned in Section 1.3.1. In general, we note that many of the aforementioned classifying techniques have often been developed in the more generic field of Machine Learning and later applied to the specific case of WSD. It is also thinkable to combine many of these classifiers at once and see how much and how well they agree; this method has been shown to obtain good results and to improve the scores of the single classifiers. One of these "aggregative methods" is AdaBoost [Freund et al., 1999].
A critique to supervised approaches is that they are strongly dependent on their training sets and are prone to overfitting: An algorithm trained and tested on the same domain will likely not perform as well on an other, unknown domain. To retrain it, however, new annotated corpora would be necessary, and obtaining them would be expensive both in terms of time and costs. Unsupervised approaches try to circumvent these problems and will be discussed in Section 1.3.

1.2.1 Evaluation

Evaluation for supervised Word Sense Disambiguation is quite straightforward: the obtained results can be directly compared to the ground truth at disposal. Nonetheless, how such ground truth should be represented can be a cause of debate (as evidenced in Section 1.1.1), and consequently inter-annotator agreement can be low, making the validity of the evaluation questionable. There are some projects that focus on representing word meanings in a way as unequivocal as possible [Hovy et al., 2006]. However, many problematics still arise which are in common with unsupervised clustering evaluation, and which regard the interpretation of different scores and their possible bias. We remand to Section 1.3.3 for a brief overview.

1.3 Unsupervised approaches

Recalling the Word Sense Disambiguation definition of Task 1, as already mentioned in Section 1.1 we can present unsupervised Word Sense Disambiguation in two flavours: either as Word Sense Induction (WSI) or as Word Sense Discrimination (again WSD).
We define the task of Word Sense Induction as

Task 2. *Given a word in a corpus, to infer all the possible senses that such word can assume in the corpus judging from its context alone.*

Along the same lines, the task of Word Sense Discrimination will be

Task 3. *Given a word in a corpus, to assign a sense to each of its occurrences, based only on the context of the occurrences.*

The key difference between Tasks 2 and 3 on one side and Task 1 on the other side is that no (structured) external knowledge is used to determine the senses of a word or the sense of an occurrence. External knowledge is represented by annotated data, knowledge bases, lexical resources and in general dictionaries and ontologies that are usually compiled by or take advantage of human annotations, as mentioned in Section 1.2. Instead, the foundational linguistic and working assumption of unsupervised approaches is rooted in the basic principle of *distributional semantics* [Harris, 1954]:

> Difference of meaning correlates with difference of distribution.

The same concept is synthesised in a famous motto by J. R. Firth [Firth, 1957]:

> You shall know a word by the company it keeps.

These assumptions imply that each occurrence can have only a single sense at a time. The advantages of unsupervised approaches over supervised ones lie in the fact that they avoid the *knowledge bottleneck* [Gale et al., 1992a], the problem that for supervised approaches arises from the necessity to gather and, most relevantly, label enough data to train a Word Sense Disambiguation algorithm and keep it up to date, taking into account also the new nuances and entities that could show up over time.

Unsupervised Word Sense Disambiguation systems differ about what they consider as *context* and how they model it. Grammatical and syntactical patterns, text formatting, co-occurrences in a specified text unit and other measurable, objective features can all be combined to describe the surrounding context of a term. The main point of divergence is then how words and relations between words are represented: either in a *continuous* or in a *discrete* space. This is reflected by the crucial subdivision into *vector space models* and *graph-based approaches*. What all these methods have in common is that in the end some *clustering* algorithm is applied on the word space: either to partition the global context of a term in subsets that we interpret as its possible senses (WSI), or to "divide the occurrences of a word into a number of classes, thus determining for any two occurrences whether they belong to the same sense or not" (WSD, after [Schütze, 1998]). In both cases, senses are not explicitly defined by stating what their corresponding referents are, but are implicitly defined by a subset of the context.

In the following, we are going to focus on these two main approaches and mention some proposals taken from the vast literature about the subject. We have decided not to discuss other WSI systems based on probabilistic models or other techniques.

1.3.1 Vector space models

Vector space models of semantics try to represent words as vectors in a given space. A fixed set of features is assigned to each word depending on its context, and interpreted as the entries of a vector in a space of corresponding dimensionality. The advantage of using a vector space $\mathcal{V}$, most of the time of the type $\mathbb{R}^n$ for some natural number $n \in \mathbb{N}$, is its continuity and the fact that it generalizes the original discrete space of words, so that each point $v \in \mathcal{V}$ can be seen as a word, defined by the "sum" of its components. Since $\mathbb{R}^n$ comes with many possible metrics, this allows to compute distances in $\mathcal{V}$ between words, e.g. through cosine similarity, L_1 or Euclidean distance, and so on, and to adopt clustering algorithms like k-means, unsupervised k-nearest neighbours, and similar ones.

Often, the Euclidean vectorial representation is achieved through a frequency matrix [Turney and Pantel, 2010] which represents the document. The entries of the matrix are the number of occurrences (frequency) of a word, a pair of words, or other observed items inside a particular context, like a syntactical or grammatical pattern, a window around the word, the entire document, and so on. In the example of a word-paragraph frequency matrix, each row would represent the word through its frequency in each paragraph of the document as a vector $v \in \mathbb{N}^n$, where n is the total number of paragraphs. We notice that other, secondary values, more sophisticated than frequency, can be used in the matrix, such as TF-IDF score, pointwise mutual information (see Section 2.2.1) and variants thereof. In any case, a similar matrix will usually be very sparse or noisy, and smoothing techniques might find an application here. In this case, e.g. Latent Semantic Analysis (LSA) [Deerwester et al., 1990] is a method to reduce the size (the rank) of a frequency matrix through Singular Value Decomposition (SVD): the original matrix M is mapped into a lower-dimensional matrix which only retains the k biggest eigenvalues of M, which are those representing the biggest axes of variation among words, so that only these eigenvalues are later used for the computation of similarity between words.
Vector space models in general are not limited to unsupervised approaches; in fact, they can and often are used in conjunction with any machine learning algorithm [Witten and Frank, 2005, Collobert and Weston, 2008].

In [Schütze, 1998] a vector-based context-group discrimination is proposed for Word Sense Discrimination. Here the aim is to cluster a *word space* whose elements represent all the contexts of the occurrences of a word w. Context vectors are themselves obtained from the sum of the single word vectors belonging to each context. In the word space of w, vectors are grouped so as to minimize the distances between the elements in a group, using a centroid, and to maximize the distance between groups. The centroid of a group is the vector that minimizes the distance from all elements of the group; it is the group's "mean vector" and it is used as a condensed representation of the sense implicitly associated to that group. We notice that the definition and computation of a centroid is only possible in a continuous space. The centroid need not truly correspond to a word in the document.
Another classical approach is that of [Pantel and Lin, 2002], where *clustering by committee* is introduced. Words are represented by (syntactical) context features, for which a discounted version of pointwise mutual information (PMI) is computed and used for the entries of the word vectors. The cosine coefficient of two word vectors is taken as their similarity. Here all the senses in the document are discovered simultaneously in a first phase and represented by *committees*, that is, small clusters of similar words. Again, centroids are used to represent committees. Committees originate from clustering the top k most similar elements of each term. Two threshold values are used to form committees and decide if more have to be created. Subsequently, each element of the document is linked to its most similar committees, according to another threshold. This approach might be seen as the inverse of the previously discussed one of [Schütze, 1998], since senses are located first and then the words are assigned to them, instead of selecting words for the disambiguation and inferring senses from their contexts.
A more recent take to vector space models are *word embeddings*, which are linked to the concept of neural networks. Language models inspired by neural architectures like in [Bengio et al., 2003] have received increasing attention during the last years. The most discussed and debated vectorial representation of words through embeddings has been proposed in [Mikolov et al., 2013a]. There, real vectors of fixed length are randomly initialized for each word. Subsequently, a very shallow neural network is used to recompute each word vector singularly, based on a given context, e.g. the four preceding and the four following words. The new form of a vector is used immediately in all computations. In the end, the recursive nature of such calculations will return word vectors that encode distributional and semantic properties of the word, especially in relation to all other word vectors. It has been claimed that in some cases word pairs with very similar relations, such as *Italy*:*Rome* and *Germany*:*Berlin*, might tend

to have very similar differences, so that we could obtain

Italy-Rome+Berlin=Germany,

but it is not clear if this fact can be generalized to other examples. The effectiveness of a vectorial representation through word embeddings depends from the chosen size of word vectors and the function implemented in the neural network to recompute them, but altogether it looks like a very solid basis to expand with clustering techniques for WSI [Pelevina et al., 2016].

1.3.2 Graphs

Graphs are discrete mathematical structures particularly suited to the direct representation of relations between words. Compared to vectorial space representations, graph-based models focus more on how words are interconnected, by means of edges, rather than on how words themselves have to be represented (usually they are just put into a biunivocal relation with the graph's vertices: one word corresponds to just one vertex, and viceversa). This might be interpreted as mirroring the discrete structure of language, where each word represents a single atomic unit of meaning and is part of many local substructures that can reveal its different functions and facets in the global discourse. Formally, a graph G is defined as the couple (V,E), where the vertex set V is any given set (in our case, it nearly always corresponds to the vocabulary of the corpus), and the edge set $E \subset V \times V$ is a subset of the Cartesian power of V, representing all the pairs of connected nodes (see Section 2.1 and in particular Section 2.1.1).

In Word Sense Induction the interest lies in the modelling of relations between words and their relative distribution rather than in the description of syntactical patterns, so that graphs are usually undirected. We can distinguish two main types of word graphs: *co-occurrence graphs* and *semantic similarity graphs* (see Section 2.1.8). An obvious strategy to convey the difference in importance between word relations is to employ weighted graphs. A weight w of $G = (V,E)$ is usually defined as a mapping $w : E \longrightarrow \mathbb{R}^+$ (again, see Section 2.1.1). This makes graph-based models more meaningful, but also represents a further challenge, since many widely adopted graph-theoretical concepts are not well defined in the weighted case, and sometimes their weighted interpretation is outright controversial, like in the case of the clustering coefficient (see Section 2.1.6), as discussed in [Opsahl and Panzarasa, 2009]. A recurring fundamental property of word graphs is that they have been shown to be *small-world* and *scale-free* networks [Ferrer i Cancho and Solé, 2001]. Small-world networks substantially differ from random graphs; they can be defined through the behaviour of the two parameters of clustering coefficient and

characteristic path length, as described in [Watts and Strogatz, 1998] (see Section 2.1.7). Graph theoretical concepts, different types of word graphs and methods to induce them from text will be discussed more in detail in Chapter 2, in Section 2.1. Here we are going to briefly present some graph-based approaches, upon which it will be expanded in Chapter 3.

In [Dorow and Widdows, 2003], following [Widdows and Dorow, 2002], a co-occurrence word graph is constructed by extracting coordinative patterns (`X` *and* `Y`, `X` *or* `Y`,...) from the text and using them to define edges between words (Section 3.2). This operation can be performed globally to model the whole text, but, to disambiguate target word v, only its local subgraph (its neighbourhood) is taken into consideration. The aim is to cluster the node set of this subgraph via the Markov Cluster algorithm (MCL) of [van Dongen, 2000], originally developed in the field of biology, but adapted to many graph-based tasks (Section 3.1). The MCL algorithm is based on random walks and on the assumption that denser, more connected regions can be detected by a higher probability of staying there when starting from one of its nodes. The same denser regions represent the possible senses that v can have in the document.
A more sophisticated approach called *HyperLex* was proposed in [Véronis, 2004] (Section 3.4). Again, a co-occurrence graph is built, using paragraphs as the text units within which to measure co-occurrences. The weights are distances, or better dissimilarities between word pairs, computed from their frequency and co-frequency. Given the local co-occurrence graph G of a possibly ambiguous word v, HyperLex's aim is to exploit its small-world property to identify "sense hubs", i.e. the most important node in a dense region that likely represents a sense of v. Motivated by the skewed degree distribution of a small-world graph, it is made the assumption that the neighbourhood of a hub node in G corresponds to a sense cluster of v. Progressively identifying hubs and removing their neighbourhoods, the graph is partitioned into its senses. Some heuristic corrections, based on thresholds on weights and number of neighbours, are taken to improve the overall sense induction.
A known and efficient clustering algorithm for WSI that is inspired by MCL is *Chinese Whispers* (CW, Section 3.3), which was first described in [Biemann, 2006]. It can be considered a simplified version of MCL, similarly simulating the flow of information in a network. Initially, every node in a word similarity graph starts as a member of its own class; then, at each iteration every node assumes the prevalent class among those of its neighbours, measured by the weights on the edges incident to it. This algorithm is not deterministic and may not stabilize, as nodes are accessed in random order. However, it is extremely fast and quite successful at distinguishing denser subgraphs. The resulting clustering is generally relatively coarse. Besides its use in the field of WSI, CW was also used for language separa-

tion and word class induction tasks.
A clustering algorithm that splits the word graph based directly on its structure is presented in [Hope and Keller, 2013], where it is called *MaxMax* (Section 3.5). The originally undirected local co-occurrence graph G is rewritten as a directed graph H where an edge exists and goes from v to w if and only if its weight is the maximal one of all the edges departing from w in G. The derived graph H can then be split into quasi-strongly connected components, which represent different senses of the target word and may overlap.

We notice that especially the last three methods that we discussed in this section can not be reinterpreted by means of vector space models.

1.3.3 Unsupervised Evaluation

Evaluation in the field of unsupervised Word Sense Induction and Discrimination is difficult, the same way it is for clustering evaluation in general. An internal evaluation[12] is not of great interest in our case, since we want to compare the results of a clustering with a pre-existing notion of word senses, i.e. we want to make use of external knowledge to produce a sensible evaluation. To perform this kind of external evaluation, manually obtained gold truth is required. Here we have to make a distinction between mere Word Sense Induction and Word Sense Discrimination (defined respectively as Task 2 and Task 3 in Section 1.3). In the latter case, evaluation is more straightforward: We can easily visualize it as the comparison of two different labellings of term occurrences in a document, and thereby apply standard procedures, such as the computation of precision, recall and their resulting F-score (see for a reference [Martin and Jurafsky, 2000]). However, the usual F-score and its variants, like the paired F-score [Artiles et al., 2009], could be biased towards very coarse clusterings, especially considering the fact that the sense distribution of a word tends to be skewed in favour of one sense (see Section 1.1.1).

On the contrary, other scores hailing from Information Theory, as in the case of Normalized Mutual Information[13] (NMI, Section 2.2.1.1) [Strehl, 2002], show a similar, if opposite behaviour of preferring very fine clusterings (see Section 5.4.1.1). This can be seen from how both scores rank the two standard baselines: the *most-frequent-sense* baseline (MFS), which

[12] We say that an evaluation is *internal* if it is based only on the same data (elements, similarity scores, and so on) that were used also for clustering, whereas an external evaluation also exploits further data.

[13] Often also called V-measure.

assigns just the prevalent sense to each occurrence, and the *one-sense-per-word baseline* , where each occurrence obtains its own label. Especially the MFS baseline has proven to be hard to beat, and the F-score in particular would value it greatly, as opposed to NML.

To account for the bias and a possible randomness in the overlap of the two compared labellings, other scores like the Adjusted Mutual Information (AMI) [Vinh et al., 2009] were proposed (Section 2.2.1.2). There is also another family of metrics, the BCubed metrics [Bagga and Baldwin, 1998] (Section 2.2.2), whose principle is to compute a mean of precision and recall relative to each element in the clustering. In [Amigó et al., 2009] it is demonstrated how its properties are ideal under many aspects, but from experimental evaluation it still seems to be subject to a bias (again, see Section 5.4.1.1). We can conclude that it is very difficult to strike a balance in the field of unsupervised clustering evaluation, and the matter remains quite controversial, as with how fine-grained a clustering should be (see again Section 1.1.1). For example, going back to the case of evaluation for Word Sense Induction, given the aforementioned evaluation scorings, the definition of a ground truth already looks problematic: How can we define what is a right subdivision in sense clusters of a word's context? To this end many strategies have been developed, some with the aim of creating a synthetic evaluation framework, as in the case of pseudowords [Gale et al., 1992b, Schütze, 1992], which will play a major role in Chapter 5.

We will give a more detailed insight into some of the cited evaluation measures in Section 2.2 and use and discuss them extensively in Chapter 5.

1.3.4 An unsupervised dilemma: parameters and external tools

Here we want to remark a fundamental conceptual issue that is shared by all unsupervised approaches: the use and tuning of parameters. At the beginning of this section we stated that unsupervised methods differ from their supervised counterparts in that they refrain from exploiting any kind of external knowledge, represented by human-compiled resources. This independence from outside inputs is parallel to the avoidance of prior assumptions on the discrimination task. For example, a recurrent feature of unsupervised WSI and WSD clustering algorithms is that they do not need to specify the number of senses that a word can assume, and consequently the number of clusters that have to be found. Still, under these theoretical premises it is often (if not always) the case that unsupervised systems make a rather generous use of parameters and thresholds. In the case of an algorithm, a parameter is by definition an arbitrary constant which characterizes its execution and determines its outcome. Now, we could argue that parameters are just postponing an informed (and thus super-

vised) decision to a higher level of the algorithm. Let us take DBSCAN [Ester et al., 1996] as an example: This clustering algorithm is driven by two parameters, a minimum distance ε and a minimum number of neighbours n, which together are used to define an equivalence relation that produces a clustering. A greater ε and a smaller n will ensure that fewer clusters are found, and viceversa. So, we can actually state that the final number k of found clusters can be expressed as a (non-linear) function $k(\varepsilon,n)$: by tweaking the parameters we control the outcome. The question arises: how do we decide how to set them? From a strictly unsupervised point of view, it emerges a basic contradiction: the notion of tuning a parameter to obtain optimal results implies reasonings that stray from the mere execution of the algorithm and do involve external knowledge. The same holds for the concept of thresholds, which are used to shape the results of a clustering by setting arbitrary or heuristically determined limits. What we are arguing here is that such parameters and thresholds represent supervised elements that are imposed on the structure of the data to be clustered, whereas we would expect a pure unsupervised approach to discover patterns by tapping exclusively from such structure. Instead, very often the introduction of (numerous) parameters can be seen just as an unintended shift of required external intervention from the training step to the inference step: from an *a priori* set up to an *a posteriori* refinement.

Similar remarks could be made about the application in unsupervised approaches of supervised tools for linguistic analysis, such as part-of-speech taggers like TreeTagger [Schmid, 1994] or syntactical parsers like the Stanford Parser [Manning et al., 2014]. If a WSI system relies on modules that make use of external resources, it is itself indirectly using the same resources. Can we still call it unsupervised?

From the perspective stated above, hardly any unsupervised approach can be considered purely unsupervised. To the best of our knowledge, there has been little to no debate in the research community about what we might call the *unsupervised dilemma*, that we phrase as follows:

> Is the use of parameters actually going against the principles of unsupervised Word Sense Disambiguation, and of unsupervised approaches in general? And how much is it really possible to avoid the use of any form of external knowledge and at the same time to get meaningful and interpretable results?

A more thorough investigation of these theoretical matters is beyond the scope of this work. We will just suggest that the attribute *unsupervised* might be seen as referring to the definition of *meaning* and the way to represent it rather than to the actual absence of any supervised element at all.

We conclude noticing that the new methods described in Chapter 4 all try to make as little as possible use of parameters and external tools, in the spirit of a pure unsupervised approach.

1.4 Conclusions

The aim of this chapter is to give a brief overview over the general task of Word Sense Disambiguation and to introduce the complex linguistic phenomena that motivate it, along with the controversies on the notion of word sense (Section 1.1). We shortly describe different approaches to Word Sense Disambiguation, mainly based on whether they make use of external knowledge (supervised approaches, Section 1.2) or not (unsupervised approaches, Section 1.3), and on how they model word contexts.

The rest of our dissertation, after the introductory notions of Chapter 2, will focus on the definition and implementation of graph-based models for Word Sense Induction, first reviewing (in Chapter 3) the graph-based clustering algorithms selected from literature and cited in Section 1.3.2, from which the novel techniques proposed in Chapter 4 take their inspiration, and finally concentrating on the evaluation step in Chapter 5, where we will introduce a novel pseudoword evaluation framework.

Chapter 2

Basic notions

In this chapter we want to introduce the mathematical concepts and instruments that form the basis of the discussion in the second part of this dissertation (Chapters 3, 4, 5). First, Section 2.1 will mostly focus on some basic definitions and results of graph theory that form the backbone of graph-based WSI approaches, and Section 2.1.8 in particular will briefly deal with the idea of *word graph*. Section 2.2 gives an overview of some measures used to assess the significance of the co-occurrence of two terms and to evaluate the similarity between different clusterings, which will be used extensively in Chapter 5.

2.1 Graph theoretical and mathematical instruments

In this Section we will describe and give definitions of graph-related and other mathematical objects and concepts that will be used extensively in Chapters 4 and 5. For generic references on the subject and basic definitions we point the reader to [Harary, 1969, Berge and Minieka, 1973, Ruohonen, 2013]. For more detailed insights on specific touched-upon topics of this chapter, references will be given in the corresponding section, and others are to be found throughout this dissertation.

2.1.1 Graph basics

We define a graph G as a couple (V,E) of two sets, where:

V is an arbitrary *discrete* set and is called the *vertex* or *node set* of G;

E is a subset of the Cartesian product $V \times V$ and is called the *edge set* of G.

The cardinality $|V|$ of the node set is called the *order* of graph G, and the cardinality $|E|$ of the edge set is its *size*. A subgraph $G' = (V',E')$ of G is a graph such that $V' \subseteq V$ and $E' \subseteq E$. In other words, it has either less nodes (a smaller order) or less edges (a smaller size, and the graph is sparser) than G, or both.

In this work, we will always assume that V is also a *finite* set, and thus G a finite graph. The edge set represents the existing connections between nodes of G. An element (v,w) of E, with $v,w \in V$, can be interpreted as giving a *direction* from node v to node w. If $(v,w) \in E$ does not necessarily imply $(w,v) \in E$, the graph is called *directed*, meaning that a direction, or *orientation*, is associated to each edge. If, on the contrary,

$$(v,w) \in E \quad \Leftrightarrow \quad (w,v) \in E,$$

the graph is called *undirected*. This means that the edge can be traversed equally in both directions. Most of the graphs we will consider will be undirected, and if not said otherwise, a graph will be always assumed to be undirected.

A node v is said to be *adjacent* to w if $(v,w) \in E$, and (v,w) is said to be *incident* to w. Clearly, adjacency and incidence are symmetrical relations for undirected graphs. The *degree* of a node v, written as $\deg(v)$, is equal to the number of nodes adjacent to it, or equivalently of edges incident to it (in directed graphs, a distinction is made between *indegree* and *outdegree*, counting edges that are respectively incident to the node or originating from the node and incident to adjacent nodes). The degree can be considered the most basic measure of centrality on a graph (cf. [Koschützki et al., 2005]): the higher the degree of v, the more connected v is to other nodes (and the more paths will pass through v; see Section 2.1.2). A node with degree 0 is called *isolated*, or a *singleton* (and it forms its own connected component, see Section 2.1.2).

A subset $V' \subseteq V$ always induces a subgraph G' of G, that we denote as $G\langle G' \rangle$, whose node set is V' and whose edge set is defined as

$$E' = \left\{ (v,w) \in E \mid v,w \in V' \right\}.$$

Analogously, an edge set $E' \subseteq E$ induces a subgraph with edge set E', whose node set will be

$$V' = \left\{ v \in V \mid \exists w \text{ s.t.} (v,w) \in E' \vee (w,v) \in E' \right\}.$$

The *density* δ of a graph is the ratio of the size of G, the number of existing edges $|E|$, to the number of possible edges on the node set V. In the case of an undirected graph, the latter is $\binom{|V|}{2} = \frac{|V|^2-|V|}{2}$, so that we have

$$\delta = \frac{2\,|E|}{|V|^2 - |V|}.$$

This value can range from 0 (meaning the graph is totally disconnected, see Section 2.1.2) to 1, which means that all nodes are connected to each other. Such a graph is called *complete*. A complete subgraph of G is often called a *clique*. Many graph theoretical problems revolve around finding cliques in a graph, and many if not all of them have been proved to be NP-hard or extremely difficult to treat (c.f. [Bomze et al., 1999], and [Garey and Johnson, 1979] for a more general reference).

Another particular graph shape is the *star* (sometimes called claw): an n-star is a graph where one single node has degree $n-1$ and the other $n-1$ nodes have degree 1.

Weights can be associated to the edges of a graph. We define them through a *weight function* or *mapping*

$$p : E \longrightarrow \mathbb{R},$$

which associates a real number to each edge. Usually, we will only consider and assume positive weights, i.e. the case

$$p : E \longrightarrow \mathbb{R}^+.$$

A graph provided with a weight function p is called a *weighted graph*, and we sometimes refer to the set of all possible weights, the image $p(E)$ of p, as the *weighting scheme* of G. The weighted counterpart of the degree, i.e. the sum of all the weights on the edges incident to a node v, is sometimes called *strength* of v and written as $s(v)$.

2.1.1.1 The handshaking lemma

A very basic result about node degrees in a finite undirected graph is the *handshaking lemma* or *degree sum formula*. It states that the sum on the degrees of all nodes in graph G is equal to twice the size of G:

$$\sum_{v \in V} \deg(v) = 2\,|E|\,.$$

This occurs because each edge (v,w) is counted twice: once as incident to v and once as incident to w.

The handshaking lemma will be useful for the estimate of time complexities of graph-based algorithms. Its generalization to the sum of a given power n of the degrees $\sum_{v \in V} \deg^n(v)$ is much less immediate, even for $n = 2$; we will deal with this issue in a very limited way in Section 4.1.2.

2.1.2 Paths and connected components

There exists a *path* from node v to node w in G if and only if either $(v,w) \in E$ or if there is a subset

$$\pi_{vw} = \{(v_1,w_1),(v_2,w_2),\ldots,(v_n,w_n)\} \subseteq E$$

of $n \geq 2$ edges such that

$$w_i = v_{i+1} \quad \text{for } i = 1,\ldots,n-1, \tag{2.1}$$
$$v_1 = v \quad \text{and} \tag{2.2}$$
$$w_n = w. \tag{2.3}$$

Clearly, if G is undirected, a path from v to w necessarily implies a path from w to v, namely the same path with reversed order. Intuitively, these conditions say that we can go from node v to node w traversing edges that form a continuous path in the graph, respecting their orientations. The *path length* of π_{vw} is equal to its cardinality.

We define the *path distance* between v and w as the minimum path length among all the paths between v and w, and denote it[1] as $d(v,w)$. If there is not any path between v and w, we conventionally set $d(v,w) = \infty$. We notice that if G is directed, $d(v,w) = d(w,v)$ does not necessarily hold, whereas it does for undirected graphs. So, in general, d is not always a metric on the node set.

A graph is said to be *connected* if and only if there exists a path between all pairs of nodes. A *connected component* of an undirected graph G is a maximal connected subgraph, i.e. a connected subgraph $G' = (V',E') \subseteq G$ such that for all $v \in V'$ and for all $w \in V \backslash V'$ there does not exist any path between v and w. Connectivity is more complex for directed graphs, with more specific notions such as weak or strong connectivity, but we will not delve into them. Just in Section 3.5 we will briefly refer to *quasi-strongly connected components*.

[1] The notation $d(\cdot,\cdot)$ on a graph is usually intended as the path distance. In Section 4.1.1 we will define a different distance and use subscripts to distinguish different distances.

Assuming that G is connected and undirected, we define the *eccentricity* $e(v)$ of a node v to be the maximum path distance from it to any other node:

$$e(v) = \max_{w \in V} d(v,w).$$

Intuitively, the eccentricity of a node is a measure of how peripheral it is in the graph. The *radius*[2] ρ and the *diameter* Δ of G are respectively the minimum and the maximum eccentricity in the graph:

$$\rho = \min_{v \in V} \max_{w \in V} d(v,w), \tag{2.4}$$

$$\Delta = \max_{v \in V} \max_{w \in V} d(v,w). \tag{2.5}$$

The smaller the radius, the more intraconnected the graph; the larger the diameter, the more dispersed the graph and the more peripheral nodes it possesses.

2.1.3 Minimum cut

The minimum cut is a graph theoretical concept (with many variants) that has got particular attention from the field of flow theory on networks (about this topic, see [Ford Jr and Fulkerson, 1969]). We introduce it here because during an early experimental phase we have considered some minimum cut variants of the novel clustering algorithms described in Chapter 4, especially for the gangplank clustering algorithm of Sections 4.2.1 and 4.3.2, as detailed in Section 4.3.2.1 (not limited to the case of disemous terms, like in Chapter 5). Some considerations about its implementation will be made in Section 5.4.3.

In graph theory, a *node cut* of a graph $G = (V,E)$ is defined as a subset $K \subset V$ such that the subgraph

$$G \backslash K = G\langle V \backslash K \rangle$$

has more connected components (Section 2.1.2) than G [Ruohonen, 2013]. If $|K| = 1$, i.e. K consists of a single node, it is called a *cutpoint* [Harary, 1969]. An analogous definition is possible for edges, and in that case one speaks of *edge cuts* and *bridges*[3]. A node cut K that achieves the minimum cardinality in the set of all possible node cuts of G is called a *minimum node cut* and it is not always unique; the same goes for the analogous notion of *minimum edge cut*.

[2] Often written as δ, but we want to avoid confusion with the density of Section 2.1.1.

[3] This meaning of the term *bridge* in graph theory has led us to name the weak connections that will be defined in Section 4.1.4 *gangplanks*.

The problem of finding a minimum edge cut is particularly interesting in the case of a weighted graph. Weights can be reinterpreted as representing the volume of a flow traversing the graph, possibly through directed connections in the case of directed graphs (see Section 2.1.1). The notion of minimum edge cut can then be rephrased as an edge cut of G that has the minimum possible total sum of weights. The *max-flow min-cut theorem* [Dantzig and Fulkerson, 2003] establishes a tight bond between flow theory on networks (see again [Ford Jr and Fulkerson, 1969]) and minimum cuts: given two nodes in V, a *source* v and a *sink* w, the maximum possible flow that goes from v to w corresponds to the smallest total weight of the edge cut, i.e. the minimum edge cut, that leaves v and w in two different connected components. An edge cut naturally induces a node cut: if the edge (v,w) is in the cut, both nodes v and w will be in the matching node cut.

Section 4.3.2.1 will deal with the implementation of the minimum cut variant for the gangplank clustering algorithm.

2.1.4 Matrices associated to graphs

There are two principal kinds of matrices associated to a graph G that we will consider: the *adjacency matrix* C and the *weighted adjacency matrix* A. They are both square matrices of size $|V|$, where the i-th column or row refers to the i-th node of G, having indicized its node set as $V = \{v_1,\ldots,v_{|V|}\}$. By the same logic, the entry corresponding to the i-th row and the j-th column will refer to the edge (v_i,v_j). Also, the general concept of *distance matrix* can be adapted to graphs.

The simple adjacency matrix C of $G = (V,E)$ has only binary entries: $c_{ij} = 1$ if and only if node v_i is adjacent to v_j or equivalently if $(v_i,v_j) \in E$, and $c_{ij} = 0$ otherwise. Of course, if G is undirected C is symmetric (reflecting the symmetry of the adjacency relation). The adjacency matrix reflects the basic topology of G: the sum of the non-null elements on the i-th row of C is equal to the outdgree (see Section 2.1.1) of v_i, while the sum on the i-th column corresponds to its indegree. In the case of an undirected graph they clearly coincide. The adjacency matrix is useful to obtain insights e.g. on the connectivity and the path lengths of a graph through algebraic operations. For example, the (i,j)-th entry of $C^2 = C \cdot C$ corresponds to the neighbours common to v_i and v_j, and if it is not zero it consequently tells us that there exists a path of length at most 2 between the two nodes.

For a weighted graph we can define a weighted adjacency matrix A: we put

$$a_{ij} = \hat{p}(v_i,v_j),$$

where $\hat{p}$ is the extension to $V \times V$ of the weight function p (see Section 2.1.1) such that

$$\hat{p}(v_i,v_j) = p(v_i,v_j) \quad \text{if } (v_i,v_j) \in E \tag{2.6}$$

$$\hat{p}(v_i,v_j) = 0 \quad \text{otherwise.} \tag{2.7}$$

The weighted adjacency matrix does not provide direct information about node degrees like its unweighted counterpart, but instead on node strengths (see Section 2.1.1). Still, the interpretation of the entries of A^2 is similar: a non-null entry shows that two nodes have common neighbours. On particular graphs called Markov networks (see Section 3.1), however, under given conditions the powers of the weighted adjacency matrix represent the probability of a random walk on G to arrive to a node starting from another node.

A distance matrix can be associated to any discrete set where a distance function d has been defined (the notion of distance will be explored more in detail and in a wider context in Section 4.1.1). In the case of a graph, a distance function will be a mapping

$$d : V \longrightarrow \mathbb{R}^+$$

satisfying some additional properties. Then, the distance matrix D has entries

$$d_{ij} = d(v_i,v_j)$$

if and only if $d(v_i,v_j) < \infty$, and $d_{ij} = 0$ otherwise. The most straightforward distance on a graph is the path distance defined in Section 2.1.2, but we will also consider other kinds of distances (see e.g. Section 4.1.3).

2.1.5 Node neighbourhoods and ego graphs

Analogously to the more general concept of neighbourhood, as encountered e.g. on a topological space[4], we can define the *neighbourhood of a node* v of G as any subgraph $G' = (V',E') \subseteq G$ such that $v \in V'$. The neighbourhood of a node can be used to investigate the local properties of G around v. In our work, we will always assume that the graph is undirected and that only a specific kind of neighbourhood is meant, namely the *n-th degree neighbourhood*. For any v, it is defined as the set of all nodes at a maximum distance (see Section 2.1.2) n from v:

$$N^n(v) = \{w \in V \mid d(v,w) \leq n\}.$$

[4] [Arkhangel'skii et al., 2012] is an introduction to these topics.

The first-degree neighbourhood of v thus consists of all nodes adjacent to v and is often simply called its neighbourhood and written just as $N(v)$. It can be also expressed in terms of the adjacency matrix (see Section 2.1.4) as

$$N(v_i) = \left\{ v_j \in V \mid a_{ij} = 1 \right\}.$$

The subgraph of G induced by $N(v)$ is also called the *ego graph* or *ego-network* of v. For degrees greater than 1, we speak of n-th degree ego graphs or ego-networks.

Independently from its definition, a neighbourhood $N(v)$ of v is called *open* if $v \notin N(v)$, and conversely *closed* if $v \in N(v)$. If we need to specify this property in the notation, we will use $N(v)$ for open neighbourhoods and $\bar{N}(v)$ for closed ones.

We can see the notion of n-th degree neighbourhood defining a *neighbourhood function* or *mapping*

$$N^n : V \longrightarrow \mathcal{P}(V),$$

where $\mathcal{P}(V)$ is the power set of V, i.e. the set of all its subsets. We can extend this function to $\mathcal{P}(V)$, i.e. to all subsets of V, not limited to singletons. We define the neighbourhood of a node subset $V' \subseteq V$ straightforwardly as the union of all neighbourhoods of all the elements of V', namely:

$$N^n(V') = \bigcup_{v \in V'} N^n(v).$$

Lastly, we notice that for all degrees $n \geq e(v)$ (see Section 2.1.2) a node neighbourhood is constant, as the nodes farthest from v have already been reached. Moreover, if G is connected we will have $N^{e(v)} = V$.

2.1.5.1 Generalized adjacency matrices

The concept of weighted and unweighted adjacency matrices seen in Section 2.1.4 can be expanded to neighbourhoods of an arbitrary degree as seen in Section 2.1.5. Using the path distance d defined in Section 2.1.2, we call n-neighbours of v all elements of the set $N^n(v)$. Then, we can define the n-th degree adjacency matrix C_n as having entries $c_{n,ij} = 1$ if and only if $d(v_i,v_j) \leq n$, and $c_{n,ij} = 0$ otherwise. This way, the fact that $c_{n,ij} \neq 0$ means that there exists at least one path of length at most n from v_i to v_j.

Analogously, the n-th degree weighted adjacency matrix A_n has entries $a_{n,ij} = d(v_i,v_j)$ if and only if $d(v_i,v_j) \leq n$, and $a_{n,ij} = 0$ otherwise. We remark that the entries in the diagonal will be equal to 0 because $d(v_i,v_i) = 0$ by definition, without implying the absence of a path or an infinite distance.

These definitions imply closed neighbourhoods. If instead we want to represent open neighbourhoods, for the matrix entries we will also have to check that the distance is strictly greater than 0, which corresponds to putting $c_{n,ij}$ or $a_{n,ij}$ for every i.

The matrices defined in Section 2.1.4 can then be considered as first degree adjacency matrices, and conversely an n-th degree adjacency matrix might be considered a truncated path distance matrix. The considerations made in Section 2.1.4 for them can be extended to higher degrees, considering paths of a given length instead of simple edges.

2.1.6 Clustering coefficient

Density, defined in Section 2.1.1, measures how intraconnected a graph is on a global level. Observing that cliques or complete graphs have by definition density 1 and are thus the most possibly dense kinds of graphs, we might roughly define a *dense region* in a graph G as a subgraph G' with density δ' in some sense close to 1. If we focus on the single nodes of G, we can restrict local densities to the densities of first degree closed neighbourhoods. Assuming an undirected graph, in terms of the adjacency matrix C of $G = (V,E)$ (see Section 2.1.4), given $v = v_i \in V$ the size of the induced ego graph (see Section 2.1.5) $G\langle\bar{N}(v)\rangle$ will be

$$\sigma_v = \frac{\sum_{j=1}^{|V|}\sum_{k=1}^{|V|} c_{jk}c_{ki}c_{ij}}{2}. \tag{2.8}$$

The division by 2 comes from the handshaking lemma of Section 2.1.1.1 applied to the ego graph. Then, the local density of v can be written as

$$\delta_v = \frac{2\sigma_v}{|\bar{N}(v)|^2 - |\bar{N}(v)|}. \tag{2.9}$$

In other words, δ_v is the ratio of the number of existing edges in the ego graph of v to the number of possible edges $\binom{|\bar{N}(v)|}{2}$. This quantity is also called *local clustering coefficient* and was introduced in [Watts and Strogatz, 1998] (see also Section 2.1.7). We can associate to G the mean value $\frac{\sum_{v\in V}\delta_v}{|V|}$, the overall local clustering coefficient, which measures how similar to a clique the graph is around a node on average.

The local clustering coefficient defined by (2.9) of course does not take into account the possible weighted structure of a graph. Furthermore, it seems to be biased with respect to the degree of a node [Ravasz and Barabási, 2003]: the local density of a node's neighbourhood tends to be inversely proportional to that node's degree.

Formula (2.8), and in particular the third-degree monomial $c_{jk}c_{ki}c_{ij}$, hints to an alternative definition of (global) clustering coefficient based on triplets and triangles, first found in [Feld, 1981, Karlberg, 1997]. A *triplet* is any connected subgraph of G consisting of exactly three nodes. Only two kinds of triplets are possible: an *open triplet*, with two edges, or a *closed triplet*, with three edges. The *global clustering coefficient* (also called *transitivity*) is then defined as the density of closed triplets with respect to all existing triplets. We define this ratio as

$$K = \frac{\tau_\Delta}{\tau}, \tag{2.10}$$

where τ is the total number of triplets in G, and τ_Δ the number of closed triplets. We can see K as the probability that, given a node v connected to two other nodes w and z, there exists also a third edge (w,z) (so that it might also be considered a sort of conditioned local clustering coefficient). In a complete graph, this probability is 1, as the usual density δ. On the other hand, if G is a random graph [Erdős and Rényi, 1959, Bollobás, 1998], where any two nodes have an independent probability $0 < p < 1$ of being connected, we will have $K = p$ and moreover, K will tend to 0 as the order of the random graph increases and its size remains fixed.

In general, the clustering coefficient (2.10) does not arise from a mean value taken over locally computed coefficients, and thus avoids the degree bias of the local clustering coefficient. However, again K does not have an immediate weighted interpretation, and many alternative weighted versions have been given (see the exhaustive [Opsahl and Panzarasa, 2009], on which we also based this section), though none has emerged over the others.

Despite the unclear nature of a weighted version, the clustering coefficient helps identify some peculiar structural properties of a graph. In particular, it can give a measure of how much a graph differs from a randomly generated one. An interesting and pervasive kind of graph structure is presented in Section 2.1.7.

2.1.7 Small-world and scale-free graphs

In [Watts and Strogatz, 1998], the authors identified the characteristics of a particular graph structure which they named *small world*. Graphs and networks of this kind appear to be pervasive in the description of natural or real-world phenomena like social networks or word graphs [Ferrer i Cancho and Solé, 2001]. Their behaviour differs markedly from that of random graphs (about this topic see [Bollobás, 1998]), and can be seen as

intermediate between regular[5] and random graphs. Many small worlds are also scale-free, a more general property involving the degree distribution.

Informally, we can describe a small-world graph as a **connected**[6] graph composed of small, dense subgraphs connected by *hub* nodes of very high degree. These hubs embody the popular notion of *six degrees of separation* in a social context, originating from the Hungarian novel *Láncszemek* ("Chain links") [Karinthy, 1929] and successively researched in the fields of social, computer and mathematical science: each person belongs to some social group, but every individual in the world can be reached through an average of six common acquaintances.

More formally, according to [Watts and Strogatz, 1998], a connected graph $G = (V,E)$ can be identified as a small world observing the values of two quantities: its characteristic path length L and its overall local clustering coefficient C (already seen in Section 2.1.6 and expressed for a single node by (2.9)). The *characteristic path length* L is defined as the mean value of the path distance (see Section 2.1.2) over all $\binom{|V|}{2}$ possible couples of nodes (since we assume that G is connected, there is a path between each node couple). The parameter L presents a global property of G, namely how closely tied together the graph is, i.e. how far we have to move from one node to reach another on average. The local clustering coefficient, on the contrary, measures how much the graph locally resembles a clique.

A small-world graph is then characterized by a **small** characteristic path length and a **high** overall local clustering coefficient: this formalizes the notion of small, dense regions easily reachable one from another. This is opposed to the structure of random graphs, where, assuming the same order and size, but totally random assignment of edges, L is still small (many shortcuts are randomly created between nodes), but the graph is locally sparse, so that C is also small. On the other extreme, a regular graph will be locally relatively highly clustered, sporting a large C, but will also possess a large characteristic path length, due to the absence of nodes acting as connectors.

A small world could then be described as actually many small worlds held together by hub nodes; while removing a random node will not disconnect the graph in most cases, removing one of the hubs, on the contrary, will split the graph in many connected components of medium size.

[5] A *regular* graph is defined as a graph whose nodes have all the same degree.

[6] If the graph itself is not connected, each of its connected components might separately be a small-world graph.

We remark that the here given definition of small-world graphs does not take into account the possible weighted structure of a graph, but only makes a statement about its topological arrangement. In fact, neither the characteristic path length (see the first part of Section 4.1.1) nor the local clustering coefficient (as observed in Section 2.1.6) have a univocal weighted counterpart. The question if the weighted structure alters the mere topological nature of small-worldness is an interesting issue, that we will however unfortunately not address in this work.

Small-world graphs often appear to show a sort of fractal[7] nature, in that the smaller, denser regions tend to locally reproduce the global small-world structure. The expression *scale-free* [Barabási and Albert, 1999] describes this phenomenon: each sublevel of the mathematical structure behaves like the whole global structure. In a graph, this is associated to a particular degree distribution that follows a power law: the probability that a node of the graph has k connections (i.e. incident edges) is expressed as

$$P(k) = k^{-\gamma},$$

with γ a parameter typically not much greater than 2. This feature is completely non-existent in random graphs. It means that only a vanishing subset of the node set has very high degrees, whereas most nodes have small degrees and thus are confined to small regions, as already observed for small-world graphs.

2.1.8 Word graphs

This section will briefly deal with the broad definition of word graph in its two main versions: the *semantic-similarity-based* and the *co-occurrence-based* word graph. Word graphs are a primitive notion in Word Sense Disambiguation and Induction that is often taken for granted and left to the intuition of the reader. Actually, the modelling of a text in a natural language is a very complex topic that deserves deep attention. Here we will just sketch some of the themes that will recur in the rest of this dissertation; for more accurate treatments, we point among others to [Biemann, 2007] and [Biemann and Quasthoff, 2009] (in [Ganguly et al., 2009]).

We will consider word graphs based on raw text (i.e., not organized in a structure like with HTML or XML), either a single document or many documents in a *corpus*, written in any natural language. The most simple and common form of word graph $G = (V,E)$, and the one that we will adopt throughout this work, is an undirected and usually weighted

[7] On the fascinating topic of fractals and how they can describe naturally arising structures, we point to the classic reference of [Mandelbrot, 1977].

graph (see Section 2.1.1) that identifies the node set V with a portion of the *vocabulary* of the corpus, i.e. the set of unique tokens of one or more particular word classes, eventually lemmatized or normalized to have the same capitalization, depending if we want to consider the strings in couples like *peach*/*Peach*, *peach*/*peaches*, etc. as the same or different words[8]. In Section 4.3.1 we will describe more in detail and comment our particular setting for the implementation of WSI clustering algorithms. For the present discussion, we simply assume that one node of G corresponds to one word. What really distinguishes word graphs based on the same node set V is primarily the way they represent relationships between words (i.e. the edge set E), and secondarily how these relationships are evaluated. Here we want to differentiate co-occurrences and (semantic) similarities.

Word graphs based on *co-occurrences* possess an edge (v,w) between two words $v,w \in V$ if and only if v and w appear at least once in a given textual unit of a document. A *textual unit* might take the form of a sentence, a paragraph, a window of size n, or even a tweet or any other rather self-contained and recurring portion of the text which is deemed to be significant. For example, choosing the sentence as the basic textual unit, a minimal co-occurrence word graph of *They eat the peach* would have e.g. (they,eat) as an edge. We immediately notice that some words are more pervasive than others, such as *the*, so that in the word graph of a longer text the node *the* would have an extremely high degree compared e.g. to nouns like *peach*. The reason is that *the* is a so-called *function* or *stop word* that fulfills a grammatical role and does not really carry a meaning by itself (for more on this topic, see [Lyons, 1968]). In WSI, since we are interested in word meanings, this is seen as a good reason to filter out stop words or to use significance measures that penalize very frequent co-occurrences. Some sort of significance measure is indeed needed to weight the importance of the co-occurrence of two words: the fact that an edge exists is actually a very weak statement, as it does not make a difference between two words appearing randomly once in the whole corpus or of one word appearing every time another one appears. In fact, the most simple significance measure is *frequency*, but there are also more sophisticated quantities, like mutual information (see Section 2.2.1), TF-IDF or log-likelihood. Whatever is chosen as the weighting scheme of the word graph, in the case of co-occurrences we usually speak about *first-order* relations: we weigh an edge judging and using data only from the immediate context surrounding the two involved words.

[8]Tokenization, lemmatization and part-of-speech tagging are nontrivial tasks of Natural Language Processing that we will take for granted and not tackle here. A generic reference about this topic is [Martin and Jurafsky, 2000].

Word graphs based on *semantic similarities* (or on similarities, for short) go a step further. They involve *second-order* relations between words to define the edge set and its weighting scheme. Second-order relations are based on first-order ones to establish not immediately apparent connections between two terms. For example, the fact that we recognize the two sentences *They eat the peach* and *The lioness devoured the gazelle* as essentially similar and that we see a commonality between *eat* and *devour* is not directly inferrable from the observed contexts of the two words, but requires other considerations regarding the syntax of the sentences, the parts of speech of the surrounding words, and so on (compare the discussion in Section 1.1). So, the difference with respect to co-occurrence-based word graphs is that the same significance measures are used not on the words themselves, but instead on the features that describe those words. This logic is at the basis of the weighted Jaccard distance that we will present in Section 4.1.1, which measures the distance of two words according to their respective node neighbourhoods in a co-occurrence word graph, and of the *semantic-similarity-based ego graphs* presented in Section 5.2.1. We are of course not limited to second-order relations, but could go on computing relations of the third, fourth or even higher order, albeit their linguistic and semantic interpretation would not be so clear.

Two word graphs with the same node set V might then have very different edge sets E and weighting schemes. However, they are, in a sense, complementary. First-order relations like co-occurrences take place on a *syntagmatic*, *horizontal* level: in a sentence, words interact by means of position and grammatical mechanisms to express a particular meaning. So, a co-occurrence-based word graph explores the possible and acceptable combinations that can give rise to a meaningful sentence. On the other hand, second-order relations belong to a *paradigmatic*, *vertical* level[9]: they correspond to the associations we make between words that allow us to substitute one term for another leaving the syntagmatic structure of a sentence unchanged. For example, the statement *The lioness devoured the gazelle* could become *The lioness devoured the gnu* without altering the general sense. Therefore, a similarity-based word graph tries to represent the semantic relations between words.

Both kinds of graphs have their purposes in Word Sense Induction, and, while still being both small-world and possibly scale-free graphs, graph-based clustering algorithms react differently to their structures, as will be discussed in Section 5.4.4.

[9]This fundamental distinction was proposed by [De Saussure, 1916]; see also [Lyons, 1968]. Similar considerations are also made in Section 1.1.1.

2.1.9 Other mathematical instruments

2.1.9.1 Partition lattices and refinements

A discrete *lattice*[10] is a discrete set X provided with a partial order, denoted as $\leq$, i.e. a binary relation between elements of X that satisfies for any $x,y,z \in X$:

Reflexivity: $x \leq x$;

Antisymmetry: $x \leq y$ and $y \leq x \Rightarrow x = y$;

Transitivity: $x \leq y$ and $y \leq z \Rightarrow x \leq z$.

Furthermore, for every couple $x,y \in X$ of elements, the lattice possesses also a unique *infimum* and a unique *supremum*, denoted respectively as $x \vee y$ and $x \wedge y$. The infimum of x and y is the largest element z of X such that $z \leq x$ and $z \leq y$, and viceversa for their supremum. A very simple discrete lattice is the set of natural numbers $\mathbb{N}$ with the usual ordering: the infimum of n and m will be $\min(n,m)$ and their supremum $\max(n,m)$.

If X is a **finite**, **discrete** set, we observe that we can associate a lattice structure to the set of all its possible partitions. A *partition*[11] of X is a collection of subsets or *clusters* $\{C_1,\ldots,C_n\}$ such that

1. $C_i \subseteq X$
2. $C_i \neq \emptyset$
3. $C_i \cap C_j = \emptyset$
4. $\bigcup_{k=1}^{n} C_k = X$

for any $i,j = 1,\ldots,n$, $i \neq j$. We write the set of all partitions of X as $\mathcal{C}(X)$, and, even if its cardinality is very large[12], we know it will still be finite. We distinguish two trivial partitions: $\{X\}$ and $\{\{x_1\},\ldots,\{x_k\}\}$, where all clusters are singletons.

The partial order is given by refinement. We say that a partition $K = \{K_1,\ldots,K_m\}$ is finer than another partition $C = \{C_1,\ldots,C_n\}$ (and conversely, C is coarser than K) if every cluster of K is contained in a cluster of C, namely:

$$K \leq C \quad \Leftrightarrow \quad \forall i = 1,\ldots,m\ \exists j = 1,\ldots,n \text{ s.t. } K_i \subseteq C_j.$$

[10] For an exhaustive discussion about the topic of this section we point to [Grätzer, 2011].

[11] Often the terms *partition* and *clustering* are used interchangeably, even if for the sake of precision a clustering might be *soft* or *fuzzy* (in contrast to a *hard* clustering) and admit overlapping clusters (i.e. the third condition of partitions does not need to hold). In this case the term *covering* would be more appropriate (see [Arkhangel'skii et al., 2012]).

[12] The cardinality of the partition set of X, with $|X| = N$, is given by the N-th Bell or exponential number [Bell, 1934], which can be expressed by $\frac{1}{e}\sum_{k=0}^{\infty}\frac{k^N}{k!}$.

The refinement clearly satisfies reflexivity, antisymmetry and transitivity, and since we have

$$\{\{x_1\},\ldots,\{x_k\}\} \leq C \leq \{X\}$$

for every partition C, we are granted to have an infimum and a supremum for every couple of partitions. Indeed, this also grants us to have an infimum and a supremum for every subset of $\mathcal{C}(X)$.

The *partition lattice* $\mathcal{C}(X)$ behaves quite differently from the power set $\mathcal{P}(X)$, i.e. the set of all subsets of X. As the latter, it is complete, but it fails to be distributive, i.e. the infimum and supremum operations $\vee$ and $\wedge$ are not distributive. This means that refinement as a partial order is not comparable to union and intersection, which underlie the partial order of $\mathcal{P}(X)$.

We will use the concept of partition lattice and partition refinement when defining the hyperclustering operator in Section 5.3.2.

2.1.9.2 Mean absolute deviation

The *mean absolute deviation* (MAD) is a measure of statistical dispersion (about this topic we remand to [Dixon and Massey Jr, 1957]) that represents how far the values of an observed random variable stray from a central index (like e.g. the mean) on average.

If $x_1,\ldots,x_n$ are n observed real values, we write their mean as

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and define the mean absolute deviation from it as

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n} |x_i - \mu|\,,$$

i.e. as the average of the absolute values of the differences between mean value and observations. The only case where the MAD is 0 occurs when all observations assume the same value. Otherwise, this quantity is not normalized and its interpretation is susceptible to the magnitude of the x_i's, but it can be normalized e.g. dividing by μ.

We will be interested in using the MAD in Section 5.4.4 on the observed cluster sizes in a clustering, to compare the skewness inherent to each clustering algorithm.

2.2 Evaluation and significance measures for WSI

In Section 1.3.3 we briefly touched the issues related to the evaluation of unsupervised systems for Word Sense Induction. Without further indulging in the theoretical debate, here we want to present and describe in succinct detail some of the more wide-spread evaluation measures cited in that section, besides the canonical F-measure of precision and recall (see [Martin and Jurafsky, 2000]). We will use some of them in our pseudoword evaluation framework (BCubed, Section 2.2.2; NMI, Section 2.2.1.1) and in the creation of our ego graph data sets (LMI, Section 2.2.1.3; see also Sections 2.1.5 and 5.2.1). In Sections 5.4.1 and especially 5.4.1.1, based on the data at our disposal, we will make some considerations about their behaviours.

In an unsupervised setting, in the absence of a ground truth known *a priori* it is useful to be able to compare two different clusterings of the same set (in our specific case, a word set modelled in a given way). The following measures can all be seen as giving an interpretation of what it means for any two sets to be "similar".

2.2.1 Mutual Information

Mutual information (MI) is a quantity that arises from the field of Information Theory [Cover and Thomas, 1991, Shannon, 2001] and is usually interpreted in terms of bits or simply as the indicator of how much information is shared between two random variables. In other words, mutual information measures how much a discrete random variable can tell us about another discrete random variable, and viceversa. The definition of *entropy* at the basis of MI has given rise to a whole family of entropy-based evaluation metrics, some of which are presented in this section.

Mutual Information between random variables X and Y is based on the concept of *entropy*. Mathematically, we define the entropy of X as

$$H(X) = -\sum_x P_X(x) \log P_X(x) = -E_{P_X}(\log P_X), \tag{2.11}$$

where P_X is the probability distribution according to X and E_{P_X} is the expected value with respect to it. The function $H(X)$ satisfies some desirable conditions and measures the uncertainty of X, interpreted (if log is the binary logarithm) as the minimum number of bits necessary to describe all possible outcomes of X. Using conditional probability distributions, the conditional entropy $H(X \mid Y)$ is defined analogously as in (2.11). We note that conditional entropy is not symmetric, i.e. $H(X \mid Y) \neq H(Y \mid X)$. Now,

the mutual information between X and Y can be defined as

$$I(X;Y) = H(X) - H(X \mid Y), \tag{2.12}$$

that is, the uncertainty about X reduced by the knowledge about X given us by Y. It can be shown that by definition

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) && (2.13)\\ &= H(Y) - H(Y \mid X) && (2.14)\\ &= E_{P_{XY}}\left(\log \frac{P_{XY}}{P_X P_Y}\right), && (2.15) \end{aligned}$$

where P_{XY} is the joint probability distribution of X and Y, and P_X and P_Y its marginals. This quantity is absolute, i.e. it depends on the values that X and Y can assume. We notice that very often in practice, if the mentioned distributions are not known *a priori*, their maximum likelihood estimates are used.

2.2.1.1 Normalized mutual information

It can be shown that $I(X;Y)$ is a metric and that it is not bounded. Thus, to be better able to compare the degree of mutual information present between different couples of random variables, it is often practical to use a normalized version, usually falling in the interval $[0,1]$. There is not just a unique possible normalization. However, the one that we will use in Chapter 5 when performing our pseudoword evaluation follows the interpretation of [Strehl and Ghosh, 2002]: there it is observed that

$$I(X;Y) \leq \min\left(H(X),H(Y)\right)$$

and

$$H(X) = I(X,X),$$

so that an analogy can be made with the normalization of an inner product. For example, on a real Euclidean vector space we can define the inner product of $\vec{v} = (v_1,\ldots,v_n)$ and $\vec{w} = (w_1,\ldots,w_n)$ as $\vec{v}\cdot\vec{w} = \sum_{i=1}^{n} v_i w_i$, and the norm of a vector as $\|\vec{v}\| = \sqrt{\vec{v}\cdot\vec{v}} = \sqrt{\vec{v}^2}$. The corresponding normalized inner product will be

$$N(\vec{v}\cdot\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{\sqrt{\vec{v}^2\vec{w}^2}}.$$

On the same note, the definition of *normalized mutual information* (NMI) that we will use is

$$\mathrm{NMI}(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}. \tag{2.16}$$

It has the desired property that $\text{NMI}(X;X) = 1$ and is of course symmetric like I: $\text{NMI}(X;Y) = \text{NMI}(Y;X)$.

Normalized mutual information, more so than its absolute counterpart, can be used to compare any two sets, and we are particularly interested in the case of two partitions (or clusterings) of the same space, where a NMI of 0 represents total disagreement and 1 identity. A clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ of set V can be represented as an ordered sequence of labels $\{1,1,3,5,3,4, \ldots\}$, where the label in position i points to the cluster C_i to which the i-th element of V belongs. These labels can be seen as the observations of a random variable. The normalization makes the final score independent from the particular value associated to a cluster.

Homogeneity and completeness Sometimes NMI is referred to as V-score, as this is the name of an evaluation measure presented in [Rosenberg and Hirschberg, 2007]. However, both measures are one and the same, as e.g. proved in [Remus and Biemann, 2013]. The definition given for the V-measure gives nonetheless an alternative interpretation of NMI: it is the harmonic mean of the *homogeneity* and *completeness* scores of a clustering $\mathcal{C}$ with respect to another clustering $\mathcal{K}$ (often thought of as the ground truth). Homogeneity is similar to purity and is defined through entropy: a clustering $\mathcal{C}$ is homogeneous if the conditional entropy of its clusters with respect to clustering $\mathcal{K}$ is zero. This means that the distribution of the elements in each cluster $C \in \mathcal{C}$ is skewed towards one cluster $K \in \mathcal{K}$, i.e. C contains (nearly) only elements of cluster K. Conversely, completeness is symmetrical to homogeneity and is based on the conditional entropy $H(\mathcal{K}|\mathcal{C})$: this value tells us how much elements of a cluster $K \in \mathcal{K}$ are effectively concentrated in a single cluster $C \in \mathcal{C}$.

We will discuss the properties and biases of NMI in Sections 5.4.1 and 5.4.1.1. The implementation of NMI that we use is that found in scikit-learn [Pedregosa et al., 2011], a Python package.

2.2.1.2 Adjusted mutual information

A further variant of mutual information used in the field of Word Sense Induction is *adjusted mutual information* (AMI). It was proposed in [Vinh et al., 2009, Vinh et al., 2010] and derives from the observation that mutual information tends to assume high values between clusterings with a large number of clusters, even if no real information is shared. Therefore, a correction for chance, using the expected value of mutual information as an index of centrality, can be introduced and the following new evaluation

measure is defined as

$$\mathrm{AMI}(X;Y) = \frac{I(X;Y) - E_{P_{XY}}\left(I(X;Y)\right)}{\max\left(H(X),H(Y)\right) - E_{P_{XY}}\left(I(X;Y)\right)}.$$

Again, as for NMI, this quantity is relative and independent from the exact values assumed by X or Y.

The definition of an adjusted mutual information was strongly motivated by the evaluation of task 14 on Word Sense Induction & Disambiguation at SemEval-2010 [Manandhar et al., 2010]. There, a bias of F-score and NMI (which will be discussed in Section 5.4.1.1) was noticed, after which AMI was employed and a new evaluation (also using BCubed, see Section 2.2.2) was published[13]. However, in our pseudoword evaluation framework (described in Chapter 5) we decided to just use NMI, as the number of found clusters is quite contained.

2.2.1.3 Lexicographer's mutual information

The final variant of mutual information that we will consider and put to use to assess the significance of word co-occurrences or syntactical dependencies (see Section 5.2.1) is the *lexicographer's mutual information*, also called *local mutual information* (LMI) [Kilgarriff et al., 2004, Evert, 2004]. The quantity defined by the relation (2.12) takes into consideration all the values assumed by two random variables, and is the expected value over all elements of the form

$$\log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}, \tag{2.17}$$

where x and y are observations of their respective random variables. The quantity (2.17) is called *pointwise mutual information* (PMI) and measures how significant the co-occurrent observation of two values assumed by X and Y is. If X and Y are independent, their PMI is 0: no information is gained and the co-occurrence of x and y are merely random. On the contrary, the higher the PMI, the more meaningful is their co-occurrence, i.e. a stronger dependence is implied. Again, maximum likelihood estimates are used to approximate these probability distributions.

In a text, we might consider the number of times a given word appears as the value assumed by an underlying random variable, and thus measure the significance of the co-occurrence of two terms. However, PMI has been shown to assign too high scores to word pairs with low frequencies: the measure has too few data to give a right estimate and thus overestimates the co-occurrence. To counterbalance this phenomenon, LMI

[13]This new evaluation can be found at `https://www.cs.york.ac.uk/semeval2010_WSI/task_14_ranking.html`.

is defined as the product of PMI with the frequency of the observed pair (x,y). If we denote the frequencies of words v and w by c_v and c_w, the frequency of their co-occurrence by c_{vw} and the total number of words in the text as N, we will write their lexicographer's mutual information as

$$\mathrm{LMI}(v,w) = c_{vw} \log \left(N \frac{c_{vw}}{c_v c_w} \right). \tag{2.18}$$

This way, it will be given much more importance to word pairs that also appear a significant number of times in the text than to random infrequent co-occurrences.

2.2.2 BCubed measures

BCubed comprises precision and recall metrics, together with their resulting F-score, and its origin can be retraced to the paper by [Bagga and Baldwin, 1998]. In [Amigó et al., 2009], it is claimed that BCubed satisfies many desirable properties, like the *rag bag* one defined by them, as opposed to other metrics like the entropy-based ones explored in Section 2.2.1. However, we will also identify some bias of BCubed when commenting our pseudoword evaluation in Section 5.4.1.1.

The BCubed metrics do not take whole clusters as the object to evaluate, but instead take an average over "local" precision and recall scores associated to each element. Let us assume that $\mathcal{C} = \{C_1,\ldots,C_n\}$ and $\mathcal{K} = \{K_1,\ldots,K_m\}$ are two (not necessarily different) clusterings of a set V (we will assume that they are partitions and thus overlaps between clusters are not allowed). For each element $v \in V$, we will denote with

$$c_{\mathcal{C}}(v) \in \{1,\ldots,n\} \quad \text{and} \quad c_{\mathcal{K}}(v) \in \{1,\ldots,m\}$$

the cluster to which v belongs according respectively to $\mathcal{C}$ or $\mathcal{K}$. Then, we can define two symmetrical quantities for each element v:

$$p(v) = \frac{1}{C_{c_{\mathcal{C}}(v)}} \sum_{w \in C_{c_{\mathcal{C}}(v)}} \mathbb{1}\left(c_{\mathcal{K}}(w) = c_{\mathcal{K}}(v)\right)$$

and

$$r(v) = \frac{1}{K_{c_{\mathcal{K}}(v)}} \sum_{w \in K_{c_{\mathcal{K}}(v)}} \mathbb{1}\left(c_{\mathcal{C}}(w) = c_{\mathcal{C}}(v)\right).$$

If our perspective is to take $\mathcal{K}$ as representing the (true) classification of the elements of V to which we compare $\mathcal{C}$, we will conventionally call $p(v)$ the precision of v and $r(v)$ its recall. However, if we are instead comparing $\mathcal{K}$ to $\mathcal{C}$, the situation is reversed. In our notation we will consider the first case.

The quantity $p(v)$ measures the ratio of elements in the cluster of v (according to clustering $\mathcal{C}$), including v itself, that also belong to the same cluster of v according to $\mathcal{K}$. For $r(v)$ the computation is reversed and symmetrical, from $\mathcal{K}$ to $\mathcal{C}$.

From here, we compute the averages over V of the two quantities:

$$P(\mathcal{C},\mathcal{K}) = \frac{1}{|V|}\sum_{v\in V} p(v), \tag{2.19}$$

$$R(\mathcal{C},\mathcal{K}) = \frac{1}{|V|}\sum_{v\in V} r(v) \tag{2.20}$$

and call them respectively the BCubed precision and BCubed recall of $\mathcal{C}$ with respect to $\mathcal{K}$ (or viceversa, as observed before). The BCubed scores P and R have an intuitive interpretation: when finding an element v in a cluster C, $P(\mathcal{C},\mathcal{K})$ will tell us how many other elements in that cluster belong on average to the same class of v. On the other hand, $R(v)$ will tell us how many other elements of the class of v we will find in C.

Finally, the BCubed F-score will be the harmonic mean of BCubed precision and recall:

$$F(\mathcal{C},\mathcal{K}) = \frac{2P(\mathcal{C},\mathcal{K})R(\mathcal{C},\mathcal{K})}{P(\mathcal{C},\mathcal{K}) + R(\mathcal{C},\mathcal{K})}.$$

This quantity is symmetric and unique for each couple $(\mathcal{C},\mathcal{K})$.

Due to this symmetricity, we can use the BCubed F-score as a generic similarity measure between two sets, and in particular between two clusterings of the same set. We will do this in Chapter 5 and discuss some properties and biases of BCubed in Sections 5.4.1 and 5.4.1.1. In [Amigó et al., 2009], BCubed measures are easily extended to the case of soft clusterings, i.e. clusterings that are not strictly speaking partitions (see note 11 in Section 2.1.9.1), in which clusters can overlap. We make use of the implementation of this extended BCubed measures found in the Github repository [Hromic, 2015].

Chapter 3

Existing graph-based clustering algorithms for WSI

In this chapter we will detail the functioning of each of five existing graph-based approaches and algorithms used for WSI that we selected from literature, highlighting their stronger points and what might be their weaknesses, and introducing the recurring *leitmotifs* that we encounter in Word Sense Induction. The final Section 3.6 recapitulates the characteristics of the examined algorithms and gives the motivations for presenting our novel clustering algorithms in Chapter 4.

3.1 Markov Cluster Algorithm

The Markov Cluster Algorithm (MCL) is a generic clustering algorithm based on flow simulation originally expounded in [van Dongen, 2000], where it was first applied in biology for the detection of protein families [Enright et al., 2002], but subsequently found a great range of applications wherever graph clustering is needed, including the field of Word Sense Induction. The DW algorithm in Section 3.2 is one of the systems making use of it.

MCL is based on what the author calls the *graph clustering paradigm*: A random walk in a graph that starts in a dense region (see Section 2.1.1) is likely to leave that region only after many iterations. From the point of view of (information) flows in networks (cf. Section 2.1.3), this means that the flow should be stronger inside a dense region and weaker between two such regions; if strong flow can be emphasized and at the same time weak flow deemphasized up to the point of being removed, the natural cluster structure of the graph will result from the regions still connected by a flow.

Inspired by previous notions of *combinatorial clustering* (for an overview on this topic see [Levin, 2015]), which determines if two vertices belong to the same cluster based e.g. on a given number of connections between them, the MCL algorithm accomplishes to simulate flow by transforming the graph into a Markov network. A *Markov network*, or *Markov random field*, can be defined as a weighted graph where for every node the weight sum (i.e. the strength) on its outgoing edges is exactly 1 (see [Kindermann and Snell, 1980]). We notice that an undirected graph G, like the typical word graph, can be transformed into a Markov network by making it directed and rewriting each edge as a couple consisting of an outgoing and an ingoing edge. The Markov Cluster Algorithm is then driven by an algebraic process on the weighted adjacency matrix M (see Section 2.1.4) of this newly obtained Markov network. This matrix M represents a Markovian random walk[1] on the original graph G, taking the weights of G as a measure of how much one node is attracted to another, i.e. m_{ij} measures the probability that a random walk starting from i will make a step towards j. The sum over each column of M is 1; such a matrix is called *column stochastic*. Clearly, M is no longer symmetric like the original weighted adjacency matrix A of G, although it is still diagonally similar to it, i.e. it exists a *normalizing matrix* Δ such that $M = A\Delta$.

The k-th power of M is still a column stochastic matrix. This means that an entry m_{ij}^k represents the probability that a random walk starting from node i will wind up in node j after exactly k steps. According to the graph clustering paradigm, we expect the probabilities inside a dense region to be higher than between different dense regions. For $k \longrightarrow \infty$ the matrix M^k possesses a limit, either in the form of a matrix M^∞ or of a cyclical sequence of n matrices $M_1^\infty, \ldots, M_n^\infty$. Under some very weak assumptions[2], verified in practice for every word graph, the limit will be a single matrix. Its columns are all equal, representing the state of equilibrium of the random walk. This homogenous limit matrix is not meaningful to our ends, since it simply asserts that eventually every node will be equally attracted by any other node. To identify the desired natural cluster structure of the graph, the MCL algorithm intervenes in the Markov process by applying an *inflating operator* Γ_r at some time point k of the random walk when contrasts in the values of the columns are still sensible, with the aim of accentuating strong transition probabilities (i.e. strong flows) at the expenses of weaker ones. The operator Γ_r acts non-linearly on the columns of M^k, taking the r-th power of each entry and rescaling each column to

[1] A good reference for these kinds of processes is [Norris, 1998].

[2] All the components of the graph G have to be regular and ergodic. Here, differently from Section 2.1.7, a graph is regular if the greatest common divisor of the set of lengths of all its circuits is 1, and it is ergodic if from every node there is a path to any other node of the graph (see Section 2.1.2).

obtain again a column stochastic matrix. Values of r between 0 and 1 have the effect of increasing the homogeneity, whereas values greater than 1 increase the dishomogeneity, in the sense that the gap between higher and lower transition probabilities grows. The resulting matrix $\Gamma_r(M^k)$ can be taken as a new random walk process and iterated again to strengthen the stronger flows.

The MCL algorithm thus consists of two cyclically repeated phases: an *expansion step*, iterating k times the random walk on G, and an *inflation step*, dishomogenizing the transition matrix M^k taking the r-th power of its entries and rescaling its columns. It is shown, heuristically for the generic case and with the aid of mathematical results for specific starting assumptions, that eventually also the MCL process converges to a matrix M^∞, an equilibrium state where the remaining non-zero entries represent the strongest attractions between nodes. The number of necessary iterations for convergence is between 10 and 100. On the basis of theorems proven in [van Dongen, 2000], the structure of M^∞ corresponds to a (possibly fuzzy; see footnote 11 in Section 2.1.9.1) clustering of G, determining which nodes are at the center of dense areas and which nodes are attracted by them. Usually both parameters k and r are taken to be 2. The granularity of the final clustering is strictly dependent on the value of r: the higher its value, the more weaker flows are penalized and eventually removed. For this reason, a higher value of r causes a greater splitting of G and a more fine-grained clustering.

All in all, the MCL algorithm is very flexible, elegantly defined and based on strong mathematical foundations. The complexity of its straightforward implementation is $O(|V|^3)$, where $|V|$ is the number of nodes in the graph. However, actually used implementations perform a column-wise pruning of the intermediate matrices, retaining only the c largest values. The parameter c normally falls between 500 and 1500. Although the resulting process does no longer coincide with the exact mathematical definition of MCL, convergence to a limit is still granted and sped up. The order of its time complexity falls then between $O(|V|\,c^2)$ and $O(|V|^2 \log(c))$ for very dense graphs. We notice that pruning might introduce a slight non-deterministic factor to the Markov cluster algorithm if a choice has to be made between two equally retainable or discardable values: different runs might yield slightly different results.

The MCL algorithm has found use in many fields of application where graph clustering is relevant. However, the author reports that its behaviour can be quite unpredictable when the graph has a big diameter and is very dense, even if its degrees and weights have a homogeneous distribution. In such cases, it becomes very susceptible to small variations in the graph structure and outlier nodes, leading to perturbations in the clustering and

a greater-than-expected splitting of the supposed natural clusters. Also, the coarseness of the "natural cluster structure" found by the algorithm is very finely regulated by the inflation parameter r, which needs to be set *a priori*, a fact that is not always obvious, especially when handling word graphs with unknown patterns. This problematic actually pertains to the generic issues encountered with the unsupervised nature of clustering, as discussed in Sections 1.1.1 and 1.3.4, and the need to choose the best r (alongside the pruning parameter c) just remands to an *a posteriori* tuning and to the necessary use of external knowledge. The very generic nature of MCL, despite being one of its strong points, means that its results applied to Word Sense Induction depend more on the way the underlying word graph was built than on the mathematical intuitions that lead to its definition. In other words, using the MCL algorithm as the last step in a WSI process requires a big focus on the construction of the context-modelling word graph. Under this light, we notice that the effectiveness of the MCL algorithm on small-world graphs (see Section 2.1.7) has not been sufficiently investigated. In a small-world graph the proposed graph clustering paradigm does not seem to hold quite that clearly: the hub nodes are natural attractors, so that the flow is very likely to move away from denser regions and to be distributed rather homogeneously in the graph. The consequence of this would be an inability to detect a cluster structure, despite the small diameter. In mathematical terms, the inflation step might prove not sufficiently strong to offset the extremely rapid convergence of a Markov chain to its uniform stationary distribution (cf. [Tahbaz-Salehi and Jadbabaie, 2007]). Of course, the weight distribution on edges is an important factor, and again implies that the construction of the word graph has a notably large impact on the MCL clusters.

The MCL algorithm has inspired other systems in the realm of WSI, the most known of which is Chinese Whispers, detailed in Section 3.3.

3.2 Dorow & Widdows

The basic approach presented in [Dorow and Widdows, 2003] (hereafter: DW) is based on the assumption that the local subgraph of an ambiguous word w in a global word graph will naturally decompose in relatively independent regions, each of them representing a different sense that w can assume in the corpus. The paper itself harks back to [Widdows and Dorow, 2002], where an allegedly unsupervised (actually semi-supervised) system for lexical acquisition is proposed.

Instrumental to the execution of the DW method is the way the word graph is built. The aim is to obtain a weighted similarity word graph (see Section 2.1.8) from a corpus, exploiting common lexical patterns of co-ordination and disjunction. In this specific case, only word couples (X,Y) occurring in the contexts "`X and Y`" or "`X or Y`" are considered, and their number is further reduced by requiring both X and Y to be nouns; a part-of-speech tagger is used to this purpose. Other patterns of parts of speech, like verb-noun, where the noun is assumed to be an object, or noun-noun, where the first noun is assumed to be modifying the second one, are all possible, but they are discarded because they are not symmetric relations like those based on the co-ordinating conjunctions *and* and *or*. A symmetric relation naturally induces an undirected graph $G = (V,E)$, where the node set V corresponds to the vocabulary and the extracted couples (X,Y) form the edge set E. The weight on the edge (X,Y) is the number of times X and Y co-occur in one of the contemplated patterns. A preliminary pruning is applied, leaving each node X linked only to its top n neighbours, i.e. the n words that co-occur the most with X. The parameter n is determined by the user of the algorithm as a way to regulate the graph's density. The ratio behind taking into account only co-ordinating patterns is that they often occur in lists and comparisons between similar entities, and as such already infer a coarse first stage of semantic equivalence. Restricting the attention only to nouns is also useful to avoid mixing different relations of more complex types, like the dependence between a verb and its subject, and instead keeps the focus on finding words whose referents share common characteristics (cf. Section 4.3.1).

In their previous work [Widdows and Dorow, 2002] the aim is to produce semantic word classes by gathering a given number m of similar words around one or more seed words. We call $S \subseteq V$ the seed node subset of the semantic class $\mathcal{C}$. Initially, we put $\mathcal{C} = S$. The algorithm progressively adds to $\mathcal{C}$ the best neighbouring node X from $N(\mathcal{C})\backslash\mathcal{C}$ (see Section 2.1.5), where *best* means that X maximizes the ratio

$$\frac{|N(X) \cap N(\mathcal{C})|}{|N(X)|}.$$

This ratio measures how strong the connection of X to the subgraph induced by $\mathcal{C}$ is. After having chosen X, the process is repeated considering $\mathcal{C} \cup X$ as the new set to expand. When m nodes have been added, the algorithm ends. According to the authors, this method is particularly effective in limiting the risk of "contamination" of a semantic class due to wrong associations caused by ambiguity. Expanding the semantic class of *tree*, we might for example end up merging terms relative both to vegetables and to industry. The key consideration here is that a polysemous term like *plant* will act as a connector between all the different semantic word

classes to which its senses belong; in the word graph, this means that an appropriate neighbourhood of the node *plant* will be disconnected into disjoint components if the node *plant* is removed from it. Each of such components will be interpretable as one of the possible senses of *plant*. Selecting an ambiguous term as the next node to add to a given semantic class could then lead to the introduction in $\mathcal{C}$ of conceptually unrelated words. This is the reason why we can argue about the true unsupervised nature of the method presented in [Widdows and Dorow, 2002]: There is the need to choose good "prototypical" terms as seeds, and the quality of the resulting semantic class will be heavily dependent on this choice.

The prototype theory in semantics and cognitive science [Rosch, 2005] asserts that among the members of each semantic category, like e.g. that of *plants*, there are particular elements that can be defined as more central than others, in the sense that they are more iconic or better embody the concept of that category than other ones in the mind of the speaker. In our example, a *tree* is a more prototypical *plant* than *moss*. On a linguistic level, we can reinterpret a prototypical element as a word that belongs to a given semantic class in a less ambiguous way than other members of the same class. For example, a *screwdriver* will be more easily ascribable to the category of *tools* than a *nail*, which is also often used to refer to a *body part*. Polysemous, hence potentially ambiguous terms can be seen as hubs in the small-world word graph of a corpus. This intuition has been expanded into a synthetic definition of curvature for word graphs [Dorow, 2006] (see Section 3.2.1).

The DW method of [Dorow and Widdows, 2003] tries to attack the hubs of the word graph G locally to induce word senses. Given an ambiguous word w, an appropriate open neighbourhood G_w (see Section 2.1.5) is extracted; subsequently, also all one-degree nodes are progressively removed. Depending on the kind of ambiguity of w, we expect G_w to break down into one or more connected components. In particular, in the case of homonymy we expect clearly distinct disjoint components with relatively high density, whereas in the case of polysemy we are more likely to obtain loosely connected dense regions, since the senses of a polysemous word are often semantically related and distinctions are not as clear-cut as for homonymy (cf. Section 1.1). For example, *wood* as the material and as a forest will probably share at least *tree* as a common neighbour. To identify the dense regions corresponding to the semantic spheres of w, the DW method resorts to using the Markov cluster algorithm (see Section 3.1) with a high inflation parameter r, trying to exploit the fact that the considered word graph has a rather predictable structure. Noticing that the sense distribution is often skewed (cf. again Section 1.1), to prevent a dominant sense from overshadowing less frequent ones, the MCL is run multiple times on the graph, each time finding the best cluster and then

removing its elements from G_w for subsequent runs of the algorithm, until no further clustering is possible. The obtained clusters will represent the possible senses of w. Thus the time complexity of this WSI clustering approach is proportional to that already given for MCL, repeated a certain number of time on successively smaller graphs. The authors devise to name each cluster explicitly by aid of an algorithm found in [Widdows, 2003] that exploits the broad-coverage taxonomic structure of WordNet, going thus one step beyond mere Word Sense Induction into the step of (supervised) Word Sense Disambiguation.

The intuition behind the DW algorithm could be viewed as the prototypical one among graph-based clustering methods for Word Sense Induction. The different senses of a word will share different contexts, each consisting of terms with numerous interconnections between them, but much more weakly connected to terms outside that context. These assumptions are substantially confirmed by the small-world structure of word graphs [Ferrer i Cancho and Solé, 2001] (see Section 2.1.7). However, we notice that the ease with which a word graph G can be decomposed into sense clusters directly depends on how the graph is built. In the case at hand, the selection criterion of the edge set, based on extracting just co-ordinating patterns, already creates a partition induced by the particular equivalence relation "`X and/or Y`". This procedure, though, is only viable on very big corpora, where such restrictive lexical sequences are more likely to appear in significant numbers. Further, only a fraction of all the nouns will be represented: in [Widdows and Dorow, 2002] it is reported that only a quarter of them makes it into the graph, i.e. only the most frequent ones. More specifically, the authors construct their word graph from the British National Corpus[3], and of around 400000 different noun types therein, the final graph consist of 99454 nodes connected by 587475 edges. Given these premises, the inducted senses will be very coarse, and it will not be possible to disambiguate many less frequent words. Although only two parameters are at work here, the interaction between them is not clear: the pruning parameter n regulating the density of the graph requires a bigger inflation parameter r to separate the sense clusters, but in general we might expect a finer clustering for low values of n, because would already split the graph into many connected components. As always with parameters, there is no clear ideal choice of n. The authors propose an evaluation against WordNet senses, but do not provide exact results. It is questionable how the coarse inducted senses would compare to the very fine-grained distinctions present in WordNet, and how e.g. the absence of the corporate sense of *apple* in WordNet could be handled. Finally, we remark that the DW method seems most suited to a coarse sense induction

[3] `http://www.natcorp.ox.ac.uk/`

on very big corpora, and less adequate for discrimination or disambiguation. As is the case with most similarity-based approaches (here inferred by lexical patterns), the elements of a sense cluster will be in a paradigmatic relation with the examined ambiguous word and therefore most of the time not co-occurring with it (see Section 1.1.1 and Section 2.1.8).

As a final remark, we notice that the definition of particular co-ordinating patterns is heavily language-dependent, and not even easily definable or even existing in many languages.

3.2.1 Curvature reinterpreted

In [Dorow, 2006] the author introduces the geometrical concept of curvature, redefining and applying it to a graph and subsequently using it for ambiguity detection and semantic class acquisition. There, curvature is a quantity associated to a node and identified with its local clustering coefficient. As seen in Section 2.1.6, the clustering coefficient of node w can be equivalently expressed either in terms of density of its neighbourhood, or in terms of possible triplets in which w participates with respect to all possible triplets. Given any two nodes u,v in the neighbourhood of w, these two points of view allow us to see the clustering coefficient $c(w)$ as linked to the average distance $\hat{d}$ between u and v, in the hypothetical triangle delimited by w, u and v, expressed by the formula

$$c(w) = 2 - \hat{d}.$$

So, the larger $c(w)$, the smaller the third side uv of the triangle, and thus the higher the curvature of a space where an equiangular triangle with side lengths 1, 1 and $2 - c(w)$ can be drawn. This point will become more understandable when we present the notion of curvature in paragraph 4.2.3. However, here it suffice to say that consequently, the higher its curvature, the more relevantly w is connected to its neighbours. It follows that a node with low curvature is not particularly tied to any of its neighbours, which could mean that it happens to be a polysemous word: it acts as a hinge between denser regions that represent some senses. So, removing nodes under a given curvature threshold will fragment the word graph into different components, that we see as semantically homogenous regions of the graph. Thus, in [Dorow, 2006] curvature is a way to cluster the graph and to identify ambiguous words. We are not quite convinced of this synthetic interpretation of curvature, in that it reduces simply to considerations of density and does not take weights into considerations, and still needs a threshold to obtain a clustering.

3.3 Chinese Whispers

One of the most known and used graph-based algorithms in the field of Word Sense induction is *Chinese Whispers* (CW) [Biemann, 2006], for its speed and relative simplicity. The name is a hint to the children game where a word is whispered from one participant to another until it comes to the last one as a different word, transformed by the small distortions during its transmission. The aim of the algorithm is to simulate the diffusion of the information carried by each node in the network to determine how it is distributed and what regions share the same information at the end of the process.

Chinese Whispers is directly inspired by the Markov cluster algorithm of Section 3.1 and could be considered a simplified version of it. As discussed there, the cycles of expansion and inflation steps define a sequence of matrices converging to a limit. Even if the starting transition matrix of the Markov random field related to word graph G is normally dense, the new matrices in the sequence rapidly become sparser and sparser. To further favour the converging process, the possibility to prune the intermediate matrices leaving only values over a given threshold are discussed in [van Dongen, 2000]. Chinese Whispers brings this to an extreme and substitutes the inflating operator Γ_r with *maxrow*: This new operator acts row-wise instead of column-wise on a matrix M and sets to zero all entries of a row except the largest one. This way, $maxrow(M)$ has exactly $|V|$ non-zero entries, where V is the node set of the graph $G = (V,E)$, the same size of the square matrix M, and induces a hard clustering. The matrix $maxrow(M)$ is then multiplied to the weighted adjacency matrix of G (see Section 2.1.4) for the expansion step, then *maxrow* is applied for the next inflation step, and so on. Convergence is again granted, but instead of a single process-invariant matrix the limit might be a pair of matrices that represent an oscillating state of the clustering, in which some nodes are constantly swapped back and forth between clusters. This, and that of singleton clusters, is partially avoided by the graph-based definition of the Chinese Whispers algorithm.

The Chinese Whispers algorithm is run on a weighted, undirected graph $G = (V,E)$ where weights are intended as similarity scores. Initially, each node represents its own class. At each iteration of CW, every node is visited in random order and its new class is computed. A node w acquires the class that is most represented in its neighbourhood: each neighbour contributes to the strength of its own class by the weight on the edge that connects it to w. The class with the highest score between all the neighbours of w becomes also the class of w. We can see this process as a majority voting. Each nodes exerts on its neighbours an influence equal

to the strength of its connections, and information is passed on along the strongest links. The process is continuous, contrarily to the previously given matricial definition. This means that during the same iteration a class is assigned immediately to a node, and this information is used right away for the rest of the iteration. Homogenous regions tend to appear rapidly and to stabilize when they expand until reaching the boundary of other strongly represented regions. The state of balance that gives the final clustering is attained in few iterations. However, true convergence is often not reached: it can happen that a node w's neighbourhood presents a certain symmetry that makes it equally possible to assign w to either of two or more clusters. In this case, the algorithm will oscillate back and forth between these two states, since the final result has to be a hard clustering. Such situations are however quite marginal in the economy of the whole clustering, and at that point the process can be ended. We notice that the simultaneous updating of node classes and the random visit of the graph make Chinese Whispers a non-deterministic algorithm, like the pruned version of MCL, as discussed in Section 3.1. This has as a consequence that different executions may yield different clusterings.

Because of its nature, time complexity of the CW algorithm is at most quadratic in the cardinality of G's node set, as was the case for MCL. Actually, since in each iteration all edges have to be visited at least once to determine the strongest classes in a neighbourhood, the algorithm has a time complexity of $O(|E|)$. It can be linear in $|V|$ for a very sparse graph and quadratic if the graph is complete. In any case, a pretty stable cluster configuration is reached in relatively few iterations.

Chinese Whispers has been applied to many flavours of Word Sense Induction and previously tested on many kinds of artificially generated graphs. A clear example of graph with separate dense regions is the unweighted n-partite clique, consisting of two cliques of n nodes interconnected on a 1-to-1 basis. If each clique has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges, between cliques we will have n edges. The ratio between inner and outer edges of each clique is thus $\frac{n-1}{2}$ and grows linearly. Experimental results in [Biemann, 2006] show indeed that as n grows, two clusters are nearly always reliably identified, whereas the algorithm struggles for small values of n and often assumes there is only one cluster. Of interest to WSI and NLP in general is the behaviour of Chinese Whispers on small-world graphs. In [Biemann, 2006] it is investigated how successful CW is in discriminating two small-world graphs of equal or very skewed sizes that were merged along different percentages of the nodes of the smaller graph. As expected, the more merged, the more difficultly the two components are recognized, and not surprisingly the more skewed mixtures are the easiest to solve. This highlights a first tendency of CW to provide a coarse cluster-

ing and to identify a bigger, central cluster accompanied by smaller ones, originating from dense and very tightly interconnected marginal regions. Chinese Whispers has been further tested for different NLP tasks, among which for Word Sense Induction. The approach of the DW algorithm has been followed closely (Section 3.2). A global co-occurrence graph is built from the British National Corpus using all co-occurrences and not just co-ordinating patterns. Edges are weighted by log likelihood ratio [Dunning, 1993] and pruned if their weight falls under a given threshold. Then, the open neighbourhood of a given term is extracted and Chinese Whispers used instead of MCL. Comparison with the method described in [Bordag, 2006] show that Chinese Whispers yields results in line with those of algorithms specifically designed for WSI. In [Biemann, 2007] a more thorough analysis is conducted and some ways to approximate a deterministic outcome are devised. A hierarchical adaptation of the algorithm is also presented by means of successive runs of Chinese Whispers on the biggest cluster in the clustering at each step. Clearly, as with every hierarchical approach, the problem is at which level in the hierarchy to stop the splitting. Moreover, in later works different parameters are introduced to allow greater control on Chinese Whispers' clustering process.

Since its conceptual filiation from the Markov cluster algorithm, Chinese Whispers shares many peculiarities with it. It is extremely fast and easy to implement, and the fact that it found so many usages in NLP bears testimony to its versatility. Even adding different kinds of parameters to regulate the output, the basic clustering of Chinese Whispers remains coarse. One can wonder how this and the information distribution concept cope with the structure of one single dense small-world graph to be clustered. Since there are hubs, i.e. nodes with very high degree that reach many other nodes in the graph, it arises the possibility that information starting from a small and very intraconnected subgraph (possibly a clique) can propagate extremely quickly to other parts of the graph by means of the hubs. With a metaphor, if one locally very strongly represented node class can "conquer" the strategic crucial points of a small-world graph, which grant access to most of the other nodes, it will have the opportunity to impose itself on the whole graph, overriding other significantly represented local classes. We have to ask ourselves is this is a desired effect, and of course this claim has to be proved experimentally. However, it gives evidence to the difficulties possibly encountered by Chinese Whispers in detecting less frequent senses in a co-occurrence or semantical-similarity-based word graph. This behaviour can be mitigated by pre-processing the word graph, e.g. pruning it under a given threshold, as it was done in [Bie-

mann, 2006], and making it sparser. On the contrary, Chinese Whispers seems to be less sensitive to outliers with respect to MCL. Of course, these considerations further illustrate how decisive to the final result the form of the underlying word graph is.

3.4 HyperLex

HyperLex [Véronis, 2004] is a system especially developed with the task of Word sense Induction and Discrimination in mind, in particular to organize a web user's queries, and explicitly exploits the small-world and scale-free structure pervasive to word graphs. It comes also equipped with a visualization tool to explore the clustering; unfortunately, this tool does not seem to be publicly available.

HyperLex works on a co-occurrence word graph G. Two words are said to co-occur if they appear in the same paragraph of a text, and they form a couple of nodes connected by an edge in G. Only nouns and adjectives are considered, using specific tools for part-of-speech tagging and lemmatization. A first pruning leaves out contexts, i.e. paragraphs, containing less than 4 terms after filtering, all words with less than 10 occurrences and all co-occurrences with a frequency less than 5, three arbitrarily set thresholds. A weight, intended as a semantic distance, is assigned to each edge, corelated to the conditional probability of observing one word when also the other one appears in a paragraph. This probability is computed by means of maximum likelihood. A value of 0 means that two words always co-occur. The resulting weighted, undirected graph G is again pruned removing all edges with a weight above 0.9, another arbitrarily determined threshold. As is to be expected, G satisfies all the criteria to be considered a small-world, scale-free graph. It is shown that raw frequency of a word is directly correlated to the degree of its corresponding node in the word graph. This can be explained noting that a word which appears very often is more likely to have more significant similarity scores to other words than less frequent terms.

As is common to the other approaches examined in this chapter, the basic assumption of HyperLex is that in the co-occurrence word graph modelling the context of a target word w, its different senses will form very dense regions, loosely interconnected between them. In particular, the starting consideration is that every dense region of this kind will have a component which rises above the others in terms of degree, and that the author calls *root hub*. Thus, reversing this observation, it will be possible to identify the high-density regions by locating each time the node with the highest degree, along with its neighbourhood. As an example, let us suppose that we are examining the local graph G relative to the word

peach. If we determine that the node *fruit* has the highest degree in G, we can isolate it together with all its neighbouring nodes and assume that this is the first dense region, and consequently the first sense, of *peach*. The root hub is seen as the element that holds this component together, the "least common factor". Now, with a take similar to that of the DW algorithm of Section 3.2, removing this subgraph from G means very probably that the remaining graph does not contain this sense anymore, allowing us to select the next node with highest degree as the next root hub and to start again the process. The process is iterated, but some heuristics are introduced to find sensible candidates for the root hubs towards its end, when most of the nodes will already have been removed. It is required that a node have at least 6 neighbours and that its weighted clustering coefficient[4] be above a threshold indicated as 0.8 in the paper; both thresholds were determined experimentally. Further, in the actual implementation of the algorithm the visit in order of degrees is actually approximated by a visit in order of frequencies, which were obtained during the text pre-processing step. At the end of this process, the root hubs, and so the number of sense clusters, will have been identified. In the subsequent step of the algorithm, all the nodes are assigned to their nearest root hub to give form to the sense clusters. Considering target word w as the root of a tree and the retrieved root hubs as its sons, a minimum spanning tree is computed over G. Since the weights in the graph represent distances, the distance between two non-adjacent nodes is just the minimum sum of the weights on a path connecting them. The used minimum spanning tree algorithm, like e.g. Kruskal's [Kruskal, 1956], will then assign each node to its nearest root hub, possibly correcting incorrect assignments of the first step. The whole clustering process has time complexity $O(|V| + |E| \log |E|)$. The first addend comes from the progressive descending visit of the nodes in the first step, and the second addend amounts to the worst case for Kruskal's minimum spanning tree algorithm.

The structure of the so-obtained minimum spanning tree T that describes all the sense clusters of target word w is used to disambiguate a co-occurrence of w in the original text. A score s_i is associated to each node v in T relative to a root hub h_i: precisely,

$$s_i = \frac{1}{1 + d(h_{i,v})}$$

if v descends from root hub h_i and 0 otherwise. The distance function d is defined again as the minimum weight sum on the path connecting the two nodes in T. The score is defined so as to be 1 for a root hub. Given a context of w in the text, each word therein that also appears as a node

[4] As discussed in chapter 2.1.6, there is no univocally defined weighted version for the clustering coefficient. HyperLex makes use of one defined by the author in the paper [Véronis, 2004].

in T contributes to the score of one of the root hubs of w. In the end, the root hub that scores highest will determine the sense of w in that context. HyperLex even takes into account a *reliability coefficient* to measure how confident the disambiguation is, which is defined as $\rho = 1 - \frac{1}{1+\Delta}$, where Δ is the difference between the best and the second best score.

Since one of the goals of HyperLex is to help a web user to organize its online queries, word graphs for evaluation were induced by Internet pages where the target word appears. This led the pre-processing step to include a list of web-specific stop words like *menu* or *home*, that introduces too much noise in the word graph, along with the already present numerous parameters that regulate the pruning of G and the clustering step of the algorithm. Ten target words which had been reported to be problematic for human annotators in [Véronis, 1998] were considered. The algorithm was used to discriminate random contexts of each target word, thereafter computing standard precision and recall scores with respect to the judgment of a single human annotator. High scores were reported for sense tags with a reliability score ρ above 0,5. In general, HyperLex seems to encounter some difficulties detecting very general uses of a word, as it could be for *home* in the expression *be home to*, which can be used in just about any context, but succeeds in isolating more specific uses with a frequency above 5% of the total occurrences of the target word. How many specific senses a word will have and their specificity will depend from the kinds and the specialization of the texts used to build the word graph. To this regard, we want to briefly express some thoughts about the mathematics behind HyperLex. We notice that if the root hub detection process is iterated in its non-approximated form (i.e. degree is used instead of frequency with no heuristics) until no more clusters can be isolated, an *independent dominating set*[5] of G will be found. From graph theory it is known that such set is also a *minimal dominating set* and a *maximal independent set*. Although the computation of a minimal dominating set with minimum cardinality is an NP-hard problem[6], it is possible to define boundaries for its size depending on characteristics such as the graph's maximum or minimum degree and its degree sequences. This means that in principle it is possible to exactly determine *a priori*, or at least to approximate very closely, the number of sense clusters that HyperLex will find when running on the word graph. At one extreme, if G is complete just one cluster will be returned, which may be an unwanted result. The pruning in the text pre-processing step is introduced also to avoid this inconvenient, but since it alters the

[5] A *dominating set* of graph $G = (V,E)$ is a subset D of the node set V such that each other node in $V \setminus D$ is neighbour to a node of D. In other words, the neighbourhoods of the nodes in D cover the whole node set V. An *independent set* of G is a subset of V consisting of not adjacent nodes. For a complete study on these subjects, see [Haynes et al., 1998].

[6] On the issue of NP problems see [Garey and Johnson, 1979].

degree distribution of the graph, it is possible to measure its impact on the number of found clusters; in particular, the higher the thresholds, the more clusters will be found, even if G remains connected. Apart from this difficultly controllable side effect caused by parameters (compare Section 1.1.1), the definition of this clustering process has as a consequence that the number, and in practice also the structure, of the sense clusters is exactly determined by the mere topology of G and is not influenced by the weighting scheme. Information given by the weights is only used in the pruning and in the discrimination step; the reassignment of nodes to the root hubs will largely correspond to their neighbourhoods, with only minor shifts. This comes out as rather counterintuitive, as weights are introduced in word graphs with the purpose of revealing more about the structure of a word's context, but here they seem to be totally disregarded. Once again the method used for building the graph, along with the tuning of many parameters, crucially determines the final result, in ways which are not always easily predictable.

3.5 MaxMax

Starting from a word co-occurrence model similar to that used by the DW algorithm seen in Section 3.2, *MaxMax* [Hope and Keller, 2013] is a soft-clustering algorithm that exploits the weighting scheme of the word graph and the topological structure of a derived graph. The algorithm itself is very simple and boasts a linear time complexity in the number of edges.

The weighted, undirected word graph G is like in [Dorow and Widdows, 2003] obtained from a part-of-speech tagged corpus by extracting nouns appearing in comma-separated lists of noun phrases and coordinating patterns governed by the conjunctions *and*, *nor* and *or*. This way, though originating from co-occurrences, G simulates a quite sparse similarity word graph, as discussed in Section 3.2. Log likelihood ratio [Dunning, 1993] scores measuring the significance of the co-occurrence between words are used as edge weights.

Given a word w to disambiguate, its open neighbourhood G_w is extracted as w's local graph from the global graph G. The key point of MaxMax is the transformation of this pseudo-similarity local graph into a new unweighted, directed graph that represents semantic dependencies between words. To this end, a notion of *maximal affinity* between neighbouring nodes is defined. A node u is said to have maximal affinity to another node v if the weight on the edge (u,v) is maximal among all the weights on edges incident to u. The node v is then called a *maximal vertex* of u, implying a directed relation from v to u. A node can have more than one maximal vertex. The principle is to obtain clusters made out of just

affine nodes. Now, the original graph $G_w = (V,E)$ can be rewritten as the new graph G'_w with the same node set V, but retaining only the edges between maximally affine nodes, directed so as to go out from the respective maximal vertex. Since G'_w is dispossessed of weights, the information that we can infer from it are merely of a topological kind. In particular, the aim is to decompose the graph into maximal quasi-strongly connected subgraphs. A graph is said to be *quasi-strongly connected* if, for every couple of nodes v and u, there exists a node z, not necessarily coinciding with v or u, such that there is a directed path from z to v and from z to u [Ruohonen, 2013]. A quasi-strongly connected subgraph of G'_w is maximal if no node or edge of G'_w can be added to it keeping it quasi-strongly connected. Every quasi-strongly connected graph has a tree-like structure, in that it possesses a root node from which every other node can be reached through a directed path. Thus, a maximal quasi-strongly connected subgraph of G'_w is determined by all the descendants of a root node, and the number of root nodes is equal to the number of clusters which represent the possible senses of w. The identification of root nodes is done algorithmically by first reducing every possible bidirected edge to a directed edge pointed in either one of its two directions. Then, each node in the graph is marked as a possible root. Iterating on the node set, all the descendants of a node are marked as non-roots. In the end, the nodes still marked as roots will be the actual roots of all the maximal quasi-strongly connected subgraphs of G'_w. It is possible that a node is at the same time the descendant of more than one root, meaning that it belongs to more than one cluster. Therefore, the clustering of MaxMax can often be fuzzy (see footnote 11 in Section 2.1.9.1). Since during the execution of the algorithm each edge needs to be visited only once both to determine its maximally affine node and later to label all descendants of a node as non-roots, the time complexity is linear in the number $|E|$ of edges of G. In practice, this means that for a connected graph its complexity falls between $O(|V|)$ (in a connected component each node has at least one incident edge) and the worst case $O(|V|^2)$.

The root of a MaxMax cluster can be interpreted as its central element, in the sense that it attracts the most affinity among all the terms that co-occur with it and the ambiguous word w. Given its tree structure, a quasi-strongly connected graph defines a hierarchy. Going up in the tree structure, the root is the element to which all its descendant words can be traced back in terms of logical associations (given by maximal affinity). Taking into account the co-ordinating *and/or* patterns that led to the construction of G, let us imagine that *apple* is consistently compared to *pears* and *peaches*, occurring in sequences like *apple and pears* and *apples and peaches*. *Peaches* themselves might be most often compared to *apricots*, e.g. when saying *I like the taste of peaches and apricots*. We explain this with

the fact that apricots and peaches are seen as very similar fruits, and that in turn peaches are compared to apples only in the more generic sense of fruit. So, here we see again the notion of prototype theory (discussed in Section 3.2) emerging: *apple* embodies the protoypical fruit, and (in the ideal case) MaxMax recognizes this by outlining a quasi-strongly connected subgraph through strong similarity associations. Such associations are also interpretable as strong flows in the graph, and under this light the root assumes the role of an attractor node, in a conceptually similar way to the MCL algorithm of Section 3.1. In fact, the connections between the MCL algorithm and MaxMax deserve further investigation, which could highlight if both approaches share the same difficulties. Still, there are differences, e.g. the fact that MaxMax is purely deterministic, does not depend on the convergence to a limit nor does it require the tuning of any parameter.

The previous analysis and motivation of the decomposition into quasi-strongly connected subgraphs might induce to the consideration that the use of MaxMax for Word Sense Induction can not be uncoupled from the underlying word graph used by the authors. In other words, the results of MaxMax are not so readily interpretable on a pure syntactical co-occurrence word graph, and the graph building obviously presents the same criticalities discussed in Section 3.2. On the other side, it does not appear to be too much dependent on a given weighting scheme, since the precise scores do not enter in any calculation, and instead only local maxima are of importance. One wonders if simple frequency scores could be as efficient as log likelihood ratio.

The authors report state-of-the-art results in the application of MaxMax to different WSI tasks: The SemEval-2010 task 14 for Word Sense Induction and Disambiguation [Manandhar et al., 2010] and the induction of WordNet senses on the British National Corpus (for which [Pantel and Lin, 2002] is a reference). We do not wish to enter in the details of the evaluation here, for which we point to [Hope and Keller, 2013], but just limit ourselves to notice that, especially for the SemEval task, the construction of the word graph is quite laborious in terms of feature extraction and their relative pruning by means of parameters, and that the goal of the algorithm actually becomes to discover patterns that substantially have already been laid out by the graph-building process.

3.6 Final considerations

We concisely summarize the main characteristics of the clustering algorithms presented in this section in Table 3.1.

System	Word graph type	Focus	Estimated time complexity
Markov Cluster	any	clustering	$O\left(\lvert V\rvert^2 \log(c)\right)$
Dorow&Widdows	co-occurrences (patterns)	graph building	$k \cdot O\left(\lvert V\rvert^2 \log(c)\right)$
HyperLex	semantic similarities	graph building	$O\left(\lvert V\rvert + \lvert E\rvert \log\left(\lvert E\rvert\right)\right)$
Chinese Whispers	semantic similarities	clustering	$O(\lvert E\rvert)$
MaxMax	co-occurrences (patterns)	graph building	$O(\lvert E\rvert)$

Table 3.1: Characteristics of clustering algorithms on a word graph $G = (V,E)$.

In the column labelled as *Word graph type* we point out the type of word graph for which the algorithm was principally conceived or to which it was first applied in the works describing it. With *semantic similarities similarities* we generically denote second-order semantic relations normally computed from co-occurrences, which are of first order (see Section 2.1.8); with *patterns* we mean that only text portions satisfying specific constraints were considered to extract co-occurrences. In the column labelled as *Focus* we want to indicate how much the clustering process using a given algorithm depends on the form of the word graph itself, or in other words, how much effort is required during the *graph-building* step with respect to the actual *clustering* process. Finally, the *Estimated time complexity* is directly reported from the papers in which the respective authors first introduced the algorithm. For MCL, the estimate for its pruned version is given, and c is the number of retained largest values (refer to Section 3.1). Parameter k in the DW row corresponds to the number of times that MCL is run on the graph; we notice that this estimate is rather high, as it does not take into account that the algorithm is run on increasingly smaller graphs (refer to Section 3.2).

The discussion of the five chosen systems Markov cluster algorithm, Dorow&Widdows, Chinese Whispers, HyperLex and MaxMax gives concrete examples of how graph-based Word Sense Induction is conceived and implemented, and provides a more practical overview of its basic concepts, complementary to the more theoretical discussion of Chapter 1. In Chapter 4 this theoretical and mathematical background (along with that of Chapter 2) will be extended to lay out the bases of our novel proposals. What we want to underline here is the slight prevalence of *graph building* in the *Focus* column. This brings us back to the issue briefly touched upon in Section 1.3.4: to what extent is the actual clustering, i.e. sense induction step, driven by external tools (part of speech taggers, parsers,...) and external knowledge about the world (co-ordinating patterns, pruning thresholds,...)? How prevalent is the pre-processing step with respect to

actual sense induction? We see as the major crux of the **unsupervised** task of WSI the necessity to pass from a raw, first-order representation, i.e. a text where we can only observe co-occurrences, to some second-order structure that can represent some kind of lexical semantic relationships. Chapter 4 will thus have the aim to present definitions and instruments which allow to create a cohesive framework for WSI where text pre-processing is kept to a minimum and clustering algorithms let a natural partition of the word graph arise just by exploiting its structure, and not by forcing one on it. Also, these approaches are developed with the goal to cope with the peculiar small-world structure (see Section 2.1.7) of word-graphs, consisting of high-density subregions connected by nodes with very high degrees, a fact that has not always been considered in the systems that we describe in this section, but which in our opinion greatly impacts clusterings. Further comments on this matter will be made when examining results on our pseudoword data sets, in Chapter 5, particularly in Section 5.4.

Part II

Novel approaches and pseudoword evaluation framework

Chapter 4

Novel clustering algorithms for WSI

In this chapter we will present three novel approaches to graph-based clustering algorithms for Word Sense Induction (Section 4.2 gives their generic descriptions and Section 4.3 details their implementations), along with the intuitions and mathematical definitions behind them, which will be first delineated in Section 4.1 as a theoretical background expanding Section 2.1 of Chapter 2. All these algorithms are run on co-occurrence word graphs with a rather minimal pre-processing and make use of the *weighted Jaccard distance*, defined on the graph's node set, introduced in Section 4.1.1. One *leitmotiv* of the systems presented here is the effort to avoid the use of parameters as much as possible. Instead of acting on the graph by pruning and reducing it prior to the clustering step, as was common to most of the approaches examined in Chapter 3, our pre-processing step aims to keep as much information as possible and to let new one naturally emerge from the data, in a way that allows our clustering algorithm to exploit it. One of the challenges that arises is how to treat very dense small-world graphs preventing the risk of always dumping all the nodes in a single, catch-all sense cluster. The sections about weighted Jaccard distance and the gangplank and aggregative clustering algorithm mainly expand the material presented in [Cecchini et al., 2015] and [Cecchini and Fersini, 2015].

The basic assumption of this chapter will be that the starting word graph is built as a **co-occurrence word graph** (see Section 2.1.8). Of course, the clustering module of each algorithm can be applied to different kinds of graphs and weighting schemes than the ones proposed here, and the interest and possible applications of the instruments developed in Section 4.1 are not limited to the use we make of them in this work.

4.1 Concepts and instruments

We will first introduce and discuss the concepts and mathematical definitions which lie at the core of all the graph-based clustering approaches that will be presented later in this chapter. Of central importance is the *weighted Jaccard distance*, into whose ramifications we will delve extensively in the next section. A distance allows us to compute second-order relations and to perform the clustering on a distance graph or a metric space. Ours is not the first generalization of the Jaccard coefficient to the weighted case, but to our knowledge it is the first one applied to a graph and examined in detail. Later, the simple concept of *gangplank edge* is also introduced as a device to identify clusters in the node set of a graph. Each algorithm will give to these instruments more or less focus, according to the algorithm's strategy.

4.1.1 Definition and properties of Jaccard distances on graphs

Given a weighted, undirected graph $G = (V,E)$, we want to define a distance function d between nodes. Generic edge weights might already give a measure of how similar two words represented by adjacent nodes are, but similarities can not be used as a distance function. Similarity is indeed the opposite of distance: the more similar two words, the greater their similarity and the smaller the distance between them, representing their closeness down to the minimum of 0. Also, when comparing two non-adjacent nodes v and w in G, there is no clear way to take advantage of similarity weights. Sometimes the sum of similarity scores on the shortest path between v and w is used to this end, but we comment that this quantity is not readily interpretable under different aspects. First of all, the shortest path in the case of a co-occurrence word graph does not have a semantic interpretation beyond a path length of 2 (see Section 2.1.2). It just tells us that a sequence of text units is linked by the pairwise co-occurrences of two or more words, but apart from that, there is no clear logical connection. Co-occurrence does not necessarily imply semantic relationships like meronymy or synonymy, and even if this is the case, there is no way to tell these kinds of relationships apart from one edge in the path to another. Co-occurrences do not establish followable hierarchies as is the case for WordNet [Miller, 1995]. Moreover, the sum of similarity scores is not well defined: similarity does not add up, because it is strictly local for any pair of elements.

The motivation to define a distance over a graph where some kind of similarity score is already present in form of edge weights lies in the possibility of highlighting second-order relations between words. As we did at the end of Section 3.2, we can argue that two words sharing a semantic sphere are very likely in a paradigmatic relation (see Section 1.1.1 and Sec-

tion 2.1.8). This means that they will appear in similar contexts more often than they appear together in the same context, as they carry out the same role. In a co-occurrence graph, sharing the same context can be expressed as "having common neighbours". We will use this concept to define our distance function.

We recall the conditions that a real function $d : V \times V \longrightarrow \mathbb{R}$ has to satisfy to define a distance, or metric, on V, for any choice of v, w, z:

Positivity: $d(v,w) \geq 0$

Reflexivity: $d(v,w) = 0 \Leftrightarrow v = w$

Symmetry: $d(v,w) = d(w,v)$

Triangle inequality: $d(v,w) + d(w,z) \geq d(v,z)$

Positivity and symmetry conditions are generally met for any positive weighting scheme on an undirected graph, and the same goes for reflexivity. However, triangle inequality is the most crucial point. One function satisfying all these conditions is the *Jaccard distance*, derived from the Jaccard coefficient[1]. It is defined between two finite sets A and B as

$$d_J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}. \tag{4.1}$$

The Jaccard distance is limited to the interval $[0,1]$. The *Jaccard coefficient* $\frac{|A \cap B|}{|A \cup B|}$ measures how much two sets overlap with respect to their combined total size; in other words, how relevant their common elements are compared with the elements that respectively distinguish A from B. If $A = B$, all elements will be in common, so that their overlapping coefficient is 1 and consequently the Jaccard distance 0 (satisfying reflexivity). A coefficient of 0 is attainable if and only if $A \cap B = \emptyset$, and the corresponding distance will reach its maximum value 1. This value can be seen as a limit. Assume that A and B are of the same cardinality n and that they share just one element. Then we will have

$$d_J(A,B) = 1 - \frac{1}{2n-1} = \frac{2n-2}{2n-1},$$

so that for $n \longrightarrow \infty$ we will have $d_J(A,B) \longrightarrow 1^-$. This liminal discontinuity between $d_J(A,B) < 1$ and $d_J(A,B) = 1$ leads us to treat the maximum distance as the *infinite distance*. This consideration will be used later.

[1]The same coefficient is sometimes called the *Tanimoto coefficient*, or alternatively *Tanimoto measure*, or even *similarity* [Tanimoto, 1958].

Whereas the Jaccard distance obviously satisfies the first three criteria of a metric, the triangle inequality for it is not so easily deducible from just simple computations. A way to show it is using the Steinhaus transform [Clarkson, 2006]. If (X,d) is a metric space with distance d, for any $z \in X$ the Steinhaus transform of $d(x,y)$ is

$$\hat{d}(x,y) = \frac{2d(x,y)}{d(x,z) + d(y,z) + d(x,y)} \quad ,$$

and it holds that $(X,\hat{d})$ is again a metric space, that is, $\hat{d}$ is a metric (this can be shown with simple algebraic operations). We define the *symmetric difference* $A \triangle B$ of two finite sets as the set of the elements that appear in either of the two sets, but not in both:

$$A \triangle B = \{x \mid x \in (A \cup B) \backslash (A \cap B)\}.$$

The quantity $|A \triangle B|$ then measures how dissimilar two sets are, and is itself a (non-bounded) metric on a space of finite sets. We have

$$|A \triangle B| = |A \cup B| - |A \cap B| = |A| + |B| - 2\,|A \cap B|\,,$$

so $|A \triangle B| \leq |A \cup B|$. Further, the symmetric distance is both commutative and associative (this descends from commutativity and associativity of the set union operator $\cup$) and the empty set is a neutral element for symmetric difference. So we have $A \triangle A = \emptyset$ and consequently

$$|A \triangle C| = |(A \triangle B) \triangle (B \triangle C)|\,.$$

Then, we have the following chain for any three sets A, B and C:

$$|A \triangle B| = |(A \triangle B) \triangle (B \triangle C)| = |(A \triangle B) \cup (B \triangle C)| \leq |(A \triangle B)| + |(B \triangle C)|\,,$$

showing that the triangle inequality holds. Now, if the Steinhaus transform is applied to $|x \triangle y|$ setting $z = \emptyset$, using the previously shown properties of the symmetric difference we will have:

$$\begin{aligned}
\widehat{|x \triangle y|} &= \frac{2\,|x \triangle y|}{|x \triangle \emptyset| + |y \triangle \emptyset| + |x \triangle y|} \\
&= \frac{2\,|x \triangle y|}{|x| + |y| + |x \triangle y|} \\
&= \frac{2\,|x \triangle y|}{2\,|x \cup y|} = \frac{|x \triangle y|}{|x \cup y|} \\
&= \frac{|x \cup y| - |x \cap y|}{|x \cup y|} = 1 - \frac{|x \cap y|}{|x \cup y|}.
\end{aligned}$$

Therefore for finite sets the Jaccard distance is the Steinhaus transform of symmetric difference, and since the latter is a metric, the Jaccard distance is a metric too, with the advantage of being bounded.

We wish to apply the Jaccard distance on a word graph G. As sets, we will consider the first-degree closed neighbourhoods of the nodes. For $v \in V$ we write $\bar{N}(v)$ (see Section 2.1.5). Then, similarly to (4.1), we can write:

$$d_J(v,w) = 1 - \frac{|\bar{N}(v) \cap \bar{N}(w)|}{|\bar{N}(v) \cup \bar{N}(w)|} \tag{4.2}$$

for any two nodes v, w in V. We are using closed instead of open neighbourhoods so that $d_J(v,v)$ be defined and to avoid some counterintuitive behaviours, as will be shown later. From a lexical point of view, the neighbourhood of a node represents the context of its corresponding word, prosaically defined as all the words co-occurring with it in some text unit. The idea is to define the distance of two words as a quantity **proportional to their shared contexts**. The Jaccard distance is able to capture the overlapping of two contexts and has the advantage of being bounded. As mentioned before, d_J is a second-order relation between the nodes of the graph: two nodes can have common neighbours, and so a non-1 Jaccard distance, even if they are not adjacent. Let us suppose that a corpus contains the terms *peach* and *apple*. It is possible that both appear in sentences such as *Johnny likes eating peaches/apples* or *Apples/Peaches grow on trees*[2]. Their Jaccard distance is then more significative than their mere raw co-occurrence frequency or frequency-based similarities, which might be quite low if not enough sentence of the type *Johnny likes peaches more than apples* are present. So, what d_J does is taking an already existing local measure of the correlation of two words (such as the aforementioned frequency or log likelihood) and generalizing it to find deeper analogies. On the contrary, if two words v and w never appear together and they do not even share any neighbour, their neighbourhoods will be disjoint and $d_J(v,w) = 1$. The fact that their neighbourhoods do not overlap means that v is not reachable from w. This is a further motivation of why we can regard two nodes at distance 1 as "infinitely distant".

Observation 4.1.1. Before delving into the properties of the Jaccard distance on graphs, we first need to make a subtle formal observation. The function d_J as we defined it on the node set V is a distance actually only on the node neighbourhood set of G, but not on its node set. In Section 2.1.5 the neighbourhood function N is defined as $N : V \longrightarrow \mathcal{P}(V)$, i.e. it is a function associating a node to a certain subset of the node set. Conse-

[2]In these examples we are taking into account lemmatization: *peach* and *peaches* are treated as the same token, since they have the same root. See also Section 4.3.1.

quently, we have $N(V) \subseteq \mathcal{P}(V)$. Now, by our definition the function

$$d_J : N(V) \times N(V) \longrightarrow [0,1]$$

is a distance on the image $N(V)$ of the neighbourhood function on V. On the node set V, the reflexivity condition is not satisfied. As Lemma 4.1.8 tells us, two distinct nodes in a clique have distance 0, and Lemma 4.1.7 gives a general condition for this phenomenon. So, if we were to be completely formally correct, to speak of a distance we should consider the quotient space $V/\sim$, defining the equivalence relation induced by neighbourhoods

$$v \sim w \Leftrightarrow N(v) = N(w).$$

Otherwise, d_J on V is what is called a *pseudometric* [Arkhangel'skii et al., 2012]. From a practical point of view, however, this does not alter the way we will treat and use d_J and its weighted variant in the argumentations that follow. Therefore, we will still refer to a metric and its metric space instead of pseudometric and pseudometric space.

We present some easy results that characterize our Jaccard distance on graphs in terms of path lengths and graph diameter. The first identifies which node pairs are at a finite length.

Lemma 4.1.2. *Given a graph $G = (V,E)$, for any v and w in V, we have:*

$$d_J(v,w) < 1 \Leftrightarrow d_p(v,w) \leq 2,$$

where d_p is here and thereafter the path distance between v and w, defined in Section 2.1.2. If the nodes are in different connected components, $d_p(v,w)$ is set to be ∞.

Corollary 4.1.3. *For any $v,w \in V$, we have*

$$d_J(v,w) < 1 \Leftrightarrow v \in \bar{N}(\bar{N}(w)).$$

Corollary 4.1.4. *If G has diameter $\Delta \leq 2$, we have $d_J(v,w) < 1$ for all $v,w \in V$.*

Since we are handling only finite graphs, we can find a minimum and a maximum node degree Δ_{min} and Δ_{max}. Then we can give slightly stricter bounds for d_J on G based on node degrees.

Lemma 4.1.5. *Given a finite graph $G = (V,E)$, for all $v,w \in V$ such that $d_p(v,w) \leq 2$ we have*

$$0 \leq d_J(v,w) \leq 1 - \frac{1}{\deg v + \deg w + 1} < 1$$

Corollary 4.1.6. *Given a graph $G = (V,E)$, for all $v,w \in V$ such that $d_p(v,w) \leq 2$ we have*

$$0 \leq d_J(v,w) \leq 1 - \frac{1}{2\Delta_{max} + 1} < 1$$

We can also give a necessary and sufficient condition for two nodes to have distance 0.

Lemma 4.1.7. *Given a graph $G = (V,E)$, for all $v,w \in V$ we have $d_J(v,w) = 0$ if and only if v and w are adjacent and there exists an automorphism ϕ of G such that $\phi(v) = w$ and $\phi(w) = v$.*

Lemma 4.1.7 formalizes the notion of indistinguishability in the graph structure that is represenetd by a null distance. For reference on graph endo- and automorphisms, see [Cameron, 2004]. However, in [Erdős and Rényi, 1963] it is shown that as the size of the node set of G increases, it becomes less probable that G possesses a non-trivial automorphism. Therefore, without loss of generality, we can assume that distances on a word graph are always strictly greater than 0.

Some results can be given for particular graphs. We illustrate two extreme cases here.

Lemma 4.1.8. *If $G = (V,E)$ is complete (a clique), the distance d_J between any two nodes is 0.*

Clearly, a clique satisfies the conditions of Lemma 4.1.7, since every node is adjacent to any other and every permutation on the node set is an automorphism. Further, Lemma 4.1.8 implies that the more interconnected a subgraph of G is, the closer to 0 we expect the distance between any two of its nodes to be. This observation roughly describes the high-density regions that should correspond to the senses of a word in its local graph (as is discussed especially for the algorithms in Sections 3.1, 3.2 and 3.4).

Lemma 4.1.9. *If $G = (V,E)$ is an n-star (see Section 2.1.1), let $k \in V$ be the central node and $v_1,\ldots,v_{n-1}$ the peripheral nodes. Then $d_J(k,v_i) = \frac{n-2}{n}$ and $d_J(v_i,v_j) = \frac{2}{3}$ for $i,j = 1,\ldots,n-1$, $i \neq j$.*

This lemma describes the Jaccard distance on one of the sparsest connected subgraphs that can be found in a connected graph, a star. When $n > 6$, the central node will be more distant from the peripheral nodes than the peripheral nodes between each other, and the quantity $d_J(k,v_i)$ tends to 1 as n increases. In a small-world graph, the neighbourhood of a hub roughly assumes a star-like shape. As the degree of the hub increases, the central node will become more and more distant from its neighbours, whereas the peripheral nodes will stay at a constant distance. This can

be interpreted with the fact that as k co-occurs with every node, its occurrences are of lesser and lesser importance to determine the senses of the n_i's. We note that if we did not take open neighbourhoods in our definition of distance, the central node would have a distance of outright 1 from the peripheral nodes. In general, any node with degree 1 would have distance 1 from its single neighbour. We do not want this to happen, as the co-occurrence of two terms is after all still a sign of correlation, as weak as it may be. This point will become more relevant when we introduce the weighted Jaccard distance.

The Jaccard distance that we have discussed until now does not take into account the weighted structure of G and just uses the graph's topology. However, the weighting scheme of a word graph is fundamental in modelling the strong or weak relationships between words: a single co-occurrence may be random, but reiterated co-occurrences might be a sign of more significant connections. Hence, the basic topological structure of G should be markedly altered by edge weights, and a distance that we define on it should take into consideration the weighted landscape. To this end, we introduce a *weighted version* of the Jaccard distance on G [Cecchini et al., 2015]. The key move here is to treat node neighbourhoods not just as sets, but as multisets. A *multiset* is an unordered set where each element is allowed to appear more than once (see e.g. [Aigner, 2012]). In other terms, a multiset M is a set where each element e has an associated multiplicity m_e: M can be written as a set of couples (e,m_e). The cardinality of a multiset is then the sum of all the multiplicities of its elements. We could see any element not in M as having multiplicity 0. In this setting, ordinary sets are multisets where each element has mutliplicity either 0 or 1. We will denote the set underlying a multiset M as M^*. The intersection of two multisets A, B is again a multiset C such that

$$C = \{(v, \min(m_{v,A}, m_{v,B})) \mid v \in (A^* \cap B^*)\};$$

analogously, their union is

$$C = \{(v, \max(m_{v,A}, m_{v,B})) \mid v \in (A^* \cup B^*)\}.$$

The quantity $m_{v,A}$ is the multiplicity of element v in multiset A, which is 0 if $v \notin A^*$. Union and intersection between multisets remind of the greatest common divisor and the least common multiple of the prime factorizations of two numbers.

Now, it comes natural to model the weighted neighbourhood of a node $v \in V$ as a multiset where the multiplicity of each element is the weight on the edge connecting it to v. Precisely, the open weighted neighbourhood (see Section 2.1.5) $N^\omega(v)$ of v is defined as

$$N^\omega(v) = \{(w, p(v,w)) \mid w \in N(v)\}. \tag{4.3}$$

The function $p : E \longrightarrow \mathbb{R}^+$ defines the edge weights in G. We note that multiplicities in a multiset are positive numbers, so we are only considering weight mappings with positive values. The multiset union and intersection as previously defined give us a measure of the importance of a common neighbour of the two nodes: if it appears e.g. at least 3 times with both, it will be more significant than another one appearing 10 times with one but only once with the other. In the latter case, the discriminative power of that term will be lower, as it may be just a "companion word" of the first node. What definition (4.3) is missing is the central node v to make it a closed neighbourhood, and the problem is what multiplicity to assign it. One possible reasoning is that if v ever appears in the neighbourhood of w, its multiplicity in $\bar{N}^\omega(v) \cap \bar{N}^\omega(w)$ should be not smaller than the weight on the edge (v,w), interpretable as the importance of v from the external perspective of w. Thus $m_{v,\bar{N}^\omega(v)}$ should be higher than $p(v,w)$ to keep this value in the intersection (where we take the least multiplicities). At the same time, the multiplicity of v in the union $\bar{N}^\omega(v) \cup \bar{N}^\omega(w)$ should be a measure of the relative importance (in terms of weight) of v in the induced subgraph. We put this importance in relation to the weights on the edges incident to v, and consider their maximum to represent the influence of v in the graph. So, in the end we define the *automultiplicity*

$$m_v = \max_{w \in \bar{N}(v)} p(v,w)$$

and define the closed weighted neighbourhood of v as

$$\bar{N}^\omega(v) = N^\omega(v) \cup \{(v,m_v)\}. \tag{4.4}$$

The weighted Jaccard distance between v and w is then, analogously to (4.2),

$$d_J^\omega(v,w) = 1 - \frac{|\bar{N}^\omega(v) \cap \bar{N}^\omega(w)|}{|\bar{N}^\omega(v) \cup \bar{N}^\omega(w)|}. \tag{4.5}$$

The function d_J^ω is still a distance. This follows from the remark that we can rewrite a multiset as a set, if we substitute the element (e,m_e) with m_e different elements labelled $e_1,e_2,\ldots,e_{m_e}$ (Observation 4.1.1 still applies here).

Weighted versions of the Jaccard coefficient have already been proposed in literature. One appears in [Grefenstette, 1994] and is the most similar to our definition. The main difference, though, lies in the fact that the author considers vectors of weighted (in contrast to binary absent/present) features of different kinds (co-occurrences or syntactical dependencies) associated to a word, but no weight is associated to the considered word itself (as in the case of our automultiplicity, which is necessary in our setting). Moreover, the weights used in the computation are a mixture of a "global" and a "local" score assigned to each feature "depending

upon how many different objects with which it associates, and how often it appears" [Grefenstette, 1994, p. 48]. Another version proposed in [Strehl, 2002], called *extended Jaccard similarity*, generalizes the interpretation of the Jaccard coefficient as a vectorial operation and reformulates intersections and unions of feature sets as sum of norms and products between vectors (for a vectorial interpretation of our Jaccard distance see Section 4.1.2), by analogy with the cosine similarity, but it seems to us that it does not have an intuitive interpretation, especially on graphs. However both versions, like ours, coincide with the basic Jaccard coefficient when no weights are considered. In fact, similarly to what was discussed for the clustering coefficient (Section 2.1.6), there does not exist a unique weighted generalization of Jaccard coefficient. Our definition is at the same time very generic and straightforward and well suited to the case of a graph.

The behaviour of d_J^ω is less predictable than it was for d_J, which we can now see as the special case of d_J^ω when $p() = \mathbb{1}()$, the function that maps everything to 1. Lemma 4.1.2 and its corollaries and Lemma 4.1.7, if the automorphism preserves the weighted structure, still hold for the weighted case, but general reasonings about boundaries and particular graphs are more difficult to make. Lemma 4.1.8 does not necessarily hold, as we now show.

Example. Consider the clique K_n, $n > 2$, where the weights are 1 everywhere apart from a single edge (s,t) with weight $k \geq 2$. We have the automultiplicities $m_s = m_t = k$, and $m_v = 1$ for every other node. We can compute $d_J^\omega(s,t) = 0$ and also $d_J^\omega(v,w) = 0$ for $v,w \neq s$, $v,w \neq t$. This is immediate, as these node pairs share the exact same neighbourhoods. On the contrary, $d_J^\omega(s,v) = d_J^\omega(t,v) = \frac{2(k-1)}{n+2(k-1)}$ for $v \neq s$, $v \neq t$. This distance tends to 0^+ for a fixed k as n increases and to 1^- for a fixed n as k increases. This observation shows that distances on the most homogenous of graphs are warped by a single different edge weight: s and t are effectively set a little bit apart from the rest of the graph by their stronger connection. On one hand, a small perturbation will be overwhelmed by a massive amount of connections, but on the other hand s and t will be drawn away from the rest of the graph as their link grows stronger.

Example. If we instead suppose that the weights on all edges are $q \geq 1$, apart from the $n-1$ edges incident to a single node s, whose weights are $k \geq q$, we will have $d_J^\omega(v,w) = 0$ for $v,w \neq s$ and $d_J^\omega(s,v) = \frac{k-q}{k}$ for $v \neq s$. For a fixed q and a growing k, we observe again how s becomes more and more distant from the rest of the graph. This distance is independent from the size of the clique, but the gap is still mitigated by the strength of the other edges.

We stress that it is not always the case that $d_J^\omega(v,w) \geq d_J(v,w)$. The contrary is possible, and we will explore this fact in Section 4.2.3.

In the following discussion we will be using the weighted Jaccard distance arising from a given similarity weighting scheme on a co-occurrence graph as a way to model word contexts and to compute their semantic closeness, defined as the number of shared contexts.

4.1.2 Vectorial representation and implementation of Jaccard distances

We can define both Jaccard distances on a graph in terms of simple and weighted adjacency matrices (see Section 2.1.4 for definitions) and operations between row and column vectors.

Representations and computations are slightly more straightforward in the case of the unweighted Jaccard distance. We consider the adjacency matrix C of $G = (V,E)$, where entries can only be 1 if two nodes share an edge and 0 otherwise. Since we will use mostly undirected graphs, we assume C to be symmetric. We write the column relative to the i-th node of V (or, equivalently, row) as C_i. Each column vector represents the (unweighted) open neighbourhood of the i-th node. To represent closed node neighbourhoods, we take $\hat{C} = C + I$, where I is the identity matrix; this formally corresponds to adding self-loops to each node. Given $\hat{C}$, we can write the quantities $|\bar{N}(i) \cup \bar{N}(j)|$ and $|\bar{N}(i) \cap \bar{N}(j)|$ for each couple of nodes i, j in terms of column vectors, namely:

$$|\bar{N}(i) \cap \bar{N}(j)| = \hat{C}_i \cdot \hat{C}_j, \tag{4.6}$$

$$|\bar{N}(i) \cup \bar{N}(j)| = \sum_{k=1}^{|V|} (\hat{c}_{ki} + \hat{c}_{kj}) - \hat{C}_i \cdot \hat{C}_j = \left\|\hat{C}_i + \hat{C}_j\right\|_1 - \hat{C}_i \cdot \hat{C}_j \tag{4.7}$$

$$= \left\|\hat{C}_i\right\|_1 + \left\|\hat{C}_j\right\|_1 - \hat{C}_i \cdot \hat{C}_j, \tag{4.8}$$

where $\|\cdot\|_1$ is the 1-norm or ℓ_1-norm, and the dot is the vectorial scalar product. Then, relation (4.6) means that each entry of $\hat{C}^2 = \hat{C} \cdot \hat{C}$ represents the number of neighbours common to i and j. Moreover, by Lemma 4.1.2 each non-zero entry of $\hat{C}^2$ represents two nodes with non-infinite, less than 1 Jaccard distance.

To express the weighted Jaccard distance we need the weighted adjacency matrix. We denote it with A. As with C, we assume A to be symmetric. Again, we denote with A_i the i-th column of A, representing the open weighted neighbourhood $N^\omega(i)$ of node i: the entry a_{ij} is the weight on the edge (i,j), and if $a_{ij} = 0$, the j-th node does not appear in i's neighbourhood. However, to compute the weighted Jaccard distance we have to

consider closed neighbourhoods. To this end we define the diagonal matrix M, where $m_{ij} = \max(A_i)$ if $i = j$, and 0 otherwise. The columns of the matrix $\hat{A} = A + M$ then represent exactly the closed neighbourhoods defined in (4.4). Then, the fact that the scalar product $\hat{A}_i \cdot \hat{A}_j$ of two columns is different from 0 tells us that nodes i and j have common neighbours, but not heir exact number. The weighted Jaccard distance necessitates a slightly less compact notation than its unweighted counterpart:

$$|\bar{N}^{\omega}(i) \cap \bar{N}^{\omega}(j)| = \sum_{k=1}^{|V|} \left(\min(\hat{a}_{ki}, \hat{a}_{kj}) \right) = \left\| \min(\hat{A}_i, \hat{A}_j) \right\|_1 \tag{4.9}$$

$$|\bar{N}^{\omega}(i) \cup \bar{N}^{\omega}(j)| = \sum_{k=1}^{|V|} \left(\max(\hat{a}_{ki}, \hat{a}_{kj}) \right) = \left\| \max(\hat{A}_i, \hat{A}_j) \right\|_1, \tag{4.10}$$

where the maximum and the minimum of two vectors are taken pairwise on their elements. This definition is very similar to the one proposed in [Grefenstette, 1994].

We can now present a first algorithmic implementation of the computation of weighted and unweighted Jaccard distance on a graph. We assume that the nodes have been indicized with values from 1 to $|V|$. We start with the simple, unweighted Jaccard distance.

The time complexity of algorithm 1 depends on the matrix multiplication on line 3, whose time complexity can range from $O(|V|^3)$ of a naive implementation to the nearly quadratic $O(|V|^{2,37})$ of the fastest known algorithm (cf. [Bläser, 2013, Le Gall, 2014]). The real complexity also depends on how sparse the graph, and consequently the matrix is. The algorithm also needs to compute the 1-norm of each column vector (line 5), an operation performed with time complexity $O(|V|)$. Then, the actual computation of the distance in line 11 has to be performed at most $\binom{|V|}{2}$ times, for a complexity of $O(|V|^2)$. This gives a combined complexity lying between $O(|V|^2)$ and $O(|V|^3)$.

However, this complexity might be reduced using Lemma 4.1.2. For each node v, we have just to consider all its neighbours at a path distance of at most 2, i.e. its second degree closed neighbourhood $\bar{N}^2(v)$ (see Section 2.1.5 for notation). In the following we will call the elements of $\bar{N}^2(v)$ the 2-neighbours of v. If Δ_{max} is the maximum degree in G, each $v \in V$ will have at the very most $\deg(v)(\Delta_{max} - 1) + 1$ 2-neighbours. We subtract 1 from Δ_{max} because we do not want to count v multiple times. To write a second degree adjacency matrix C_2, where $c_{ij} \neq 0$ implies that there is a path with length no greater than 2 between nodes i and j (see Section

Algorithm 1 Unweighted Jaccard distance computation

Input Simple adjacency matrix C of an undirected graph $G = (V,E)$

Output Matrix D representing Jaccard distance

1: $D = \mathbb{1}$ ▷ Distance matrix initialized as all-ones matrix
2: $\mathcal{S} = \emptyset$ ▷ List of column 1-norms
3: $\hat{C} = (C + I)^2$ ▷ Square matrix representing closed neighbourhoods
4: **for** $i \in \{1,2,\ldots,|V|\}$ **do** ▷ Compute column 1-norms
5: $s_i = \sum_{k=1}^{|V|} c_{ki}$
6: $\mathcal{S} = \mathcal{S} \cup \{s_i\}$
7: **end for**
8: **for** $i \in \{1,2,\ldots,|V|\}$ **do**
9: **for** $j \in \{i+1,|V|\}$ **do** ▷ We exploit symmetry
10: **if** $\hat{c}_{ij} \neq 0$ **then**
11: $d_{ij} = d_{ji} = 1 - \frac{\hat{c}_{ij}}{s_i + s_j - \hat{c}_{ij}}$ ▷ Compute distance
12: **end if**
13: **end for**
14: **end for**
15: **return** D

2.1.4), we will need at most

$$(\Delta_{max} - 1) \sum_{v \in V} \big(\deg(v) + 1 \big) = (\Delta_{max} - 1)(2\,|E| + |V|)$$

look-up operations with a limited depth-first search[3], because of the hand-shaking lemma (see Section 2.1.1.1). We remark that Δ_{max} is an extremely high bound for degrees in the graph, especially for scale-free graphs (see Section 2.1.7), for which only few nodes reach very high degrees. To make the number of look-up operations more precise, we note that for each node v we are looking at at most $\sum_{w \in N(v)} \deg(w)$ nodes. Repeated over all nodes in the graph, we will have at most

$$\sum_{v \in V} \sum_{w \in N(v)} \deg(w) = \sum_{v \in V} \deg^2(v) \tag{4.11}$$

look-up operations. Lower and upper bounds for this quantity are

$$\frac{4\,|E|^2}{|V|} \leq \sum_{v \in V} \deg^2(v) \leq |E| \left(2\frac{|E|}{|V| - 1} + |V| - 2 \right)$$

[3]For an overview of basic graph algorithms see [Even, 2011].

when $|V| \geq 2$, as proved in [de Caen, 1998][4]. An optimal, albeit quite impractical estimate of sum of squared degrees can be found following the results in [Ábrego et al., 2009]. In general, for a fixed $|V|$, only the complete graph with $|E| = \binom{|V|}{2}$ achieves the maximum sum value of $|V|(|V|-1)^2$, a cubic polynomial expression. This means that the time complexity for building the second degree adjacency matrix C_2 is most of the time subcubical in the number of nodes, and, assuming that G is connected, linear in the best case of a path or cycle graph with $|V|$ nodes and respectively $|V|-1$ or $|V|$ edges.

Now, for each non-zero entry $c_{2,ij}$ of C_2 we can compute the Jaccard distance through the relations (4.6) and (4.8). Both operations use a time complexity $O(|V|)$, as already mentioned. In the end, if we perform these operations the minimum necessary number of times, that is, for the non-zero entries in the upper triangular submatrix of C_2 (we notice that C_2 is also symmetric), we will have a total complexity T bounded by

$$4|E|^2 \leq T \leq O(2\frac{|E|^2|V|}{|V|-1} + |E||V|^2 - 2|V|),$$

which brings to an upperly very inflated estimation of time complexity of

$$O(|E|^2) \leq T \leq O(2|E|^2 + |E||V|^2), \tag{4.12}$$

recalling that (4.11) was an upper bound in the first instance. In the best case, the algorithm will run in quadratic time; in the worst possible case of a complete graph, it will have complexity $O(|V|^4)$, but we notice that if this happens, we can avoid computations, since all distances are 0 by Lemma 4.1.8. Very probably the overall complexity will be considerably less than $O(|V|^3)$ if the graph is sparse enough.

This second implementation, based on a depth-first search, is perhaps better than Algorithm 1 if we know that the graph is relatively sparse, as we can expect from a small-world graph, where most nodes have only few local connections (represented by a high local clustering coefficient, as seen in Section 2.1.6), but a general low density (cf. Section 2.1.7).

[4] A similar complexity estimate can also be obtained using a limited depth-first search algorithm like that of [Korf, 1985], using as average branching degree the graph's mean degree. A better upper bound, making use of $|E|$ and $|V|$ minimum and maximum degrees of the graph, along with other two bounds using neighbour connectivity, is found in [Das, 2004], but their form does not change the conclusions of our discussion. Another good, but unwieldy upper bound is that of [Székely et al., 1992]: $\sum_{v\in V} \deg^2(v) \leq \left(\sum_{v\in V} \sqrt{\deg(v)}\right)^2$. However, it is incomparable to the one we use here.

The algorithm and the considerations for the weighted case are not so different from what we have already seen for the unweighted one. The 1-norm of a column vector takes again a time complexity of $O(|V|)$, as min and max operations (4.9) and (4.10) equally do. However, the entries of $\hat{A}^2$ are not interpretable as cardinalities of the union of two node neighbourhoods, so that we can not resort to matrix multiplication. We have to perform operations (4.9) and (4.10) for each of the $\binom{|V|}{2}$ node couples, resulting in a total time complexity of $O(|V|^3)$. Alternatively, also in this case we can use the same second degree adjacency matrix C_2 defined for the unweighted Jaccard distance and obtain the same final time complexity estimates of (4.12).

4.1.3 Distance graph and node metric space

A distance defined on a graph allows us to consider its node set as a metric space and to build its ensuing *distance graph*. If $G = (V,E)$ is a graph with a distance d on V, then by definition (V,d) is a metric space[5]. We define the global distance graph $D_{G,d}$ as the weighted complete graph on the node set V with d as its weight function p. In other words, the weights of $D_{G,d}$ are the distances between nodes of G (and we observe that the distance matrix defined in Section 2.1.4 is the weighted adjacency matrix of the distance graph). We can define a sparser, non-complete version of $D_{G,d}$ if we just take edges whose weight lies under a given threshold. In the case when we consider the weighted Jaccard distance d_J, we are not interested in distances equal to 1, as we already observed how they are like infinite distances to us. So we can define the *finite distance graph* $\hat{D}_{G,d_J} = (V,E_{d_J})$, where

$$(v,w) \in E_{d_J} \Leftrightarrow d(v,w) < 1$$

for all $v,w \in V$. This distance graph is not necessarily complete, but by definition we have $|E| \leq \left|E_{d_J}\right|$. The inequality is strict if G is not complete and has a connected component with 4 or more elements. For Lemma 4.1.2, the number of edges increases with respect to G as we create new links between a node v and all the nodes at a path distance of 2. The exact number of new edges is given, using the vectorial interpretation of Section 4.1.2, by the quantity

$$\frac{\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}\hat{c}_{ij}^2}{2} - |V|\,,$$

the number of non-zero entries in the upper triangular submatrix of $\hat{C}^2$ (which is symmetric), for which a generous upper bound was already presented in the form of (4.11).

[5] Actually a pseudometric space, recalling Observation 4.1.1

Using Lemma 4.1.2, we can claim:

Corollary 4.1.10. *Given a graph $G = (V,E)$, for any v and w in V, its finite distance graph $\hat{D}_{G,d_J^\omega}$ will be complete if and only if G has diameter $\Delta \leq 2$.*

For the sake of lighter notation, we will write D instead of $\hat{D}_{G,d}$ to refer to the finite distance graph of G when the context is clear.

As a final remark, we note that the same vectorial interpretation already seen in Section 4.1.2 for the Jaccard distances allows us to consider an embedding of (V,d) in the Euclidean space $\mathbb{R}^{|V|}$, where a node v in V is represented by the column vector of the adjacency matrix (with self-loops) corresponding to $\bar{N}(v)$. For Observation 4.1.1, we have to allow that two nodes can be represented by the same vector. On the generic space vector space $\mathbb{R}^n$ we can generalize the weighted Jaccard distance as

$$d_J^\omega(\vec{x},\vec{y}) = 1 - \frac{\sum_{i=1}^n \min(x_i,y_i)}{\sum_{i=1}^n \max(x_i,y_i)} \tag{4.13}$$

for each $\vec{x},\vec{y} \in \mathbb{R}^n \setminus \{\vec{0}\}$. The limit for $(\vec{x},\vec{y}) \longrightarrow \vec{0}$ is 1. It is interesting to notice that expression (4.13) is bounded to $[0,1]$ only in the first quadrant, where all vectors correspond to nodes of a graph with positive weights. Otherwise, d_J^ω is not bounded, and all points have distance 1 from the origin. However, we just want to point out this generalization and will not further examine the properties of this distance on a vector space, abstracted from the original construction with multisets..

4.1.4 Gangplanks

Suppose that we have a co-occurrence word graph $G = (V,E)$ and that we have computed its corresponding finite distance graph $D := \hat{D}_{G,d}$, based on a given distance d on V. The graph D will be much denser than G, since we are now considering second-order relationships between nodes in addition to those arising from first-order ones like co-occurrences, as seen in Section 4.1.3. In Section 4.1.1 we show how d_J^ω effectively reshapes the landscape of the word graph to account for the greater or lesser strengths of correlations. Now, starting from this observation we want to define a way to let the borders between different denser regions emerge naturally in the general case of an undirected, weighted graph G.

We introduce the notion of *gangplank edge*. We can imagine, as can be the case in Venice, a gangplank connecting the opposite sides of a big puddle or even of a canal separating two small islands; the same way we want a gangplank edge (in short: gangplank) to be a weak connection between two dense and individually strongly intraconnected subgraphs. We

chose this name because *bridge* is already used in graph theory to denote another situation, namely an edge whose removal would disconnect the graph (see Section 2.1.3); however, removing a gangplank does not necessarily creates two distinct connected components. Its nature is purely local and pertinent to the neighbourhoods of the two involved nodes.

Definition 4.1.11. (Gangplank edge) *Given a weighted graph $G = (V,E)$ with positive weight function $p : E \longrightarrow \mathbb{R}^+$ and an edge $e = (v,w) \in E$, we say that e is a gangplank if it satisfies*

$$p(v,w) > \max\left(\frac{1}{\deg v}\sum_{z\in N(v)} p(v,z), \frac{1}{\deg w}\sum_{z\in N(w)} p(w,z)\right). \tag{4.14}$$

In other words, we say that an edge is a gangplank if its weight is strictly greater than the mean weight of both its two ends. If we picture $e = (v,w)$ as connecting the two "halves" of $N(v) \cup N(w)$, we want to consider e a gangplank if it is keeping v and w apart rather than bringing them together. We remark that this definition has a sensible interpretation only in the case of positive weights. Moreover, it is defined in the case of a graph whose edge weights represent distances (the smaller, the better). We can always obtain such a graph e.g. building the distance graph of any graph, as seen in Section 4.1.3. Of course, we can also reverse the definition when weights are meant to be similarities (the greater, the better).

Definition 4.1.12. (Gangplank edge for similarities) *Given a weighted graph $G = (V,E)$ with positive weight function $p : E \longrightarrow \mathbb{R}^+$ and an edge $e = (v,w) \in E$, we say that e is a gangplank if it satisfies*

$$p(v,w) < \min\left(\frac{1}{\deg v}\sum_{z\in N(v)} p(v,z), \frac{1}{\deg w}\sum_{z\in N(w)} p(w,z)\right). \tag{4.15}$$

Gangplanks inform us of stray or random connections that arise from the data, but that when compared to the others in the context of a word are of much lesser significance. The intuition is that they can be used to outline the denser regions of the word graph that will represent the sense clusters of the target word to be disambiguated. The clustering algorithm that we will present in Section 4.2.1 will exploit this idea. Here we present an algorithm that labels the edges of a graph as gangplanks or not, according to a specific distance on V. The labelling is defined by a binary matrix Γ with an entry $\gamma_{vw} = 1$ if edge (v,w) is a gangplank, and $\gamma_{vw} = 0$ otherwise.

The computation of the weight mean for a single node v has a time complexity of $O\big(\deg(v)\big)$. Repeating this for each node has a total cost of

$$O\big(\sum_{v\in V}\deg(v)\big) = O(|E|),$$

Algorithm 2 Gangplank edges computation

Input Undirected graph $G = (V,E)$ with weight function $p : E \longrightarrow \mathbb{R}^+$

Output Binary matrix Γ representing gangplanks

1: $\Gamma = O$ ▷ Gangplank matrix initialized as zero matrix
2: $\mathcal{S} = \emptyset$ ▷ Mean weight for each node
3: **for** $v \in V$ **do**
4: $\quad s_v = \frac{1}{\deg(v)} \sum_{w \in N(v)} p(v,w)$ ▷ Mean weight for a node
5: $\quad \mathcal{S} = \mathcal{S} \cup \{s_v\}$
6: **end for**
7: **for** $(v,w) \in E$ **do**
8: $\quad$ **if** $p(v,w) > \max(s_v,s_w)$ **then** ▷ Gangplank condition
9: $\quad\quad \gamma_{vw} = 1$
10: $\quad$ **end if**
11: **end for**
12: **return** Γ

for the handshaking lemma (see Section 2.1.1.1). In addition, we have to test each edge by checking an inequality: in total, we have $O(|E|)$ operations, and this is also the overall time complexity.

4.2 The clustering algorithms

We will now present our three graph-based clustering algorithms, making use of the concepts developed in the previous sections. Here we define them in their general forms, leaving the details of their implementations for WSI tasks to Section 4.3.

4.2.1 Gangplank clustering algorithm

In this section we will describe the core of the gangplank clustering algorithm in its most general from. It was first presented in [Cecchini and Fersini, 2015]. Its implementation for WSI tasks, which makes full use of the instruments developed in previous sections, will be detailed in Section 4.3.2.

Suppose that we have a graph $G = (V,E)$ where each edge has been labelled as gangplank or non-gangplank. Formally, we achieve this by saying that there is a function $g : V \times V \longrightarrow \{0,1\}$ defined on the Cartesian product of the node set that yields 1 if the edge is a gangplank and 0 otherwise (this includes a node pair which is not connected by an edge).

In this setting it is not important what form the starting graph has or what similarity score and distance were used to finally apply Definition 4.1.4 of a gangplank edge: we will just focus on the clustering step. In this sense, g could actually be any binary edge labelling function, to which we conventionally refer as *gangplank labelling function*.

Given g, we define the *gangplank degree* in G of a node v as

$$\gamma_G(v) = \frac{\sum_{w \in N_G(v)} g(v,w)}{\deg_G v}. \tag{4.16}$$

The subscript G means that the interested object is computed only with regard to graph G. This allows us e.g. to restrict our attention to a subgraph $H = (V',E') \subseteq G$: the quantity $\gamma_H(v)$ for $v \in V'$ considers only the neighbourhood $N_H(v) = N_G(v) \cap V'$, and similarly for $\deg_H$. When it is clear to which graph we refer, we will just write $\gamma(v)$ for ease of notation. The ratio of gangplank to non-gangplank edges exiting a node is a measure of its isolatedness: a node with a great γ does not have many significant connections to other nodes and is rather marginal in the word graph. On the contrary, a node with low γ is likely to lie in the midst of one of the connected regions that we want to identify. The first, stricter objective of our clustering algorithm is to find the coarsest partition of G (that is, of its node set) that consists only of **connected subgraphs with no gangplanks**. In such a partition, gangplanks only appear as interconnecting edges between clusters, *bridging* (or better *gangplanking*) them. Of course, it is unfeasible to scan through all $2^{|V|}$ possible partitions of V to find the one that interests us. We also notice that a partition with a minimal number of clusters might not be unique. Instead, the algorithm is going to expand and define the clusters from particular seed nodes following some heuristical intuitions, and the end result will be further loosened by requiring that the partition consists of subgraphs with few gangplank interconnections and a greater number of gangplank intraconnections.

The first assumption is that, most probably, the node at the heart of a cluster will have the cluster's lowest value of γ. Therefore, the algorithm starts by selecting the node w of G with the least γ as the seed of the first cluster; if there is a tie, the seed is randomly chosen among the candidates. Then we introduce a cycle of expansion and reduction steps that will iteratively define a subgraph K of G. Initially, we will set the cluster $K = \{w\}$. We define the set of already clustered nodes Q, for which it will initially hold $Q = \emptyset$, and denote our partition as $\mathcal{C}$, which will equally start as the empty set.

Expansion: We consider the subgraph induced by K and its neighbours, ignoring any node that we may have previously included and discarded in this expansion-reduction cycle (see next step), or that has already been clustered by the algorithm. If $S \subset V$ is the subset of discarded nodes, this means that we are considering the subgraph $G\langle K' \rangle$ induced by[6]

$$K' = N\bar{(}K) \backslash (S \cup Q).$$

Thereafter, we pass to the reduction step. If it happens that $K' = K$, i.e. there are no unseen nodes by which we can expand the subgraph, we break the cycle and consider K as a cluster of our partition $\mathcal{C}$. The nodes in K will be added to Q.

Reduction: During the reduction step, all nodes that bring gangplank edges to $G\langle K' \rangle$ will be progressively discarded, until no more gangplanks are left in the subgraph. We compute $\gamma_{G\langle K' \rangle}$ for the nodes of our subgraph and start by taking out the node with greater gangplank ratio among them. We denote it by g_1. Then, we update the γ scores for the remaining nodes in $G\langle K' \rangle \backslash \{g_1\}$ and if there are still gangplanks left, i.e. equivalently a node g_2 has a γ score greater than 0, we remove it too from our subgraph. This process is repeated until there are no more gangplank edges left in $G\langle K' \rangle \backslash \{g_1, g_2, \ldots\}$, i.e. equivalently $\gamma = 0$ for all nodes left in $G\langle K' \rangle$. Our only restraint is that the seed node will never be discarded. Then we add the set $\{g_1, g_2, \ldots, g_\sigma\}$ of σ discarded nodes to the set S. Prior to the first iteration of the reduction step in this cycle, S will be empty. Finally, we go back to another expansion step.

Ties can ensue when choosing which node to remove. If two nodes have the same γ score, we regard, in order of importance: the least degree in $G\langle K' \rangle$, the absence of a connection to w, the greatest weight on such connections, the greater score γ_G. If the tie still persists, all offending nodes will be removed, as they are undistinguishable in this sense. Of course, these are not the only criteria that might be used. Since we want $G\langle K' \rangle$ to stay connected, if by removing the nodes we obtain more than one connected component, we will retain only the component of w and discard all other nodes.

Each cycle defines a cluster. At the end of each cycle, the algorithm selects a yet unclustered node in $V \backslash Q$ that has the least γ_G, and starts repeating the expansion and reduction steps anew. The set of discarded nodes S is specific to each run of the cycle: if a node had been discarded for a previous cluster, it can still be considered again for a later one. We

[6] We notice that, for the definition given in Section 2.1.5, if K has more than one element $\bar{N}(K)$ or even $N(K)$ will always contain K.

expect to identify the bigger clusters first and to detect singletons last. We notice that the definition of the algorithm is unaffected by the connectedness of G: given a seed node, the cluster relative to it will naturally be confined to its connected component. In the end, the partition $\mathcal{C}$ will have at least as many clusters as there are connected components in G.

The expansion of a subgraph of G might be slightly reminding of percolation theory [Bollobás and Riordan, 2006], where we treat gangplanks as closed edges and the subgraph grows following only open edges. The ratio $\frac{\sum_{e \in E} g(e)}{|E|}$ of gangplank edges to the totality of edges in G could be used in a maximum-likelihood way as the probability p of an edge being closed, and there might be relations between the behaviour of the algorithm and its percolation properties that deserve investigation. However, we note that just considering non-gangplank nodes for clustering gives totally different results, as will be discussed below.

The requisite that each cluster be a subgraph with no gangplanks is quite strict, and may lead to a very fragmented clustering full of singletons of the kind $K = \{z\}$. Unless z appears as the lone member of its connected component, it is always the case of a node that just happens to not have non-gangplank connections to any of the found clusters, so that the algorithm also ignores any other meaningful link it might have. This side effect further depends on the choice of the seed nodes, which is done through the heuristic of γ's computation, and is also determined by the hard-clustering nature of the gangplank algorithm. A soft-clustering version is thinkable and could solve this problem, and will be described later as a variant. With these considerations in mind, we can regard the clusters of $\mathcal{C}$ as the homogeneous nuclei of a coarser (see Section 2.1.9.1) clustering $\mathcal{C}' \supseteq \mathcal{C}$ around which stray nodes are recompacted, this time allowing gangplank edges to be present in the clusters. We define thus a recompacting step. The first singleton z to be reassigned to a cluster will be the one with lower γ_G, that is, the least isolated of the singletons and the one which we can most knowledgeably reassign.

Recompacting: For each cluster $C \in \mathcal{C}$ such that $|C| > 1$, we compute the ratio κ_C of gangplank edges among all the edges connecting z to C. Formally:

$$\kappa_C = \frac{\sum_{v \in C} g(v,z)}{\sum_{v \in C} a_{vz}}.$$

The quantity a_{vz} is the entry in the adjacency matrix A of G (see Section 2.1.4): it is 1 if there is an edge between v and z and 0 otherwise. The greater κ_C for a given cluster, the more easily z can be embedded in C. So, we define $C_{max} = \arg\max_{C \in \mathcal{C}} \kappa_C$ and reassign z to C_{max}. Formally, we are considering the new clustering $\hat{\mathcal{C}} = (\mathcal{C} \setminus \{\{z\}, C_{max}\}) \cup \{C_{max} \cup \{z\}\}$.

Algorithm 3 Recompacting algorithm

Input Undirected graph $G = (V,E)$, function $g : V \times V \longrightarrow \{0,1\}$, clustering $\mathcal{C}$ of V, $\theta \in \mathbb{N}^+$

Output Coarser clustering $\hat{\mathcal{C}} \supseteq \mathcal{C}$

1: **for** $K \in \mathcal{C}$ **do**
2: **if** $|K| \leq \theta$ **then**
3: $\mathcal{C} = \mathcal{C} \setminus \{K\}$
4: **for** $v \in K$ **do**
5: $C_{max,v} = \arg\max_{C \in \mathcal{C}} \frac{\sum_{w \in C} g(v,w)}{\sum_{w \in C} a_{vw}}$
6: **end for**
7: **for** $v \in K$ **do**
8: $\mathcal{C} = \mathcal{C} \setminus \{C_{max,v}\}$ ▷ Clusters to be updated are removed
9: $C_{max,v} = C_{max,v} \cup \{v\}$ ▷ The node is reassigned
10: **end for**
11: **for** $v \in K$ **do**
12: $\mathcal{C} = \mathcal{C} \cup \{C_{max,v}\}$ ▷ Clustering is updated
13: **end for**
14: **end if**
15: **end for**
16: **return** $\hat{\mathcal{C}} = \mathcal{C}$ ▷ The clustering has now been updated

After a singleton has been reassigned, we will choose the one with second lowest γ_G as the next one, and so on, until no more unreassigned nodes are left. This way, each subsequent reassignment will take into account the new composition of the clusters. The final output will be a clustering $\mathcal{C}'$ of V where every cluster has at least 2 elements and the amount of internal gangplanks is minimized. Possible ties are broken considering, in order of importance: the absolute number of non-gangplank connections, the size of the cluster, and finally by random choice.

This step in the gangplank algorithm could see the introduction of the only parameter envisaged for it, and precisely of a threshold θ that determines the size of the clusters that have to be broken up and whose nodes will be reassigned to bigger clusters.

Algorithm 4 Basic gangplank clustering algorithm

Input Undirected graph $G = (V,E)$, function $g : V \times V \longrightarrow \{0,1\}$ for the computation of γ

Output Clustering $\mathcal{C}$ of V

$\mathcal{C} = \emptyset$
$Q = \emptyset$ ▷ The set of already clustered nodes
while $Q \subsetneq V$ **do**
  $w = \arg\min_{v \in \mathcal{V}} \gamma_G(v)$
  $K = \{w\}$ ▷ The new, seeded cluster
  $S = \emptyset$ ▷ Set of discarded nodes
  $\mathcal{N} = N(K) \backslash (Q \cup S \cup K)$ ▷ Viable neighbours of K
  while $\mathcal{N} \neq \emptyset$ **do** ▷ Checking if new nodes can be added
    $K = K \cup \mathcal{N}$
    while $\exists v \in K \backslash \{w\}$ s.t. $\gamma_K(v) > 0$ **do** ▷ Looking for gangplanks
      $u = \arg\max_{v \in K \backslash \{w\}} \gamma_K(v)$ ▷ Possible ties need to be broken
      $K = K \backslash \{u\}$
      $S = S \cup \{u\}$ ▷ Update discarded nodes
    end while
    $\mathcal{N} = N(K) \backslash (Q \cup S \cup K)$ ▷ Look for new viable neighbours
  end while
  $\mathcal{C} = \mathcal{C} \cup \{K\}$ ▷ Add the cluster to the clustering
  $Q = Q \cup K$ ▷ Remember clustered nodes
end while
return $\mathcal{K}$

The basic gangplank algorithm without recompacting step is schematized in Algorithm 4, and the recompacting step in Algorithm 3. The threshold θ is 1 in the basic version, as discussed for the recompacting step. The soft-clustering variant is not shown.

The time complexity T of Algorithm 4 is not immediately assessable. In the worst case, we claim that T is roughly approximable by a complexity between

$$O(|E| + |V| \log |V|) \leq T \leq O\big(|V| \, (|E| + |V| \log |V|)\big). \tag{4.17}$$

This estimate is probably very generous and far from the optimal one, which actually depends on the density and the diameter of G.

The key consideration is that, like for Algorithm 2, the computation of γ_G for each node in G is $O(|E|)$, following from the handshaking lemma (see Section 2.1.1.1). Their sorting (to choose the lowest score) has a complexity of $O(|V|\log(|V|))$. Therefore, in each subgraph $g = (v,e) \subseteq G$ we need a number of operations in the order of $O(|e| + |v|\log|v|)$ to compute and sort the local γ_g to choose which node to remove. Then, updating the remaining γ's has a cost equivalent to the degree in g of the removed node. If we were to remove every node (which would terminate the expansion and reduction step cycle for that cluster), we would have again to perform a total of $O(|e|)$ operations. Now, of course $O(|E| + |V|\log|V|)$ is an upper bound of the required complexity for each cluster of the final partition $\mathcal{C}$. We note that, having a seed node v, we can perform the expansion step at most $\epsilon(v)$ times, where $\epsilon()$ is the eccentricity of a node (see Section 2.1.2). Since the eccentricity is maximized by the graph diameter Δ, we know that the expansion-reduction cycle can be performed at most Δ times for each node. The expected number of nodes that we add to g at each expansion step depends on the density δ of the graph, and is $\delta(|V| - |v|)$. So each cluster will require $O\big(\Delta(|E| + |V|\log|V| + \delta|V|)\big)$ steps. The final number of found clusters, and subsequently of performed expansion-reduction cycles, is difficult to estimate, but it is surely much less than $|V|$. All these observations lead us to the already mentioned presumed overall time complexity (4.17).

The recompacting step requires a number of operations equal to the sum of the degrees of the nodes to be reassigned, and then the sorting of the gangplank ratio for each cluster. These numbers will be small and the total complexity will be much less than $O(|E|)$, if not negligible at all.

We end the description of the algorithm with three considerations and a soft-clustering variant.

The first consideration is that the initial "pure" clustering $\mathcal{C}$ obtained performing the expansion and reduction steps is not necessarily the same one that would be found by directly removing all gangplank edges from G, and ends up being actually quite different. Consider the situation depicted in Figure 4.1. The gangplank algorithm that we described would give us the clustering $\big\{\{A,B,C\},\{D,E,F\}\big\}$ in two expansion-reduction cycles. In the first cycle, if E is selected as the first seed node, in the first expansion step the only gangplank edge in $G\langle\{C,D,E,F\}\rangle$ will originate from C. Thus C will be removed. Then, in the second expansion step A will be added but immediately removed, and we will obtain the first cluster $\{D,E,F\}$, followed by $\{A,B,C\}$. The same clusters result when choosing B as the first seed node. If, however, we were to just cancel the non-gangplank edges, we would get a unique cluster $\{A,B,C,D,E,F\}$, because we had no clear

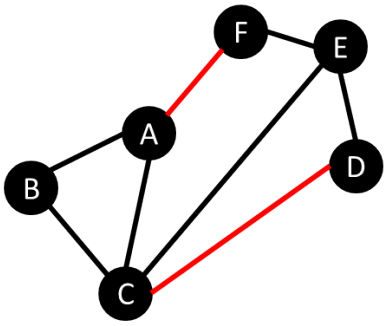


Figure 4.1: A toy graph G. Red edges are gangplanks.

means for ignoring the stray edge (C,E). In general, clusters consisting of connected components with respect to gangplank edges will be sensibly bigger and are likely to contain many gangplank edges when seen as subgraphs of G. Going back to the digression about percolation theory, it is plausible to think, and would be interesting to prove, that the gangplank ratio of G is always above its percolation threshold, and thus that removing all gangplank edges still would not disconnect G.

The second consideration is that, like the previously discussed HyperLex (Section 3.4) and MaxMax (3.5) and unlike MCL (3.1) or Chinese Whispers (3.3), the gangplank algorithm is deterministic. Its outcome is completely determined by the gangplank function g, which is itself determined by the underlying weighting scheme. The only undetermined choices that might occur depend on how the graph G or the clustering $\mathcal{C}$ is indexed by the system. Otherwise, the decisions to break the ties, presented earlier as random, actually choose the node or the cluster with the lowest index. If indexings do not change between different runs, the result will always be the same.

The third consideration, somewhat tied to the deterministic nature of the algorithm, is that the clustering is not so strongly dependent on the used weighting scheme. In fact, the gangplank function measures the relative strength and weakness between edges, so that their exact values are not needed in further computations.

Finally, let us briefly detail a soft-clustering variant of the algorithm. To allow overlapping clusters, we can ignore the set Q of already clustered nodes during the expansion step. This way it would be possible for K to include nodes that already belong to a previously found cluster. This leaves the problem of how each seed node should be chosen: if we also

relax the constraint of choosing a not yet clustered node as a seed, we risk discovering many times the same cluster, or finding nearly identical clusters. To prevent this, a suitable candidate could be taken, as for the basic version, from still unclustered nodes, or at least from nodes at distance 2 or more from an already expanded seed.

In the present work, we do not evaluate this soft-clustering variant of the gangplank algorithm, which remains as a theoretical possibility to be expanded and investigated in the future. We just point out that evaluation for soft-clusterings might require different measures than for hard-clusterings; of the three scores that we use in our pseudoword evaluation framework (see Section 5.3), only BCubed is suited to this task.

4.2.2 Aggregative clustering algorithm

Again, in this section we will describe the general functioning of our aggregative[7] clustering approach, as first presented in [Cecchini et al., 2015]. Its implementation for WSI tasks, as that of the other here presented algorithms, will be detailed in later sections.

Given a finite metric space (V,d) of any kind, where d is its metric function, we can apply any clustering algorithm that relies on a distance, such as the k-means, k-medoids or the k-nearest neighbours algorithms. Our aggregative approach is inspired by them and it is in fact the less graph-based of our proposed methods, due to its vectorial interpretation (see Section 4.1.2). We could describe our aggregative algorithm as a k-medoids algorithm (based on the original work by [Lloyd, 1982]; see also [Kaufman and Rousseeuw, 1987]) without a fixed initial k. In fact, our parameter is not the number of clusters, but the radius σ, which is used in the first iteration to initialize the starting medoids, letting them arise from the distribution and relative density of the elements in V. The radius ultimately determines the granularity of the final clustering. A bounded distance like d_J of Section 4.1.1 is of help here, since it allows us to better grasp the consequences of choosing a given value for σ. With regard to subsequent iterations, the algorithm behaves like a version of the usual k-medoids algorithm and consists of two repeated steps: the reassignment of nodes to clusters and the recomputation of cluster medoids.

[7] Here and thereafter we will use the term *aggregative* instead of *agglomerative*, more commonly seen in literature. In fact, the latter implies a hierarchical clustering and the recursive merging of smaller clusters, like in [Brown et al., 1992]. Our approach instead lacks the notion of hierarchy and *gathers*, *aggregates* points around more central elements.

A *medoid* of a discrete subset Q of V is defined as an element of Q with the least mean distance from all other elements of Q, or equivalently, whose sum of distances to the other elements of Q is minimal. Thus, a medoid can be defined as the discrete approximation of a mean value: for example, in a Euclidean space $\mathbb{R}^N$ with the norm as its distance, the medoid of a discrete subset $Q \subset \mathbb{R}^N$ is the element closest to the point of $\mathbb{R}^N$ representing the mean element, which may not lie in Q. With respect to k-means algorithms, working with medoids allows us to generalize this kind of partitioning algorithms to any kind of metric space and distance, keeping the focus on the elements we want to cluster and the distance that binds them, rather than on the Euclidean vectorial space where we might embed those elements. Besides this, a medoid readily provides us with a possible most significative representant of a cluster.

Formally, we define the medoid μ of set Q as

$$\mu = \arg\min_{q \in Q} \sum_{p \in Q} d(q,p). \tag{4.18}$$

Similarly to the gangplank clustering algorithm 4.2.1, we choose the first seed element w, the one that starts the algorithm, by some appropriate criterion, or even randomly. The first cluster K is initialized as $K_1 = \{w\}$, and w represents $K_1 = \{w\}$ as its trivial medoid. After w, every other element of V is visited. For the first visited element v, there are two possibilities:

- we have $d(w,v) < \sigma$, i.e. the two elements are close enough to be part of the same cluster. We assign v to K_1, so that now $K_1 = \{w,v\}$.
- Otherwise, if $d(w,v) \geq \sigma$, the node v is taken as the seed of the new cluster K_2, initialized as $K_2 = \{v\}$, and is defined as its current medoid.

Subsequently, each further element will be compared to the two seeds, and σ will determine if it will become part of an existing cluster or form a new one. The generic (re)assignment step for element v is as follows:

Reassignment: Let $\mathcal{C} = \{K_1, \ldots, K_n\}$ be the clusters found up to this moment. Let μ_i, $i = 1, \ldots, n$ be their respective medoids. We look for the element

$$\mu_{min} = \arg\min_{\mu_i} d(v,\mu_i),$$

that is, the medoid closest to v according to distance d. We recognize two possible cases:

- If we have $d(v,\mu_{min}) < \sigma$, we will assign v to the corresponding cluster K_{min}.

- Else, if $d(v,\mu_{min}) \geq \sigma$, it means that v is not close enough to any of the medoids, so that we create a new cluster $K_{n+1} = \{v\}$, defining $v = \mu_{n+1}$ as its current medoid.

After every element in V has been seen for the first time, we will have obtained a primitive clustering $\mathcal{C}$. The elements that were defined as their respective medoids during the reassignment step will probably not yet satisfy condition (4.18). For this reason, we recompute the medoids; in a sense, we are recentering the clusters, finding for each its own better representant.

Medoid recomputation: For each cluster in $\mathcal{C} = \{K_1,\ldots,K_n\}$, we redefine its medoid as

$$\mu_i = \arg\min_{v \in K_i} \sum_{w \in K_i} d(v,w),$$

for $i = 1,\ldots,n$.

After this first recomputation, the k-medoids part ofthe algorithm starts. We will again visit every element in V in random order and repeat the reassignment step. While many elements will still be close to their own medoids, and while many medoids might actually have not changed at all, others that found themselves at the border of a cluster might be shifted. In any case, the number of clusters will always stay the same.

The algorithm proceeds iterating a reassignment step and a medoid recomputation step. The goal is to reach a stable configuration where medoids, and consequently their corresponding clusters, do not need to be readjusted anymore. The algorithm stops when the set of lastly computed medoids is equal to the set of medoids in the previous iteration. It is possible, as was similarly the case for Chinese Whispers (Section 3.3), that the limit of the process is not a single configuration, but two or more cyclically repeated configurations. This is even more probable in our case, as we are using medoids instead of means: smaller changes in the reassignments might also change the best option among similar medoid candidates, even if they are equivalent for the composition of the clusters. Therefore, we elect to alternatively stop the algorithm after a fixed given number of iterations.

It is also possible to add a recompacting step, in the spirit of the one described by Algorithm 3 for the gangplank clustering algorithm. In this case, the role of the relative number of gangplank connections will be played by distance: each node is reassigned to the cluster represented by the medoid closest to it.

Different runs of the aggregative clustering algorithm that we just described will yield possibly different outputs if the first seed element is chosen randomly. If, on the other hand, there is a definite choice criterion, differences in the outputs of different runs will depend on the order in which elements of V are visited. Apart from these two remarks, the algorithm is deterministic: given the same seed element and the same vising order, it will always behave the same way. Since we are assuming that V is a discrete set, a sensible choice for σ lies in $[d_{min},d_{max}]$, where d_{min} and d_{max} are respectively the minimum and maximum distance between elements of V. We remark that the lower bound is more relevant than the upper bound, as the latter is influenced by outliers and can be disproportionately greater than the mean values assumed by d on V, whereas d_{min} is more telling of the distribution of the graph. Here distribution is meant in an abstract way: we picture a dense region as a subset of V in which the mean distance is relatively low.

Also, we remark that the number of possible clusterings is limited by the finite sequence

$$\Delta = \{d_1,d_2,\ldots,d_i\},$$

$1 \leq i \leq \binom{|V|}{2}$, $d_k \leq d_{k+1}$, of all the values that d assumes on V. The elements of Δ give a set of thresholds around which a change of σ also generates a new clustering. If (V,d) is obtained from a graph $G = (V,E)$, as discussed in Section 4.1.3, the cardinality of Δ and the difference $d_{max} - d_{min}$ can be seen as indicators of the homogeneity of G's degree distribution. The choice of radius σ can be made dependent on this distribution: we can choose to use a given percentile of the set Δ. For example, the 75th percentile would produce a certain coarseness and the resulting clustering would be coarser than using the 20th percentile. This way, if the aggregative clustering algorithm is performed on different graphs, fixing a percentile of Δ will also fix a given granularity, independently from the graph.

We show the functioning of our proposed aggregative algorithm in Algorithm 5.

To estimate the time complexity of Algorithm 5, we assume that distances have already been computed for each couple of elements in V. Then, suppose that we have found m medoids; this number will never change. In each reassignment step, every node will be compared to each of the m medoids, for a complexity of $O(m\,|V|)$. The medoid recomputation step is more expensive: in each cluster K_i we have to perform $|K_i|\,(|K_i|-1)$ operations, i.e. adding $|K_i|-1$ elements for each node in K_i. Since our clustering is a partition of V, we have the relation $\sum_{i=1}^{m} |K_i| = |V|$. Therefore, since we know that $\sum_{i=1}^{m} |K_i|^2 \ll |V|^2$, at each iteration we will have a maximal overall time complexity of $O(|V|^2)$.

The total complexity also depends on radius σ: the smaller, the more fragmented the clustering will be, meaning that the cost of the reassignment step increases and that of medoid recomputation decreases, with some equilibrium point in between. In the end, Algorithm 5's complexity will oscillate between linear and quadratic in the cardinality of V.

Algorithm 5 Aggregative clustering algorithm

Input Metric space (V,d), radius σ, maximum number of iterations I_{max}

Output Clustering $\mathcal{C}$ of V

$\mathcal{C} = \emptyset$
$\mathcal{M} = \mathcal{M}' = \{\}$ ▷ The sets of new and old medoids
$\mathcal{M} = \mathcal{M} \cup \{w\}$ ▷ Choice of first seed medoid
$K_1 = \{w\}$ ▷ Initialize first cluster
$count = 0$ ▷ Initialize iterations' counter
while $\mathcal{M}' \neq \mathcal{M}$ or $count \leq I_{max}$ **do**
    for $v \in V$ **do** ▷ Assignment step
        $m = \arg\min_{i=1,\dots,|\mathcal{M}|} d(v,\mu_i)$
        if $d(v,\mu_m) < \sigma$ **then**
            $K_m = K_m \cup \{v\}$ ▷ Assign node to cluster
        else
            $K_{|\mathcal{M}|+1} = \{v\}$ ▷ Initialize new cluster
            $\mathcal{M} = \mathcal{M} \cup \{v\}$ ▷ Add new medoid
        end if
    end for
    $\mathcal{C} = \bigcup_{i=1,\dots,|\mathcal{M}|}\{K_i\}$ ▷ The found clustering
    $\mathcal{M}' = \mathcal{M}$ ▷ Save current medoids before recomputation
    for $i = 1, \dots, |\mathcal{M}|$ **do** ▷ Medoid recomputation step
        $\mu_i = \arg\min_{v \in K_i} \sum_{w \in K_i} d(v,w)$
    end for
    $count = count + 1$
end while
return $\mathcal{C}$

4.2.3 Curvature-based algorithm

In this section we will describe our third and last proposed algorithm, inspired by the concept of *synthetic curvature*. The approach of [Dorow, 2006], already discussed in Section 3.2.1, also conceptually revolves around curvature. However, we can point out some important differences. First, in [Dorow, 2006] curvature is a quantity associated to the nodes of a graph,

whereas we associate it to edges; second, there, semantic clustering is achieved by pruning nodes under a given curvature threshold, whereas we aim to decompose the graph based on regions with similar curvature; third, in [Dorow, 2006] it is defined in terms of local clustering coefficient (see Section 2.1.6), whereas we tie it to the notion of distance. These differences will become clear in the following. We will first present an overview of the geometrical concept of curvature to give a better insight into the motivations of our curvature-based clustering algorithm, and then go into the details of the algorithm. The implementation of this curvature-based algorithm will be discussed in Section 4.3.4.

Curvature is a geometric notion which measures to what extent a surface differs from the *flat* Euclidean plane, i.e. a Euclidean space with only two dimensions. The fact that not all surfaces possess the same geometric properties was probably first indirectly discovered around the XVIth century by mathematicians trying to prove the famous Euclid's fifth postulate, the parallel postulate [Euclid, 1956]. This axiom of Euclidean geometry states:

> If two right lines AB, CD meet a third line AC, so as to make the sum of the two interior angles $B\hat{A}C$, $A\hat{C}D$ on the same side less than two right angles, these lines being produced shall meet at some finite distance.

Equivalently, the same axiom can be expressed as:

> Given a line AB and a point C, there exists only one other line CD passing through C and not intersecting AB, namely the one such that $B\hat{A}C$ and $A\hat{C}D$ are two right angles.

We call CD the line parallel to AB through C.

This assertion is however not true on every surface. There exist geometrical spaces where the Euclidean parallel postulate does not hold, called *non-Euclidean geometries*. The unintentional discoverer of such alternative axiomatic systems is thought to be Saccheri[8], an Italian mathematician who, attempting to prove the parallel postulate, could not actually find valid arguments against two different ways of negating it (although he believed to have found them). Only much later it was discovered that those two negations lead to different geometries: the one of spaces locally behaving like a sphere, where no parallel lines exist, and the one of hyperbolic surfaces[9], where a line can have infinite other parallel lines. Differential geometry provides the link between such intrinsic properties of a

[8] We point to [Boyer and Merzbach, 1989] for an excellent overview of this period in the history of Mathematics.

[9] The form of a saddle is a rough approximation of a hyperbolic surface.

surface and their definitions in form of an intrinsic, computable quantity called Gaussian curvature [Griffiths and Harris, 1978]. The sign, positive, negative or null, of curvature, invariantly characterizes the three mentioned surface families: sphere-like surfaces have positive curvature, hyperbolic surfaces have negative curvature and the familiar Euclidean plane has zero curvature.

Another way to express curvature is to consider two points X and Y on a surface embedded in a vector space that is also a metric space with distance d. Then, we choose a direction other than XY (the one of the line[10] passing through x and y) and represent it with vector $\vec{v}$. We start tracing two half-lines in that direction from both X and Y. We distinguish three possible cases:

1. the distance between the two lines never changes, so they are effectively parallel and we are on a Euclidean plane with no curvature;
2. the distance between the two lines lessens, so they converge and eventually intersect. We are on a spherical surface with positive curvature;
3. the distance between the two lines increases and they diverge. We are on a hyperbolical surface with negative curvature.

If we denote with $d_0 = d(X,Y)$ the initial distance and with

$$d_t = d(X + \vec{v}t, Y + \vec{v}t), \quad t \in \mathbb{R}^+,$$

the distance between the two corresponding points of the half-lines, defined through parameter t, we can define the quantity

$$\Delta_t(X,Y) = d_0 - d_t. \tag{4.19}$$

We discern again three possible cases:

i. $\Delta_t = 0$ for all values of t. We are in case 1.

ii. There exists a t such that $\Delta_t > 0$: the two lines intersect and we have again case 2.

iii. There exists a t such that $\Delta_t < 0$: the two lines diverge and this condition falls in with case 3.

[10] A line through X and Y can be defined as all the points of the form $X(t-1) + Yt$, where $t \in \mathbb{R}$.

So we can put the fact of finding zero, positive or negative curvature into relation to the difference between distances in the expected, standard case (plane, zero curvature) and in the case of the actual surface we are considering. Following this reasoning allows us to give a *synthetic* definition of curvature; that is, we have extrapolated one aspect of the geometric notion and abstracted it, so that we can apply it to spaces that can not be directly related to surfaces. The synthetic point of view is more interested in the qualitative rather than quantitative aspects. In our case, we are particularly interested in the definition of curvature for a weighted graph, whose node set is a discrete space. A survey about synthetic Ricci curvature (closely tied to Gaussian curvature), which highlights many concepts behind this research field, is [Villani, 2016].

Our aim is to define a curvature on a weighted, undirected graph $G = (V,E)$. To achieve this with a subtractive approach like that represented by the quantity in (4.19), we have first to understand what case we want to treat as the standard, flat structure (our Euclidean plane), compared to which another structure is considered to be curved. In Section 4.1.1 we define the Jaccard distance d_J on node neighbourhoods and its weighted counterpart d_J^ω. As we already observed, the unweighted distance d_J does not consider the weighted structure of the graph and is based just on the topology of G, that is, it takes into account just if any two nodes are adjacent or not. The neighbourhood of each node is treated as a word bag where only the binary variable of absence or presence of a term counts, and where we neglect the strength of the correlation between two nodes. From another point of view, we are using vectors with binary entries (as was discussed in Section 4.1.2; see also Section 2.1.4). In a sense, with d_J we are making the naive assumption that each word appears independently from the others and with uniform probability. Under this light, we can consider d_J as the standard, zero-curvature case: we are not considering anything else than the underlying, topological structure of the graph. On the contrary, distance d_J^ω assigns to each node in a neighbourhood its relative importance with respect to the central node. The result is very often different from what we obtain through d_J: two nodes v and w tend to be nearer under d_J^ω than under d_J if the weighted structure of G values the connections between common neighbours of v and w heavily, and viceversa, d_J^ω separates them if the same connections are less relevant than the connections of v and w with other nodes. This leads us to propose as curvature between two nodes v and w on a generic weighted, undirected graph G the difference

$$\Xi(v,w) = d_J(v,w) - d_J^\omega(v,w). \tag{4.20}$$

Hence, a positive curvature Ξ represents a stronger bond between two words, whereas a negative or null one might point out to the fact that v and w co-occurred randomly. We remark that this concept of curvature, as for the Jaccard distances of Section 4.1.1, is not tied to the presence or absence in G of an edge between v and w.

The quantity Ξ is *a priori* bounded to the interval $[-1,1]$ if weights are positive (see the remark on generalizing d_J^ω in Section 4.1.2), and represents a new weight function $\Xi : V \times V \longrightarrow [-1,1]$ on G. We can express it explicitly in terms of node neighbourhoods, combining definitions (4.2) and (4.5), as[11]

$$\Xi(v,w) = \frac{|\bar{N}^\omega(v) \cap \bar{N}^\omega(w)|}{|\bar{N}^\omega(v) \cup \bar{N}^\omega(w)|} - \frac{|\bar{N}(v) \cap \bar{N}(w)|}{|\bar{N}(v) \cap \bar{N}(w)|}.$$

A vectorial interpretation is possible using the relations (4.6), (4.8), (4.9) and (4.10) of Section 4.1.2.

We observe that there are just two cases in which $\Xi(v,w)$ will be zero: either

- v and w have disjoint neighbourhoods, so that

$$d_J(v,w) = d_J^\omega(v,w) = 1,$$

  or

- the weights in both neighbourhoods are all equal.

Given our considerations, the new weight function Ξ implicitly defines a clustering of V. We only retain the set E_+ of edges defined by a positive curvature, i.e.

$$E_+ = \{(v,w) \mid v,w \in V, \Xi(v,w) > 0\}$$

and define the graph $G^+ = (V,E_+)$. The graph G^+ is not necessarily a subgraph of G, as *a priori* we have that $E_+ \not\subset E$. We take the connected components of G_+ to be our clusters.

We can represent distances in the graphs by means of distance matrices D and D^ω respectively for the normal and weighted Jaccard distance on G, as discussed in Section 2.1.4 and Section 4.1.2. The complexity of this step is estimated by (4.12). Then, the matrix

$$\Xi(G) = D - D^\omega$$

represents all the curvatures in G. The associated matrix

$$\Xi^+(G) = \{\max(\xi_{ij},0) \mid \xi_{ij} \in \Xi(G)\},$$

[11] We observe that Ξ can be generalized as the difference of any couple of real functions on $V \times V$, especially when one represents a weighted version of the other.

where we substitute all negative curvatures with 0's, corresponds to the weighted adjacency matrix of G_+ (see Section 2.1.4). Computing the difference of the two matrices has a complexity of at most $O(|V|^2)$, but just considering non-zero entries[12] of D we can reduce it (confront the discussion in Section 4.1.3). From $\Xi^+(G)$ it is possible to retrieve all the connected components of G_+. Algorithms to find the connected components of a graph can run in linear time $O(|E_+| + |V|)$ [Hopcroft and Tarjan, 1973]. Therefore, the procedure we just described for curvature-based clustering has an overall, less-than-quadratic with respect to V, time complexity smaller than $O(|V|^2 + |V|)$, to which we have to add the one coming from the computation of the Jaccard distances on G.

It is possible that our curvature-based clustering algorithm produces a very fragmented partition of V with many singletons. As was proposed for the gangplank and the aggregative clustering algorithm, we can subsequently perform a recompacting step similar to that of Algorithm 3. We treat all clusters whose size falls under a given threshold θ (typically 2, so singletons) as single unities to reassign to bigger clusters. If $\mathcal{C}^+$ is the set of clusters such that $|C| > \theta$ for all $C \in \mathcal{C}^+$ and K is a cluster to reassign, we look for

$$C_{max} = \arg\max_{C \in \mathcal{C}^+} \max_{v \in K, w \in C} \Xi(v,w),$$

that is, the cluster that is connected to K by the edge with maximal curvature. Clearly, this maximal curvature will be non-positive, since all positive curvatures are already defining the clusters. We reassign small clusters progressively: it is possible for C_{max} to be initially empty for a cluster K if there are no edges between K and any C. We iterate over the clusters we want to reassign until we find a C_{max} for one of them, and then check again for the remaining ones. As the bigger clusters grow, each smaller cluster will find its final destination. In the extreme case that all clusters are too small, we will take the trivial clustering $\{V\}$.

4.3 Algorithm implementations for WSI

Section 4.1 revolves around the theoretical and conceptual groundings upon which the three algorithms presented in Section 4.2 are based. While the gangplank, the aggregative and the curvature-based algorithm can each be used on every generic graph for which the right prerequisites are

[12] Given that the Jaccard distance is non-negative by definition, if an entry of D is equal to 0 we know that the corresponding entry of $\Xi(G)$ will be non-positive.

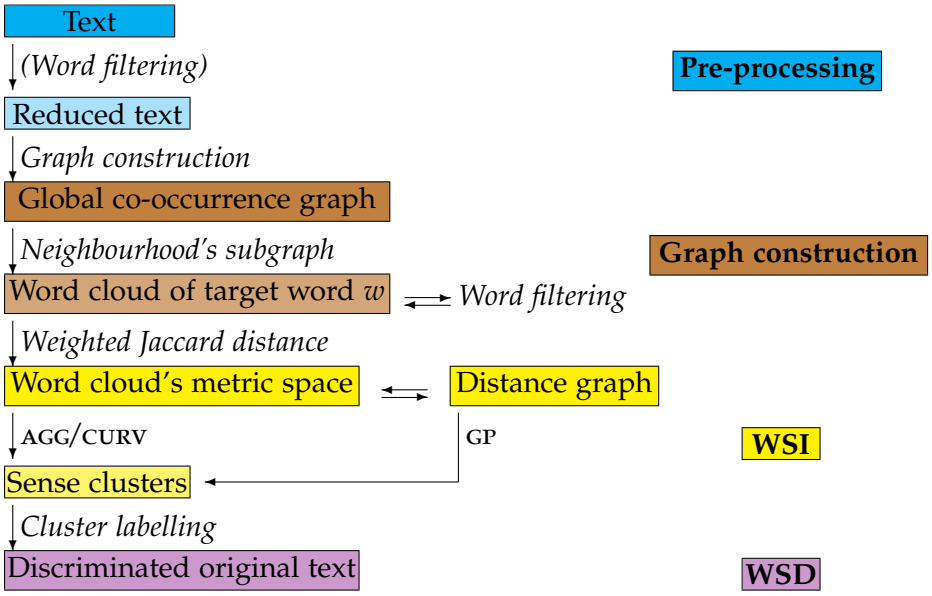


Figure 4.2: General rough framework of the implementation of the novel graph clustering algorithms for WSI proposed in this chapter. AGG, CURV and GP are respectively the aggregative, the curvature-based and the gang-plank clustering algorithms.

satisfied, they have been all first conceived with the task of Word Sense Induction in mind. We will now give the common framework for their specific implementation, following the layout seen in [Cecchini and Fersini, 2015], exemplified in Figure 4.2.

4.3.1 Pre-processing

There are a number of operations that have to be performed on a raw, unstructured text to transform it into a suitable word graph on which a clustering algorithm can be run. Here we will give a brief overview of these pre-processing steps.

The starting point for our graph-based approach to Word Sense Induction is a co-occurrence word graph G extracted by a given data set of text documents, as discussed in Section 2.1.8. The graph $G = (V,E)$ is weighted and undirected. In the simplest case, we take the weight function to be the raw frequency, assessing the number of times two words co-occur in the same text unit, but other significance measures (log likeli-

hood ratio, normal or adjusted mutual information,...) can be employed (for some of them, see Section 2.2). The goal is to link to each edge (v,w) of G a quantity ideally proportional to the significance of the co-occurrences of v and w: the higher the weight, the more significant their co-occurrence.

The context for which significance is measured is tied to a particular text unit, which can be any recurring portion of a textual data set, ranging from a fixed-size window around a word to an entire paragraph. Of course, other kinds of contexts are possible. We assume that the text is already subdivided in the chosen text units. For some documents, such as a collection of tweets, this is rather trivial. However, in the following, we will take the *sentence* as our basic text unit, without loss of generality. The problem of sentence boundary detection presents its own challenges (a survey for English is [Read et al., 2012]), but there are well established and reliable unsupervised systems, such as [Kiss and Strunk, 2006], that come to our aid.

The most interventional action in our graph-building step is the use of a part-of-speech tagger (like TreeTagger [Schmid, 1994] or ARK tweet POS-tagger [Owoputi et al., 2013]) to keep only certain grammatical classes of words as nodes in our graph. Usually, we want to retain only open-class words, in particular nouns and verbs. Members of so-called *open classes* are considered to carry meanings and/or refer to semantic entities. On the contrary, the role of most members of *closed classes*, like determiners or prepositions, is that of function words, joining and completing syntagms so that an utterance becomes grammatically correct. The traditional word classes for English are nouns, verbs, adjectives, adverbs, prepositions, conjunctions, pronouns, determiners (articles) and interjections. The first four are considered to be open classes; the second four closed classes; and interjections constitute a factually very heterogenous class which defies a clear definition. The role of interjections, compared to other classes, is much less related to the syntactical structure of a sentence than to its expressive, pragmatical aspects. Most WSI approaches focus on the four cited open classes. Still, among them nouns and verbs appear to play a more prominent role in conveying information and referring to real or abstract entities, whereas adjectives and adverbs have the secondary function of modifying nouns and verbs[13]. Thus, many approaches restrict their attention just to nouns and verbs, and between the two, nouns are often deemed more relevant to the disambiguation task. Another reason is that often nouns present a simpler morphology than verbs, even in very flec-

[13]Here we are presenting a rather classical point of view relative to English and related languages, like others in the Indo-European family, or languages with a similar structure.

tional languages, testifying their relative smaller complexity. Of course, the notion of *part of speech* varies across different languages, and distinctions between word classes can get blurred at times even in the same language. For a general introduction to these topics, see [Lyons, 1968].

More relevant			Less relevant
Nouns	Adjectives	Pronouns	Interjections
Verbs	Adverbs	Conjunctions	
		Prepositions	
		Determiners	

Table 4.1: Word classes sorted by relevance for WSI. Nouns are highlighted as the most relevant class.

Wrapping up all previous considerations, we will assume that in the general case our word graph G represents sentence-level co-occurrences between members of noun and verb classes, and that weights correspond to raw frequencies. We found adjectives to be not enough significant and in particular to lower the quality of the clusterings: most of the more frequent adjectives, like *good* or *strange*, have too rarefied meanings and co-occur with too many nouns. In fact, they act like stop words[14], but are much more difficult to identify, hence our decision. We lemmatize words, i.e. reduce all words to a common lemma independent from possible inflections, and do not keep the distinction between lowercase and uppercase letters. The Jaccard distances that we will make use of are already some sorts of similarity scores; like likelihood ratios or mutual information, they are in fact computed based on some kinds of frequencies, which are the only information that we can extract directly from a text. Computing graph Jaccard distances on top of other similarity measures might blur the whole process by adding yet another layer of representation to our WSI system, but we do not completely exclude this possibility.

This particular setting is nearly identical to that of [Cecchini and Fersini, 2015], where tweets instead of sentences were selected as text units. We will adopt it for the sake of a concrete example, although many other choices are equally viable.

The word graph $G = (V,E)$ we define is a global model of our data set. As stated in Tasks 2 and 3 in Section 1.3, the target of our disambiguation is each time a given word $w \in V$, on which we focus our attention. In graph terms, the context of w is any appropriate subset $N \subseteq V$ such that $w \in N$. The most natural choice in our setting, and the one that was also adopted e.g. by the DW and HyperLex algorithms, is to take the **open** first-degree neighbourhood $N(w)$ of w in G (see Section 2.1.5). In [Cecchini

[14] See Section 2.1.8.

et al., 2015] and [Cecchini and Fersini, 2015] this open neighbourhood is called *word cloud*, a more colourful terminology we might also use later on. We are interested in the relationships between words in w's context that determine a subdivision of $N(w)$ in sense clusters, i.e. clusters implicitly defining a particular sense or usage of w in the data set. Our way to gauge such relationships is the weighted Jaccard distance of Section 4.1.1 on the subgraph induced by $N(w)$, also called w's *ego graph*. Including w in its own neighbourhood would warp this induced subgraph, in that the maximum path length between any two nodes would be 2, altering the computation of the Jaccard distance. We want to consider relationships between w's neighbours reflecting w's senses, not between w and its neighbours. In this perspective, w is the common element that does not provide further useful information for discriminating between senses.

We also notice that extracting a local subgraph of w from the global word graph does not yield the same result as building a word graph directly from the context of w. In the latter case, we might lose information (in the form of frequency) about word pairs co-occurring together in sentences where w does not appear. Even if such co-occurrences are not directly related to w, they nonetheless tell us something about the bonds between the involved words. Some connections inherited from the global graph G might not even be present in a locally built context graph. Therefore we prefer using the global word graph as our knowledge base whence to extract local contexts each time, where we will be able to apply our clustering algorithms. This will be the premise in the following Sections 4.3.2, 4.3.3 and 4.3.4 detailing the clustering step.

After a clustering of the word cloud has been obtained, the labelling and discrimination step may follow, sketched in Section 4.3.5. This last step briefly addresses what can be formally defined as Word Sense Discrimination.

4.3.2 Gangplank algorithm implementation

As explained in Section 4.3.1, we are choosing a target word $w \in V$ for disambiguation and running the gangplank clustering algorithm on the induced subgraph $G_w = G\langle N(w)\rangle$. Algorithm 4 requires a gangplank labelling function $g : N(w) \times N(w) \longrightarrow \{0,1\}$. Since we want to use second-order relations (context overlap) that may not be present in form of mere co-occurrences, we will compute gangplanks on the finite distance graph of $G_w = G\langle N(w)\rangle$. Schematically:

1. Given co-occurrence graph $G = (V,E)$, we choose a word w to disambiguate;

2. we induce the subgraph $G_w = G\langle N(w)\rangle$ from its open first-degree neighbourhood;

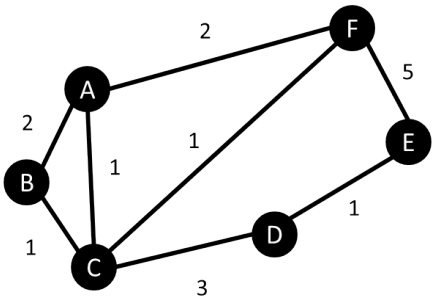
3. we compute the finite distance graph $D_w = D_{G_w,d_J^\omega}$ (Section 4.1.3) according to the weighted Jaccard distance d_J^ω on G_w (Section 4.1.1);

4. we define the gangplank labelling function g on D_w as $g(v,w) = 1$ if (v,w) is a gangplank in D_w (Section 4.1.4) and $g(v,w) = 0$ otherwise;

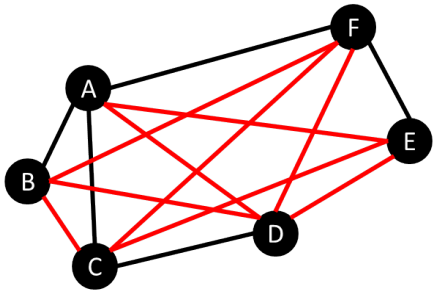
5. we finally run the gangplank clustering algorithm on D_w according to g;

6. we possibly perform a recompacting step (Algorithm 3).

We illustrate this process through the example of a very simple graph. In Figure 4.3 we show the small initial graph G_w (which is isomorphic to that in Figure 4.1) that has been extracted from the global graph G. Its distance graph, where gangplank edges are marked in red, is shown in Figure 4.3b.



(a) The initial weighted graph G_w.



(b) Its finite distance graph D_w. Red edges are gangplanks.

Figure 4.3: A small graph and its corresponding distance graph. Red edges are gangplanks.

The exact distances between nodes of G_w, which are the edge weights in D_w, used to compute the gangplank labelling g, are shown in Table 4.3, while each node's open neighbourhood as a multiset from which g is derived is shown in Table 4.2. From there, we see e.g. that

$$\bar{N}^\omega(A) \cap \bar{N}^\omega(F) = \{(A,2),(C,1),(F,2)\}, \tag{4.21}$$
$$\bar{N}^\omega(A) \cup \bar{N}^\omega(F) = \{(A,2),(B,2),(C,1),(E,5),(F,5)\}, \tag{4.22}$$

with cardinality respectively 5 and 15, resulting in a distance of

$$d_J^\omega(A,F) = 1 - \frac{5}{15} = \frac{10}{15} = \frac{2}{3}.$$

Node	Open neighbourhoods as multisets	Cardinality
A	$\{(A,2),(B,2),(C,1),(F,2)\}$	7
B	$\{(B,2),(A,2),(C,1)\}$	5
C	$\{(C,3),(A,1),(B,1),(D,3),(F,1)\}$	9
D	$\{(D,3),(C,3),(E,1)\}$	7
E	$\{(E,5),(D,1),(F,5)\}$	11
F	$\{(F,5),(A,2),(C,1),(E,5)\}$	13

Table 4.2: The open neighbourhoods of each node in graph G_w of Figure 4.3 represented as a multiset, together with their cardinalities, in the notation of (4.4) in Section 4.1.1.

Node	A	B	C	D	E	F
A	–	2/7	2/3	12/13	7/8	2/3
B		–	8/11	10/11	1	4/5
C			–	2/5	8/9	16/19
D				–	7/8	8/9
E					–	2/7
F						–

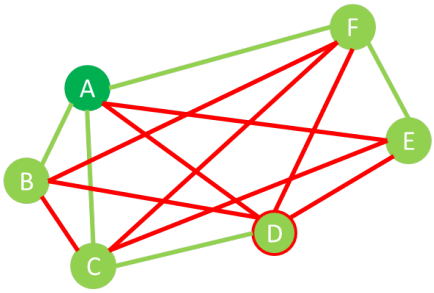
Table 4.3: Distances between nodes of G_w = edge weights of D_w of Figure 4.3.

Figure 4.4 shows how Algorithm 4 runs on D_w step by step. Here, no recompacting step is necessary, since all clusters have cardinality at least 2. We notice that, even if A, B and C form a clique, C is drawn outwards towards D, and indeed the fact that (B,C) is a gangplank edge ultimately makes the algorithm decide to detach C from A and B. We remark again that if we had just removed all gangplank edges, the graph would have stayed connected and we would not have recognized the stronger bounds between some of the node couples.

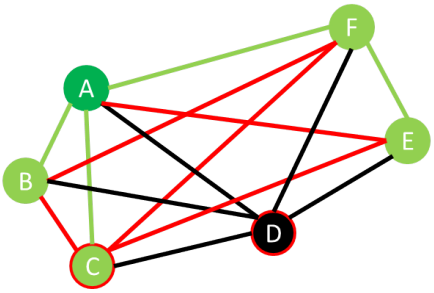
4.3.2.1 Minimum cut variant

In Section 2.1.3 the concept of minimum cut for a weighted graph was introduced. Applying it to the gangplank clustering algorithm, we want to find a minimum node cut M on the distance graph D_w relative to target word w as a preliminary step before actually running the algorithm. We are motivated to do this in an attempt to remove nodes that act as connectors between denser regions of the graph and thus might cause the algorithm to merge clusters that should be kept distinct. The gangplank clustering algorithm resumes then as normal on the graph

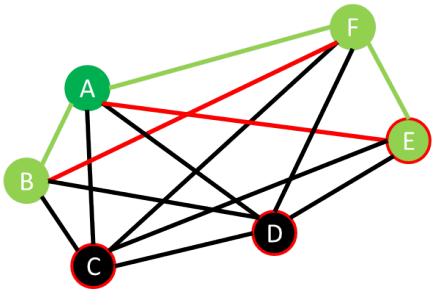
$$D_w \backslash M = D_w \langle V \backslash M \rangle.$$



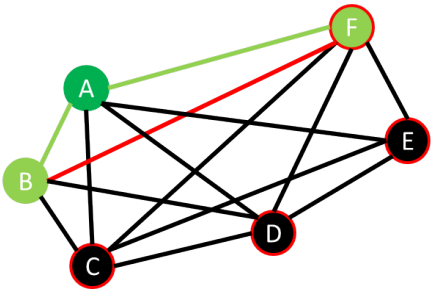
(a) A is chosen as the seed node of the first cluster. D is highlighted for having the greatest value γ.



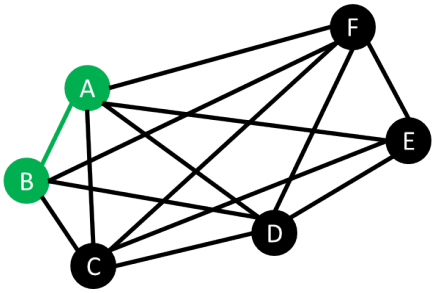
(b) D is removed and C is selected as the next node to remove for having the greatest value of γ.



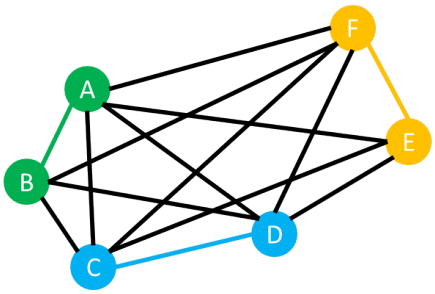
(c) C is removed. The tie at the level of γ score, degree and connection to A is broken by the greater weight on AE, $7/8$ against $2/7$ of AB . E is the next node to be removed.



(d) E is removed. A new, closer tie ensues between B and F. For the same reasons of step 4.4c, F is selected to be removed because $2/3 > 2/7$.



(e) No further expansion is possible. The first cluster is $\{A,B\}$.



(f) The final clustering $\mathcal{C} = \{\{A,B\},\{C,D\},\{E,F\}\}$ of the graph at the end of the algorithm.

Figure 4.4: An example run of the gangplank algorithm on the distance graph of the small graph of Figure 4.3. The quantity γ is the gangplank degree (4.16) of Section 4.2.1.

We do not perform the cut if D_w is complete, as this would mean to remove all but one node. After the clustering, each node of M that was removed from D_w is reassigned to the cluster to which it is connected with the smallest gangplank degree (4.16) in D_w, similarly to the process described by Algorithm 3. Our implementation of a minimum cut algorithm is that found in Python's package *NetworkX* [Schult and Swart, 2008], which is based on [Esfahanian, 2012]. Clearly, even if we chose to associate this variant to the gangplank algorithm, it is possible to use the minimum cut as a preliminary step for any of the other clustering algorithms we described in this chapter.

4.3.3 Aggregative algorithm implementation

Our aggregative algorithm is run on a metric space. As pointed out in Section 4.1.3 (with the *caveat* of Observation 4.1.1), we know that the weighted Jaccard distance d_J^ω defines a metric space on any given weighted graph. Having chosen w as the target word to disambiguate, we again consider the subgraph $G_w = G\langle N(w)\rangle$ induced by its neighbourhood and consequently the metric space (G_w, d_J^ω) (or, equivalently its distance graph D_w) on which we can run Algorithm 4.3.3.

For the practical part, we use the simple implementation of the k-medoid algorithm found in [Bauckhage, 2015]. The only novelty with respect to the general case described in Section 4.2.2 is that we fix a criterion for choosing the starting node in the medoid initialization step: As the seed of the first cluster we take the node w with highest degree in G_w. The motivation for this is the consideration that such a node is roughly speaking "more central" in the graph. Besides the notion of degree as a centrality measure[15] (as observed in Section 2.1.1), we observe that the greater w's neighbourhood, the more neighbours it will also have in the finite distance graph of G_w. In the metric space, we can imagine this as a probably more densly populated region. Starting there, we envision to capture already in the initialization step the core of one of the bigger clusters. Of course, the clustering's granularity depends on the radius σ. Since Jaccard distance is bounded, we can choose a σ in the interval $[0,1]$, knowing that $\sigma = 1$ leads to the trivial clustering $\mathcal{C} = \{N(w)\}$. As $N(w)$ is a discrete set, a sensible choice for σ, as discussed in Section 4.2.2, lies in $[d_{min}, d_{max}]$, where d_{min} and d_{max} are respectively the minimum and maximum distance between nodes in G_w.

[15] For a general discussion over centrality measures in a graph, see [Koschützki et al., 2005].

Let us take the graph G_w of Figure 4.3a. Its distances in Table 4.3 define the corresponding metric space. Here, we can choose a σ in [2/7,12/13]. More precisely, the distance sequence (see Section 4.2.2) is

$$\Delta = \{\frac{2}{7}, \frac{2}{5}, \frac{2}{3}, \frac{8}{11}, \frac{4}{5}, \frac{16}{19}, \frac{7}{8}, \frac{8}{9}, \frac{10}{11}, \frac{12}{13}, 1\},$$

with 11 elements, meaning that we have 9 possible non-trivial clusterings of G_w linked to the eight possible choices of the type $\sigma_i \in [d_i, d_{i+1})$, $i = 1, \ldots, 9$ where d_i is the i-th element in Δ. Our starting point will always be C, which has degree 4. The mean distance is 0.74, a possible value for σ_4, which we will use in our example. Figure 4.5b shows the clustering process in this case. Figure 4.5 shows all possible clusterings of G_w.

Interestingly, we notice that the middle values of σ_2, σ_3, σ_4, σ_5, σ_6 all yield the same clustering, corresponding to the one already found by the gangplank clustering algorithm in section 4.3.2. On the basis of this observation, we might see this clustering as the stablest configuration for G_w. We also remark a shift from two possible bipartitions of G_w, obtained with σ_7 on one side and with σ_8 or σ_9 on the other side, preluding to the trivial clustering for any $\sigma \geq 12/13$.

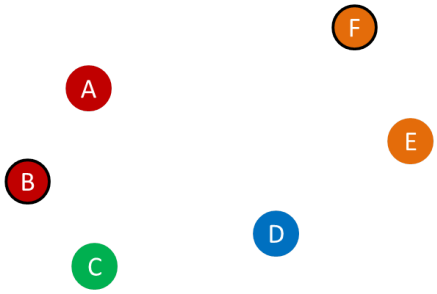
We know that we will find a single cluster if σ is greater than the maximum path distance from C (its eccentricity, see Section 2.1.2), which is 8/9. This corresponds to cases σ_8 and σ_9. For σ_1 the clustering is identical to that found with the gangplank clustering algorithm in Figure 4.4.

Node	A	B	C	D	E	F
A	–	1/4	1/5	5/6	5/6	2/5
B		–	2/5	4/5	1	3/5
C			–	2/3	2/3	1/2
D				–	1/2	3/5
E					–	3/5
F						–

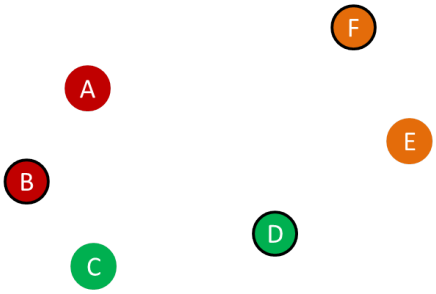
Table 4.4: Unweighted Jaccard distances between nodes of the graph in Figure 4.3.

4.3.4 Curvature algorithm implementation

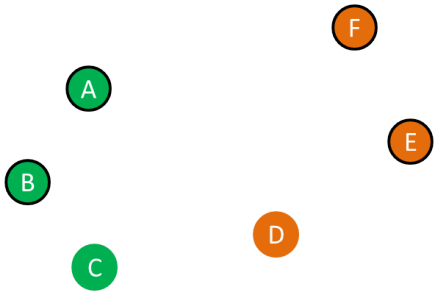
As laid out in Section 4.3.1, we run the curvature clustering algorithm of Section 4.2.3 on the word cloud G_w induced by the neighbourhood of a given node w in the global word graph G. The subgraph inherits the weights (raw frequencies) of G, so that we can compute the local weighted and unweighted Jaccard distances and obtain the edge curvatures.



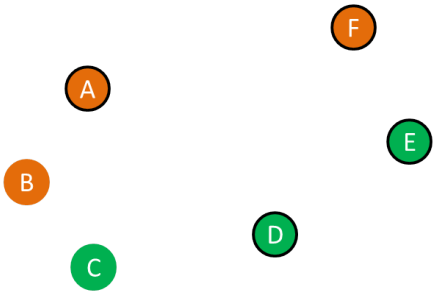
(a) The aggregative clustering obtained with $\sigma_1 \in [2/7,2/5)$ after 2 iterations.



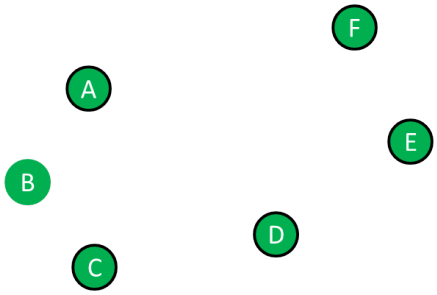
(b) The aggregative clustering obtained with σ_2, σ_3, σ_4, σ_5, σ_6, i.e. $\sigma \in [2/5,7/8)$, after 3 iterations.



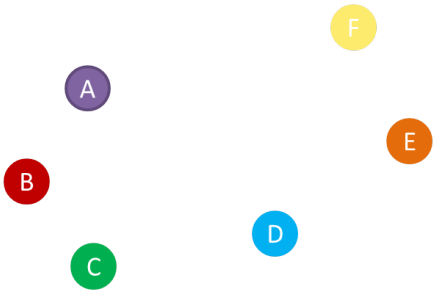
(c) The aggregative clustering obtained with $\sigma_7 \in [7/8,8/9)$, after 3 iterations.



(d) The aggregative clustering obtained with σ_8 and σ_9, i.e. $\sigma \in [8/9,12/13)$, after 4 iterations.



(e) The trivial clustering obtained with $\sigma \geq 12/13$, after 4 iterations.



(f) The other trivial clustering obtained with $\sigma < 2/7$, after 1 iteration.

Figure 4.5: All possible clusterings of the graph of Figure 4.3a, as we let σ vary. The respective medoids are the integrally coloured nodes.

We will take again the example of the graph G_w seen in Figure 4.3a. The weighted Jaccard distances are shown in Table 4.3. Table 4.4 presents their weighted version.

Pairwise subtracting the entries of Table 4.4 from those of Table 4.3, we have the sign of the curvatures in Table 4.5:

Node	A	B	C	D	E	F
A		+	-	-	-	-
B			-	-	0	-
C				+	-	-
D					-	-
E						+
F						

Table 4.5: Curvature signs between nodes of the graph in Figure 4.3.

So G_w naturally decomposes into its positively curved components $\{A,B\}$,$\{C,D\}$ and $\{E,F\}$, that were already found by the gangplank algorithm (Section 4.3.2) and by the majority of values for σ with the aggregative approach (Section 4.3.3).

4.3.5 Discrimination

After the senses of a target word w have been induced in the form of a sense clustering $\mathcal{C}$ of its neighbourhood, we can also go a step further and discriminate the occurrences of w in our data set. This is the proper Word Sense Discrimination part in the pipeline shown in Figure 4.2, according to Task 3 of Section 1.3. Ideally, $\mathcal{C}$ identifies $|\mathcal{C}| = i$ different senses. Still, some of the clusters might just represent common word associations with no discriminative power. We formalize this consideration through Algorithm 6.

Algorithm 6 Discrimination step

Input Word w, sentence $\mathcal{S}$ with w, sense clustering $\mathcal{C}$ of G_w

Output Sense cluster $C_{\mathcal{S}}$ identifying w in $\mathcal{S}$

1: $C_{\mathcal{S}} = \emptyset$ ▷ Initialize the cluster
2: $\sigma = 0$ ▷ Initialize the overlap
3: **for** $C \in \mathcal{C}$ **do**
4: **if** $|\mathcal{S} \cap C| > \sigma \wedge |C| \geq |C_{\mathcal{S}}|$ **then**
5: $C_{\mathcal{S}} = C$
6: $\sigma = |\mathcal{S} \cap C|$
7: **end if**
8: **end for**
9: **return** $C_{\mathcal{S}}$

Algorithm 6 treats the sentence $\mathcal{S}$ as a word bag $\{s_1, \ldots, s_n\}$, without taking into account word order. In general, we might argue that word order is not determinant in defining the meaning of a word in a sentence. In fact, word order is governed by the syntactical and grammatical rules of a given language and may vary arbitrarily. Moreover, we are not interested in the exact relations between the entities referred to in $\mathcal{S}$: e.g., if *the peasant bows before the king* or *the king bows to the peasant*, either way *bow* is used for a courteous greeting. So, we just want to compare $\mathcal{S}$ to each cluster C_i of $\mathcal{K}$ and see for which one we obtain the absolute greatest overlap $|\mathcal{S} \cap C_i|$. Alternatively, we might also consider the Jaccard coefficient $\frac{|\mathcal{S} \cap C_i|}{|\mathcal{S} \cup C_i|}$, or similar ones, to measure a relative overlap. A motivation for this could be that we do not want bigger clusters to be too important or dominant in the discrimination process. However, a relative overlap favours too much smaller clusters, which may well be the less significant ones. Further, as discussed in Section 1.1.1, the sense distribution of a word tends to be very skewed towards one more popular sense, so that a bigger cluster might capture this trend. Part of these considerations depends on how the pre-processing step was performed. In our case, we assume to remove every function word and retain only nouns and verbs. If, on the contrary, every word were kept, we would risk obtaining a big catch-all clustering which monopolizes discrimination, as function words are so pervasive. For all these reasons, and especially in our setting, we prefer leaning towards an absolute overlap.

We select $C_{\mathcal{S}} = \arg\max_{C \in \mathcal{C}} |\mathcal{S} \cap C|$ as the implicit sense representation to associate to w in sentence $\mathcal{S}$. If there is a tie, we choose the bigger of the possible clusters, on the ground that it likely identifies the most common sense, and if the tie persists, we just pick one randomly. This way, if V is the vocabulary of our data set, that is, the set of word forms that we consider distinct[16] for the purposes of our Word Sense Induction process, we are generically defining a mapping

$$\ell_w : \mathcal{P}(V) \longrightarrow \mathcal{P}(N(w)) \cup \{\emptyset\}.$$

Sentence $\mathcal{S}$ in fact induces a particular subset $\mathcal{S} \cap V$ of V, and a cluster of $\mathcal{C}$ is a subset of w's neighbourhood. We can see the final result as an expansion of $\mathcal{S}$ through words associated to a given meaning of w. In an unsupervised setting, we can not achieve more without resorting to external knowledge bases: we are just explaining a word through associations to other words seen in the data set. We observe that not every cluster of $\mathcal{C}$ will surface in the image of ℓ. Some word associations might be spurious

[16]For example, *peach* and *peaches* are considered the same if we use lemmatization during pre-processing; cf. Section 2.1.8 and Section 4.3.1.

and their appearance always outnumbered by more significant sense clusters, so that in the end the effective number of different senses of w in the data set might be sensibly inferior than $|\mathcal{C}|$.

As a final comment, we can think of performing an actual disambiguation, i.e. linking each discriminated occurrence of w to a knowledge base where its sense is explicitly described, by means of an algorithm similar to the one we used for discrimination. This time, we would compare the combined context of a sense of w to the entries of a knowledge base, trying to determine the best match. However, here we will not delve deeper into this question, just pointing out among the others to [Moro et al., 2014], a work that has tackled a similar task.

4.4 Conclusion

The new graph-based clustering algorithms presented in this chapter have been outlined in their general forms (Section 4.2), with a robust underlying theoretical foundation (Section 4.1), and detailed in their actual implementation for the Word Sense Induction task (Section 4.3), with basic examples of their functioning. In the next chapter we will propose an evaluation framework based on pseudowords, putting to use the algorithms presented in this chapter and comparing their results to those of the algorithms examined in Chapter 3. We will try to establish how an algorithm's performances are influenced by a given WSI task and other factors, like the type of word graph.

Chapter 5

Pseudoword evaluation framework

In this chapter we will present a new pseudoword evaluation framework consisting of two different word graph data sets, one based on co-occurrences and the other on semantic similarities (see Section 2.1.8), on which we will compare the performances of the graph-based clustering algorithms for WSI seen in Chapters 3 (apart from HyperLex) and 4. First, we will introduce the concept of *pseudoword*, how it came to be and how it is used for the evaluation of Word Sense Induction algorithms (Sections 5.1 and 5.2). Generally, as noted in Section 1.3.3, the evaluation of unsupervised systems bears many difficulties and pitfalls that derive from the unsupervised nature of clustering itself. Contrarily to a classification problem, clustering works with no clear guidelines apart from the goal to reveal some kind of patterns in the data. This makes it difficult to find an evaluation framework capable of highlighting what a clustering algorithm has really accomplished or not, without the risk of overrating trivial baselines. Pseudowords have been introduced to circumvent some of such problems, offering a way to obtain an objective and self-contained evaluation.

On a more practical level, in our opinion current WSI and WSD challenges, such as e.g. those of SemEval 2010 [Manandhar et al., 2010], present some shortcomings. A fundamental problem is the vagueness regarding the granularity (fine or coarse) of the senses that have to be determined (see Section 1.1.1). As a consequence, the definition of an adequate evaluation measure becomes difficult, as many of them have been shown to be biased towards few or many clusters (cf. Section 5.4.1.1). Further, small data sets often do not allow obtaining significant results. Pseudoword evaluation, on the contrary, presents a framework where the classification task is well characterized and gives the opportunity to define an *ad hoc* evaluation measure, at the same time automating the data set creation.

For these reasons, for now we are not interested in evaluating the WSI clustering algorithms described in Chapters 3 and 4 on other data sets like the SemEval ones: The great variability of the contexts of the target words would not allow us to easily interpret the qualities of the algorithms, and further, we are first interested in observing the behaviours of the algorithms for a well defined task, i.e. homonymy detection, before passing to other tasks. Our objective is to have a solid base for the relative comparison of algorithms and we deem their absolute scores of secondary importance, since every unsupervised evaluation possesses its own internal, not immediately generalizable logics. Conversely, we do not consider for evaluation systems that appear in SemEval runs, on one hand because of the practical difficulty to retrieve them, and on the other hand because we believe that the selection of algorithms presented in this work sufficiently covers a range of different clustering behaviours.

The second part of this chapter (Sections 5.3 and 5.4) tackles the following research questions, in the spirit of our work on graph-based clustering algorithms: What are the limitations of a graph-based pseudoword evaluation for homonymy detection? How does the structure of a pseudoword ego graph (see Section 5.2.1 and Section 2.1.5) depend on its components? How do different clustering strategies compare on the same data set, and what are the most suited measures to evaluate their performances?

5.1 Pseudowords

A *pseudoword* is an artificial lexical construct used to help in the creation of data sets for the evaluation of Word Sense Induction clustering algorithms.

Pseudowords were independently first proposed by [Gale et al., 1992b] and [Schütze, 1992]. The underlying idea is to treat usually two, but sometimes three or more, different words in a given data set as one and the same word: we treat the occurrences of the first word as if they were also occurrences of the other word. From the point of view of Word Sense Induction, where we are interested in modelling a word's context, this is equivalent to joining the contexts of both words. We are effectively creating a new, non-existent word from the conflation of two distinct terms, a *pseudoword* which potentially carries all the senses of its original components. A pseudoword is thus an artificial lexical object that is the mere sum of its parts. An example might be *banana_door*: this pseudoword encompasses all and only the meanings of *banana* and *door*. So, if our data set included the sentences

- This afternoon I ate a *banana*.

- Remember to close the *door*!

we would treat *banana* and *door* as instances of the same fictive word *banana_door*.

We are using the graphical convention of writing A_B to denote the pseudoword arising from words A and B, instead of the hyphenated A-B, because the latter would apparently refer to an existing entity[1]. We remark that a pseudoword is not a compound word: e.g. *air-crew* is a particular kind of crew distantly related to the notion of air, but it does not comprise the sense of air itself, nor is every crew an air-crew. However, the pseudoword *air_crew* would just be the union of all the senses of *air* together with those of *crew*, and nothing else.

If we denote the context of a word v generally as $C(v)$, we identify the context of v_w with

$$C(v\_w) = C(v) \cup C(w).$$

In the previous example, following the guidelines for context modelling of Section 4.3.1, we would have:

$$\begin{aligned} C(\text{banana}) &= \{afternoon, eat\} \\ C(\text{door}) &= \{close, remember\} \\ C(\text{banana\_door}) &= \{afternoon, close, eat, remember\}. \end{aligned}$$

In Section 1.3 we discuss how, from the perspective of distributional semantics, word senses are determined by a word's context. It follows immediately that the senses of a pseudoword proceed from the union of the senses of its components, and that consequently a pseudoword roughly simulates ambiguity. Pseudowords bring two main advantages to WSI:

1. they allow greatly expanding data sets of ambiguous words using only few building blocks and no human intervention;

2. knowing the sense distribution of the components translates into knowing the sense distribution of a pseudoword.

With regard to the first point, we observe that if we have n words with known distributions, there are $\binom{n}{2}$ pseudowords that can be generated from them, for a nearly quadratic increase of available terms.

The second point is crucial for the evaluation frameworks that have been proposed in the past and for the one that we want to propose: it means that, knowing the sense distribution of words v and w, we already know the ideal clustering of the context of v_w and that we can use this

[1]Many languages use, under different forms, the strategy of compounding words (a simple juxtaposition is very frequent, whereas hyphenation is typical of English orthography), as it happens e.g. for *play-group* or *air-crew* (also written *playgroup* and *aircrew*). For a general overview see [Scalise and Vogel, 2010]. For the specific case of English, [Lieber and Stekauer, 2009] is a rather exhaustive reference.

information to gauge the performance of a WSI algorithm on v_w. Despite the potentialities we have shown, before better defining the setting of pseudoword evaluation in Section 5.2 there are also issues that we need to confront.

We recall from Section 1.1 that proper homonymy arises when two semantically and etymologically unrelated senses are expressed by the same spoken or written sign, i.e. word, such as *count* in the sense of a *nobleman*, as opposed to *the act of enumerating*. This word is ambiguous because we can not assign it a meaning without first having some clarifying context. At the opposite side of the ambiguity spectrum lie polysemous words, whose different senses are more or less strictly related to each other and often revolve around a common concept. Combining random words will generate pseudowords that fall in between these two extremes. If v and w only possess unrelated senses, v_w will be equivalent to a case of proper homonymy. Sometimes, however, some kind of overlap between the semantic spheres of v and w is likely to exist, up to the ultimate point that the senses of v_w are blurred and not easily distinguishable anymore. For example, using two near-synonyms like *door* and *gate* will produce a pseudoword which is substantially equivalent to any one of its components. On another note, *door* and *window* will surely share an important amount of contexts, as they both express a metaphorical or architectural concept of *opening* in a building, and the resulting pseudoword will probably represent this more abstract notion. Even if the previous statement about the knowledge of a pseudoword's sense distribution still remains true, we notice that haphazardly mixing words will possibly yield rather unpredictable and unwanted results, in the sense that is not clear how to define the general type of ambiguity that a pseudoword simulates *a priori*. As we perform the process of pseudoword creation with three or more component words, this issue naturally sharpens.

A study of [Nakov and Hearst, 2003] shows that the performances of WSD algorithms used to disambiguate the senses of totally random pseudowords act as an "optimistic" upper bound with respect to the same performances on true polysemous words: while pseudowords might not be a case of perfect homonymy, their senses are still easier to separate in most cases, as they lack the usual kind of correlation between sense categories and sense distributions of a real polysemous term. Introducing some sorts of restriction, like requiring a minimum frequency for each component or taking components with similar distributions, we can obtain a pseudoword nearer to the real case, and in fact a drop in the disambiguation performances is noted. From a set-theoretical point of view, the quality of a pseudoword greatly depends on the cardinality of the intersection $C(v) \cap C(w)$. The smaller its cardinality, the better v_w will approximate a

case of homonymy. This point will be further investigated in Section 5.3.1. Essentially, we can claim that on the scale between clear-cut homonymy and fuzzy polysemy random pseudowords lie closer to the former; on the other hand, the more words are combined together, the more we approach the latter.

An attempt to steer the process of pseudoword creation towards a more realistic imitation of polysemous words is devised and put in practice in [Pilehvar and Navigli, 2013, Pilehvar and Navigli, 2014], themselves drawing inspiration from [Otrusina and Smrž, 2010]. Here the objective is to produce pseudoword molds of existing words with an arbitrary degree of polysemy, not necessarily limited to two senses. The method of both works relies on the semantic mapping found in WordNet[2]. There, the different senses of a term are identified by certain sets of synonymous words and expressions (lexemes), also called *synsets*. For example, *peach* will have a meaning similar to *peach tree, Prunus persica* in some contexts and to *smasher, stunner, ...* in others. Each lexeme in a synset has its own synsets, and there are also other kinds of relationships, like meronymy, hyperonymy, and so on. A word described by just one synset is considered to be monosemous. This structure makes it possible to explore the ensuing lexical tree graph in many directions and even to define a path distance on it, admitting all or just some relationships as valid steps. A pseudoword in [Pilehvar and Navigli, 2013] is constructed as the double of any polysemous word w and consists of the sum of one monosemous term for each sense of w conforming to its WordNet entry. Each such term has to be as close as possible to the synsets of the original word according to a given distance on the lexical tree: a crawl on WordNet recursively tries to find a monosemous exponent for each synset related to w, and uses it as one *pseudosense* of the final pseudoword.

Taking *horoscope* as an example, in [Pilehvar and Navigli, 2013] its relative pseudoword is *forecast_diagram*, synthesizing the two meanings distinguished by WordNet. This allows us to obtain a ground truth for algorithm evaluation directly from the data already at our disposal, with no need for human sense tagging. In our example, instead of gathering contexts for *horoscope* and tagging each occurrence with either of its two senses, we can just combine the contexts of *forecast* and *diagram*. If w has three or more senses according to WordNet, we will have to combine the needed number of pseudosenses. Pilehvar's pseudowords modelled on Gigaword [Parker et al., 2011] have found use e.g. in [Başkaya and Jurgens, 2016].

[2] `http://wordnetweb.princeton.edu/perl/webwn`; [Miller, 1995]

The *pseudocontexts* that we gather depend on the data set. If our data set does not contain *forecast* or *diagram*, or if any of these terms is not frequent enough, we will not be able to make a mold of *horoscope* on it. The system presented in [Pilehvar and Navigli, 2013], while claiming to model polysemy very accurately and providing useful pseudoword recipes, entirely depends on an external and human-compiled resource such as WordNet. It does not take into account that its sense distinctions might be too fine-grained or not relevant for the data set at hand, as was discussed for *protection* in Section 1.1.1. Essentially, it superimposes a lexical model on an *a priori* unknown data set. This brings us to another issue of pseudowords: where does the prior knowledge about its two components come from? In our unsupervised setting we prefer avoiding the use of external knowledge bases, and in Section 5.2 we will instead resolve to use a bottom-up approach to obtain our building blocks. Here, we just disclose that the difference between our work and [Pilehvar and Navigli, 2013] lies in the focus on algorithm comparison through pseudowords rather than on the definition of construction rules for pseudowords.

5.2 Pseudoword evaluation description

In Section 5.1 we have pointed out how randomly merging two words v and w to obtain a pseudoword v_w might result in unwanted side-effects. One of them is that the number of senses of v_w is not always clearly predictable. To keep the complexity of such lexical interactions as low as possible, we want to limit ourselves to the simplest case of pseudoword, namely: the **combination of two monosemous terms**. This way, our aim is to simulate a *disemous* (i.e. possessing two senses) case of homonymy.

Our motivation is that homonymy is usually more clear-cut than generic polysemy: the senses will be more distinguishable in terms of contexts, and their respective distributions will be independent from each other with good approximation. On the contrary, the senses of a polysemous words tend to have more intricate relationships and the line between one and the other is often hazy (cf. Section 1.1). WordNet's fine-grainedness is an example of how some subtle distinctions between different meanings may appear arbitrary at times. For this reason, we deem that the efficacy of a WSI algorithm should first be measured in the case of a well-defined homonymy before being tested in a more fine-grained and vague circumstance. Homonymy recognition itself is relevant for many tasks like lexical substitution, where the ability to distinguish completely different senses and uses is more important than the subtle distinction between facets of the same meaning. Also, we do not want our evaluation to depend on

the subjective granularity of an external lexical resource. For example, the sense distinctions in WordNet might not be mirrored in our data set, and conversely, some unforeseen senses might be observed.

Therefore, we will only deal with pseudowords composed of terms that are as unambiguous as possible, namely terms that in our data set are found to possess only one single sense. To explain how we define them, we have to outline the generic setting of a pseudoword evaluation in the following sections, before going into the details of our implementation in Section 5.3.

But first, we point out to another work where pseudowords play a major role as a means to evaluate a WSI algorithm, namely [Bordag, 2006] (which we also cite when discussing Chinese Whispers in Section 3.3). Effectively, our pseudoword evaluation framework bears many common traits with the one presented there. However, we remark some relevant differences. The most important one is that in [Bordag, 2006] the evaluation is "subjective", i.e. the triplet-based algorithm defined by the author is evaluated against itself: the sense-identifying clusters yielded by the algorithm for two words v and w are compared to those yielded by the same algorithm for the pseudoword v_w. On the contrary, as will be discussed in Section 5.2.2, for each pseudoword we set an objective ground truth and measure the ability of any given clustering algorithm to retrieve it. To this end, we also restrict the nature of our pseudowords (cf. Section 5.2.1), whereas in the cited paper combinations of more than two words, possibly of different lexical categories (cf. Section 4.3.1), and ambiguous components are admitted, and the context-defining terms of a word are not lemmatized as in our case. We can affirm that the pseudoword evaluation framework presented in this chapter, while describing a generic method for algorithm evaluation, is more focused on the task of graph-based homonymy detection rather than on the generic assessment of the validity of an algorithm, while at the same time it can surely be seen as an expansion of [Bordag, 2006] from the point of view of the pseudoword data sets. In particular, we note that we also consider semantic similarities alongside mere co-occurrences (see again Section 5.2.1) to build pseudoword ego graphs. Finally, as already stated in 5.1, a data set analysis is performed that takes into account not only the consequences, i.e. the scores obtained by the algorithms, but also the causes, i.e. the structure variation of pseudoword ego graphs.

5.2.1 Evaluation setting: word graphs and distributional thesauri

Our corpus is a combination of the Leipzig Corpora Collection[3] (LCC) [Richter et al., 2006] and the Gigaword corpus [Parker et al., 2011], and consists of 105 million English newspaper sentences. The corpus has been parsed using the Stanford Parser [De Marneffe et al., 2006] to extract syntactic dependencies. Following the approach found in *JoBimText*[4] [Biemann and Riedl, 2013], such dependencies are used for the computation of a term-context frequency-weighted version of pointwise mutual information called *lexicographer's mutual information* (LMI) (see Section 2.2.1.3) that allows us to obtain a similarity score between any two words. The goal is to create a first-order and second-order *distributional thesaurus* for each word w in the corpus, i.e. rankings of words from the most to the least similar to w according to the LMI computed either for syntactical contexts or co-occurrences. A word in the second-order distributional thesaurus of w does not necessarily co-occur with w in a sentence of the corpus. We notice that words are lemmatized and the difference between lowercase and uppercase letters is kept.

First we build co-occurrence ego graphs (see Sections 2.1.5 and 2.1.8). For a given term w, its *first-order distributional thesaurus* will use the lexicographer's mutual information to rank every word that co-occurs with w in a sentence. Based on these rankings, we choose a number N of significantly (syntagmatically) similar words to consider for each such distributional thesaurus. Then, for every word w, we build the graph G_w with the N highest ranked entries in its thesaurus as its node set V, so that $|V| = N$. An edge will be drawn between u,$v \in V$ if u is part of the first N entries of v's thesaurus or, viceversa, v appears in the first N entries for u. We notice that both conditions will not necessarily be satisfied at the same time, but LMI is symmetric and in either cases the score will be the same. Knowing this, we define a weight mapping on G_w by setting the weight of (u,v) equal to the respective LMI score. The graph G_w is then called the weighted *ego graph* of w. We remove w from G_w for the same reasons that lead us to work with open neighbourhoods as word clouds in Section 4.3.1. We might use some extra care to exclude stop words[5], which are already penalised by the PMI part of LMI, but which however keep appearing in the distributional thesaurus because of their high frequency.

[3] `http://corpora.uni-leipzig.de`

[4] `www.jobimtext.org`

[5] See again the considerations in Section 2.1.8.

We pass to the second order using the LMI on the first-order relations given by the syntactical dependencies, and build a *second-order distributional thesaurus* which can be interpreted as representing a semantic similarity word graph. With a procedure analogous to that used for building co-occurrence-based ego graphs, this time we build (semantic) similarity-based ego graphs.

Whichever its form, a semantic-similarity or co-occurrence-based ego graph G_w can be clustered to induce the word senses of w in form of sense clusters, as for the general process detailed in Section 4.3.1. We consider a word w to be *monosemous* with respect to a given clustering algorithm if the clustering of G_w yields a single cluster. As mentioned in Section 5.1, we refer to this approach as being bottom-up because it principally relies just on instruments already present in our evaluation setting and does not introduce new ones from outside. However, additionally we check that w has a single synset in WordNet. This double criterion is quite strict and, while the external look-up is not indispensable to our setting, it is helpful for avoiding the bias coming from algorithms which favour a coarse clustering.

To create our pseudoword ego graph data sets, we have chosen to take only **nouns** (cf. the discussion in Section 4.3.1) as the components of our pseudowords. All elements in the second-order distributional thesauri will also be only nouns, to allow us to compare their syntactic dependencies, whereas we do not restrict ourselves to any particular word class in the case of co-occurrences.

We divide all the nouns in the corpus into 5 logarithmic frequency classes, based on the frequency of the most common noun in the corpus, i.e. *year*, which appears a total of 6179666 times. We take the binary logarithm of this value, approximately 22.6, and assign the frequency class of a word according to where its logarithmicized frequency falls with respect to its quintiles[6]. For example, if a word w appears 10000 times in the corpus, the binary logarithm is approximately 13.3 and w will belong to frequency class 3. Conversely, a member of frequency class 3 will have a frequency between ca 526 and 12077. In general, the member of one of our frequency classes is expected to be 23 times as frequent in the corpus as a member of the next-lower class.

For each frequency class, we extracted random candidates and retained only 10 of them which were found to be monosemous by the above-mentioned criteria. In the end, we obtained 50 suitable words. The number of their combinations, and thus of different pseudowords, amounts

[6]The quintiles are the four values that divide a quantity in five parts: in this case, they are the multiples of ca 4.52, i.e. 4.52, 9.04, 13.56 and 18.08.

to $\binom{50}{2} = 1225$. We then proceeded to compute all their respective ego graphs. For v and w, we substituted all their occurrences in the corpus with v_w and went normally through the previously described steps to get a semantic-similarity and a co-occurrence-based ego graph G_{vw}. We remark that this graph-building process has to be performed separately for each pseudoword, as we want to merge only two contexts at a time. That is, for v_w we want to consider the generic words u and z as possible entries of the distributional thesaurus, whereas u_z would not have any sensible interpretation as part of the pseudoword's context. We limited the size of all distributional thesauri, and consequently the order of ego graphs, to $N = 500$ nodes, be they of regular words or pseudowords. For very infrequent terms, it is possible that G_{vw} will be of smaller order.

5.2.2 Structure of a disemous pseudoword and evaluation goal

A pseudoword v_w constructed with the criteria explained in Section 5.2.1 will bear the two respective senses of v and w. Looking at the graph G_{vw}, this translates into being able to tell if a node u belongs to either ego graph G_v or G_w, or if it can be found in both.

More formally, we denote the node sets of G_v, G_w and G_{vw} respectively as V_v, V_w and V_{vw}, and we write

$$\begin{aligned} V'_v &= V_v \cap V_{vw} \\ V'_w &= V_w \cap V_{vw}. \end{aligned}$$

The sets V'_v and V'_w thus represent all the terms in the reduced distributional thesauri of respectively v and w that also appear in G_{vw}. We need this precaution because not every element of V_v and V_w will necessarily appear in G_{vw}. Now we can express the pseudoword's ego graph node set V_{vw} as

$$V_{vw} = \alpha \cup \beta \cup \gamma \cup \delta, \tag{5.1}$$

where

$$\begin{aligned} \alpha &= V'_v \backslash V'_w \\ \beta &= V'_w \backslash V'_v \\ \gamma &= V'_v \cap V'_w \\ \delta &= V_{vw} \backslash (V'_v \cup V'_w). \end{aligned} \tag{5.2}$$

Elements in α are nodes in V_{vw} that are specific only to v, and analogously elements in β can be found only in w's distributional thesaurus and not in v's. The set γ gathers elements common to both components v and w and gives a measure of how much the two words overlap. The last set δ

contains all the terms that appear only in the combined context of v and w. It is indeed possible that such words are not significant enough to appear in v's or w's reduced distributional thesaurus, but the merge lets them gain enough importance to appear in the first 500.

As is to expect, the set δ is always of very small size. For the purposes of evaluation, we can not take its elements into consideration, as they show no immediate connection to v or w. The same can be said for the nodes in γ, which act as neutral elements and do not influence the evaluation process. However, later γ will be useful for studying the behaviour of the clustering algorithms in relation to the overlap of the pseudoword's components. In the end we will identify α and β as the desired underlying partition[7] of V_{vw}. We define the ground truth clustering as

$$\mathcal{T} = \{\alpha,\beta\}$$

and the goal of the evaluation is to compare it to the clustering $\mathcal{C}$ obtained by a given algorithm, or more precisely, to

$$\mathcal{C}' = \{C \backslash \{\gamma \cup \delta\} \mid C \in \mathcal{C}\}, \tag{5.3}$$

i.e. the clustering $\mathcal{C}$ deprived of all the neutral and unrelated elements. Figure 5.1 illustrates this situation.

We can now state the goal of pseudoword-based evaluation more precisely:

Evaluation goal 1. Given:

- two different words v and w and their ego graphs G_v and G_w,
- the pseudoword v_w resulting from their combination and its corresponding ego graph G_{vw},
- a graph-based clustering algorithm $\mathfrak{A}$,
- a measure μ for clustering comparison,

we want to assess how well the clustering $\mathfrak{A}(G_{vw}) = \mathcal{C}$ coincides with the desired ground truth clustering $\mathcal{T} = \{\alpha,\beta\}$, defined on the basis of relation (5.1), by means of the score

$$\mu(\mathcal{C}',\mathcal{T}),$$

with $\mathcal{C}'$ defined by (5.3).

[7] If $\gamma \neq \emptyset$ or $\delta \neq \emptyset$, we are actually considering a *non-exhaustive* partition or a *sub-partition*, i.e. a collection of disjoint, non-empty subsets whose union is not necessarily the whole set.

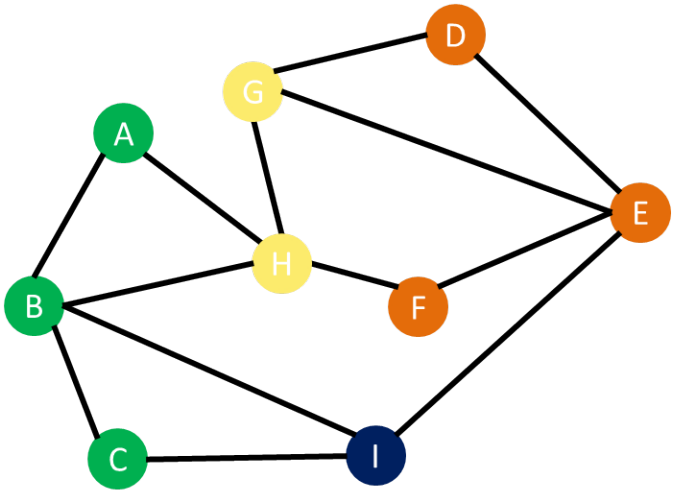


Figure 5.1: The partition of a pseudoword's ego graph, with $\alpha = \{A,B,C\}$, $\beta = \{D,E,F\}$, $\gamma = \{G,H\}$, $\delta = \{I\}$. Therefore, the ground truth clustering is $\mathcal{T} = \{\{A,B,C\},\{D,E,F\}\}$. If we had another clustering $\mathcal{C} = \{\{A,B,D,G,H\},\{C,E,F,I\}\}$, for evaluation purposes we would just consider $\mathcal{C}' = \{\{A,B,D\},\{C,E,F\}\}$.

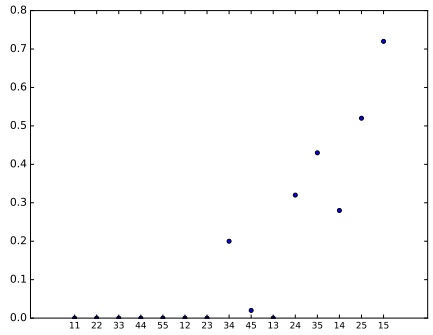
The choice of the evaluation metric μ is very delicate and can introduce a certain bias, as is discussed in Section 1.3.3 and in Section 5.4.1.1. We will not remain confined to one in particular, but instead we will use both BCubed (Section 2.2.2) and NMI (Section 2.2.1.1). Besides these two, we will also define a new, *ad hoc* measure for this evaluation setting, that we will call TOP2 and that will be explained in the next section.

5.2.2.1 Collapsed pseudowords

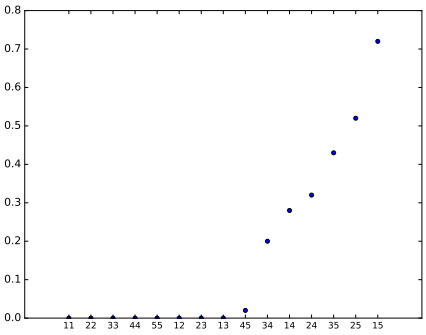
Looking back at partition (5.1) of the node set of G_{vw}, we observe that it is entirely possible that either $\alpha = \emptyset$ or $\beta = \emptyset$ holds. This means equivalently that we have either $V'_v \subseteq V'_w$ or conversely $V'_w \subseteq V'_v$. We observe this case when all the words in the distributional thesaurus of one word are also already present in the distributional thesaurus of the other one: one word totally dominates and overshadows the other one. If we assume the case $\alpha = \emptyset$, then w will be the dominant word and G_{vw} will mostly or totally collapse to a near-copy of G_w. Therefore, G_{vw} loses its interest as a potential disemous word and no real Word Sense Induction can be performed on it.

Definition 5.2.1. *We call a* collapsed pseudoword *a pseudoword v_w for which, in the notation of (5.1), either $\alpha = \emptyset$ or $\beta = \emptyset$ holds.*

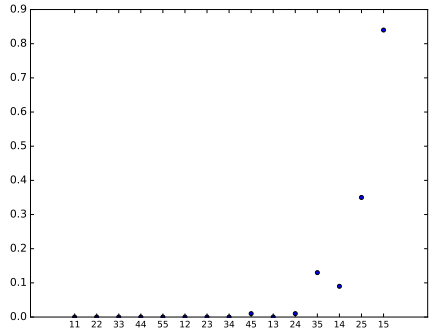
We decided to discard all collapsed pseudowords among the 1225 we constructed. This phenomenon is apparently linked to the difference in frequency classes between the components of a pseudoword (see Section 5.2.1 for our definition of frequency class). Postponing a deeper analysis of pseudowords to Section 5.3.1, we show the trend for our corpus in Figure 5.2, once for semantic-similarity-based pseudoword ego graphs and once for co-occurrence-based pseudoword ego graph, and the exact breakdown in Table 5.1 (further details about collapsed pseudowords in our data set are presented in Appendix A). Interestingly, we learn that, among all 1225 pseudowords, none collapses both for semantic similarities and co-occurrences.



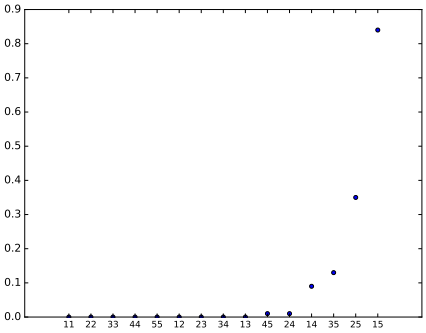
(a) Distribution of collapsed ego graphs for similarities, ordered according to frequency classes.



(b) Distribution of collapsed ego graphs for similarities, ordered according to percentages.



(c) Distribution of collapsed ego graphs for co-occurrences, ordered according to frequency classes.



(d) Distribution of collapsed ego graphs for co-occurrences, ordered according to percentages.

Figure 5.2: The distribution of collapsed pseudowords by frequency class combination.

FC combinations	Collapsed pseudowords	
	Similarities	Co-occurrences
11	-	-
22	-	-
33	-	-
44	-	-
55	-	-
12	-	-
23	-	-
34	20	-
45	2	1
13	-	-
24	32	1
35	43	13
14	28	9
25	52	35
15	72	84
Total	249	143

Table 5.1: Breakdown of collapsed pseudowords by frequency class (FC) combination and ego graph type.

Frequency classes are labelled from 1 to 5 from least to most common, and a combination of them is written in the form 12, 25, and so on. Frequency class combinations are ordered from the most balanced ones on the left, starting with components of the same class as in 11, 22 etc., up to the most unbalanced combination 15 on the right. As expected, we see that the percentage of collapsed pseudowords grows with their unbalancedness, reaching its peak just over 0.7 for semantic-similarity-based ego graphs and nearly 0.9 for co-occurrences. Symmetrically, no ego graph collapses when both components belong to the same frequency class. The rate of collapsed pseudowords also depends on the class of the most frequent component, but is overall much lower for co-occurrence-based ego graphs. We see two reasons for this behaviour. The first one is that we base the rankings in our distributional thesauri on LMI, which is itself proportional to the co-occurrence frequency of two words or of a word and a context (Section 2.2.1.3 and Section 5.2.1). If a word belongs to frequency class 5, the generic entry in its distributional thesaurus will have in average a greater LMI score than the generic entry of a word of lesser frequency. Therefore, when the contexts of the two components are merged, the most frequent word will tendentially outclass the terms specific to the less frequent word, and this effect will be the more prominent the greater the gap between frequency classes. The second reason comes from the obser-

vation that this phenomenon is slightly less evident when co-occurrences are involved; this is because co-occurrences are first-order relationships in contrast to the second-order nature of similarities computed from syntactic dependencies: for any given term w, there will be less words that share w as a neighbour than words that share the same (syntactic/semantic) context as w. We see a parallel with the discussion in Section 4.1.3, where we show that the number of edges in the finite distance graph of G is substantially always greater than the original number of edges of G.

We know that for frequency class combinations of the form 11, 22 and so on we have 45 pseudowords each, and 100 in all other cases. The numbers of Figure 5.2 mean that for our specific corpus we will be left with just around 30 or 20 meaningful pseudowords in the most unbalanced case, so that we will actually deal with around 1000 valid pseudowords in our evaluation framework. Among these ones, we foresee many skewed distributions of α and β, usually biased, as discussed before, towards the most frequent component, and we will show in Section 5.4 how they impact the algorithms' performances.

Despite pseudowords being often criticized for their supposed artificialness and for not obeying the true sense distribution of a proper polysemic word, we note that a similar skewedness is very realistic in the case of homonymy, where sense distributions tend to be dominated by a most frequent sense (MFS). In coarse-grained Word Sense Disambiguation evaluations, the MFS baseline is often in the range of 70% - 80% [Navigli et al., 2007], not unlike as for the pseudowords we present here. In view of this, we believe that the presence of collapsed pseudowords is a general behaviour, not restricted to nor arising only from our corpus. In fact, pseudoword data sets constructed from any large enough corpus will present this phenomenon, as the absolute frequency gap between more and less common terms will increase with the size of the corpus, according to Zipf's law [Zipf, 1935], leading to some terms being dominant over the others. We still include collapsed pseudowords in our data sets instead of discarding them altogether, because we deem them interesting to the end of data set analysis and, further, even if they are not suited to our homonymy detection task, they might still be used for other kinds of algorithm comparisons and evaluations.

5.2.3 The Top2 evaluation metric

When stating the Evaluation goal 1 in Section 5.2.2, we leave the evaluation measure μ undefined. Since its choice is instrumental in highlighting the strengths and weaknesses of the clustering algorithms, alongside the well-known BCubed and NMI metrics we decide to develop one that specifically takes into account the nature of our disemous pseudoword ego graphs. Basically, we exploit the fact that we know what the ground truth looks like and that it always consists of exactly two clusters.

Our intention is to penalize clusterings that stray too far from this bi-partition, rewarding however an algorithm which manages to concentrate the most significant information in two clusters, even if its clustering is otherwise rather dispersive. Our new measure is called TOP2 and is inspired by normalized mutual information in its alternative interpretation (see Section 2.2.1.1). We define TOP2 as the average of the harmonic means of homogeneity and completeness of the two clusters that better represent the two original components of the pseudoword.

We will use the setting and the notation detailed in the Evaluation goal 1 of Section 5.2.2. Let $\mathcal{C}$ denote a clustering of pseudoword v_w's ego graph obtained by algorithm $\mathfrak{A}$. We define

$$C_v = \arg\max_{C \in \mathcal{C}} |C \cap \alpha| \quad \text{and} \quad C_w = \arg\max_{C \in \mathcal{C}} |C \cap \beta| , \tag{5.4}$$

the two clusters of $\mathcal{C}$ that have the greatest overlap with v's and w's distributional thesaurus respectively; C_v and C_w are the best approximations of α and β, defined in (5.2). In a sense, we are reducing the clustering $\mathcal{C}$ to its subset $\{C_v, C_w\}$. As usual, we want arg max to yield just one element: if there is a tie, we will choose randomly.

We define the precision or purity $p_v(C)$ of a cluster C with respect to component v, and analogously $p_w(C)$ with respect to w, as

$$p_v(C) = \frac{|C \cap \alpha|}{|C|} \quad \text{and} \quad p_w(C) = \frac{|C \cap \beta|}{|C|} \tag{5.5}$$

and the recall or completeness $c_v(C)$ or $c_w(C)$ as

$$c_v(C) = \frac{|C \cap \alpha|}{|\alpha|} \quad \text{and} \quad c_w(C) = \frac{|C \cap \beta|}{|\beta|}. \tag{5.6}$$

We notice that the (5.6) are always well defined, since we chose to discard collapsed pseudowords (see Definition 5.2.1) and therefore we always have $|\alpha| , |\beta| \neq 0$. Additionally, by definition (5.4) these values will be always greater than 0.

For both clusters C_v and C_w, we compute the harmonic mean of purity and completeness with respect to the component they are representing. We write the harmonic mean of two positive real numbers $x,y \in \mathbb{R}^+$ as $h(x,y)$ and remember that

$$h(x,y) = \frac{2}{\frac{1}{x} + \frac{1}{y}}.$$

So, given $h\left(p_v(C_v),c_v(C_v)\right)$ and $h\left(p_w(C_w),c_w(C_w)\right)$ we define the TOP2 score as their macroaverage

$$\text{TOP2}(C_v,C_w) = \frac{h\left(p_v(C_v),c_v(C_v)\right) + h\left(p_w(C_w),c_w(C_w)\right)}{2}. \tag{5.7}$$

Notice that in (5.7) the first cluster is always valued with respect to v, and the second one with respect to w.

We notice that following the definition of representative clusters of (5.4) can lead to the unfortunate case $C_v = C_w$, where the algorithm concentrates and confuses all the information about the two distinct senses of the pseudoword in a single cluster; we denote this catch-all cluster as C_{vw}. However, we want to penalize this behaviour: as $\alpha \cap \beta = \emptyset$, we also want to impose $C_v \cap C_w = \emptyset$ so as to have comparable clusterings $\{\alpha,\beta\}$ on one side and $\{C_v,C_w\}$ on the other. To this end we also consider the second best matches for (5.4), which we denote respectively as C'_v and C'_w. More precisely, C'_v meets the conditions

$$|C \cap \alpha| \leq \left|C'_v \cap \alpha\right| \leq |C_v \cap \alpha|$$

for any $C \in \mathcal{C}$, $C \neq C_v$. The conditions for C'_w are expressed analogously. Now we have two possible reduced clusterings to consider, namely either $\{C'_v,C_{vw}\}$ or $\{C_{vw},C'_w\}$. The final TOP2 score is then defined as the better of these two cases:

$$\text{TOP2}(C_{vw}) = \max\left(\text{TOP2}(C'_v,C_w),\text{TOP2}(C_v,C'_w)\right). \tag{5.8}$$

If C'_v or C'_w holds just a fraction of the nodes relative to v or w, both its purity and completeness will suffer and the macroaverage will be lower than computing TOP2 just for C_{vw}.

The other case to take into account is when the algorithm produces only one cluster, i.e. when $\mathcal{C} = \{C\}$. As we can not resort to second best representative clusters, we look at the clustering $\{C,\emptyset\}$ and put the harmonic means $h\left(p_v(\emptyset),c_v(\emptyset)\right)$ and $h\left(p_w(\emptyset),c_w(\emptyset)\right)$ equal to 0. We associate C to the component it best represents: to v if

$$|C \cap \alpha| > |C \cap \beta|$$

and to w if

$$|C \cap \beta| > |C \cap \alpha|\,.$$

If there is a tie, we will choose randomly. According to this choice, we will compute either

$$\text{TOP2}(C,\emptyset) = \frac{h\left(c_v(C),p_v(C)\right)}{2}$$

or

$$\text{TOP2}(\emptyset,C) = \frac{h\left(c_w(C),p_w(C)\right)}{2}.$$

Thus, even if the overlap between C and e.g. α were nearly perfect and we had

$$h\left(c_v(C),p_v(C)\right) \simeq 1$$

(precision can not be one as $\beta \neq \emptyset$ and $\beta \subset C$), the TOP2 score would never be higher than $\frac{1}{2}$. This means that monocluster clusterings are characterized by scores not greater than 0.5.

The TOP2 score is an *ad hoc* evaluation metric that penalizes two kinds of clusterings in particular: those that are very fragmentary and that evenly distribute elements among their clusters on one hand, and those with a very skewed distribution, which tend to group all elements in one huge cluster, on the other hand, up to the extreme of the trivial clustering (see the final discussion of Section 5.4.4). They will both get low scores: the former because of the generally low recall of its clusters, despite an expectedly higher precision, and the latter because even if one representative cluster performs very well in terms of precision and recall, the other one will possess much less significant information, and their average will not be optimal. As shown in the previous paragraph, the extremely unbalanced trivial clustering is guaranteed to obtain a score lower than 0.5, intuitively representing that only half of the objective has been achieved. A perfect score of 1 is reachable by a clustering $\mathcal{C}$ if and only if $|\mathcal{C}| = 2$ and it exists a refinement of $\mathcal{C}$ containing $\mathcal{T} = \{\alpha,\beta\}$ as a subset. Expressly basing the score TOP2 on the two best clusters of $\mathcal{C}$ assures that we can avoid the proved or alleged bias towards clustering granularity present for other evaluation systems like mutual information variants or BCubed, as discussed in Section 5.4.1.1.

5.2.3.1 Generalization of TOP2

In an evaluation framework where different known degrees of homonymy or polysemy are known, the TOP2 evaluation metric (5.7) can be generalized to a TOPN score, with $N \geq 2$. In this case, we will have N components $v_1,v_2,\ldots,v_N$ of a pseudoword, and retrieving the best representative cluster for each component follows analogous definitions as (5.4), and the

final score is again the macroaverage of harmonic means. Cases of the form $C_{v_i} = C_{v_j}$ can also be solved analogously, even if this might involve a comparison between (much) more than two alternatives, as was for (5.8), and in the worst cases the need to retrieve a third or even further less best choice for the representative cluster. The constant trait of TOPN remains that an incomplete clustering will be penalized in the sense that, if just $n < N$ clusters are found, its score will never be greater than $\frac{n}{N} < 1$, giving an immediate measure of the clustering's shortcomings. This is again achieved setting the harmonic means of missing clusters equal to 0.

5.3 Evaluation implementation, techniques and data set analysis

In this section we will present results pertaining to our two pseudoword data sets. We recall the scheme of the Evaluation goal 1 in Section 5.2.2 and first of all introduce the ingredients of our evaluation framework.

Stemming from the corpus described in Section 5.2.1, we have 50 monosemous words evenly distributed among 5 frequency classes that will serve as components[8]

Frequency class 1**:** *ackee, barque, bobolink, bufflehead, carryall, euonymus, leatherette, pennywhistle, pneumococcus, tautog*

Frequency class 2**:** *afro, barman, catsup, dimwit, grosgrain, hyperthyroidism, philanderer, southerly, tranquilliser, yellowcake*

[8]Here we give a short definition for each of the more uncommon terms we use in our evaluation. They are taken from the free online lexicon *The Free Dictionary*, available at `www.thefreedictionary.com`.

ackee: A tropical African tree and its fleshy fruit
barque: Variant of *bark*; a boat or sailing ship
bobolink: An American migratory songbird
bufflehead: A small North American diving duck
carryall: A large receptacle used to carry things from one place to another
catsup: Variant of ketchup
dimwit: A stupid person
euonymus: A genus of trees or shrubs
grosgrain: A heavy ribbed silk or similar for trimming clothes
leatherette: Imitation leather made from paper or cloth
pennywhistle: An inexpensive fipple flute of cheap materials
philanderer: A man who has short sexual relationships with many women; womanizer
southerly: A storm or wind coming from the south
tailwind: A wind blowing in the same direction as the course of an aircraft or ship
tautog: A dark-coloured, edible fish found along the North American Atlantic coast
wedlock: The state of being married
yellowcake: Semirefined uranium ore

Frequency class 3**:** *astrologer, bullfight, curio, flamboyance, hairpiece, lightbulb, showtime, tailwind, ugandan, wedlock*

Frequency class 4**:** *artistry, heartache, heartbreak, marshmallow, pillow, reliability, reptile, shawl, tequila, tofu*

Frequency class 5**:** *ailment, beer, clothing, courage, credibility, dioxide, garment, paycheck, psychologist, treasurer*

As mentioned in Section 5.2.1, we select these components by first checking if their ego graphs are partitioned into just one single cluster. To do this, we use a fine-grained version[9] of Chinese Whispers (Section 3.3). We further verify that also WordNet[10] lists only one sense for the selected terms. Inspecting other lexical resources like the Oxford English Dictionary [Stevenson, 2010], even in its online edition[11], or *The Free Dictionary*[12], confirms the monosemous nature of the chosen components. Combined, they give rise to 1225 pseudowords. Looking at Table 5.1, we have to discard a total of 249 collapsed pseudowords for semantic-similarity-based ego graphs and 143 for co-occurrence-based ego graphs. This brings the actual count of pseudowords used in our evaluation down to respectively 976 and 1082 pseudowords. A further analysis about collapsed pseudowords is reported in Appendix A.

We will evaluate and compare the performances of the following clustering algorithms (in parentheses the acronyms by which we will refer to them):

- Markov cluster algorithm (MCL; Section 3.1)
  - We use the parameter 2 for the number of expansion steps, but run the algorithm both with 1.4 and with 2 as the inflation parameter. Lower values yield less fragmented clusterings. We iterate the process up to 100 times to assure convergence.
- Chinese Whispers (CW; Section 3.3)
  - Our version of the algorithm is the most basic one, without parameters, described in [Biemann, 2006].
- MaxMax (MM; Section 3.5)
  - We implement the version described in [Hope and Keller, 2013].

[9] In particular the implementation found in `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/documentation/sense-clustering/` with parameters: `-n 200 -N 200`.

[10] `http://wordnetweb.princeton.edu/perl/webwn`; [Miller, 1995]

[11] `https://en.oxforddictionaries.com/`

[12] `www.thefreedictionary.com`

- Gangplank clustering algorithm (GP; Sections 4.2.1 and 4.3.2)
  - We orient ourselves on the basic version described in [Cecchini and Fersini, 2015]. Further, we set the restriction that the algorithm does not cluster complete connected components of the ego graph. We also implement the recompacting step for singletons. In Section 5.4.3 we will briefly consider a variant using a minimum cut (see Section 2.1.3 and Section 4.3.2.1).
- Aggregative clustering algorithm (AGG; Sections 4.2.2 and 4.3.3)
  - We focus on the value 0.97 for radius σ, which yielded good results for the tweet data set used in [Cecchini et al., 2015], but also have a run which implements a variable radius using the 75th percentile of the distance set Δ. We also implement the recompacting step for singletons.
- Curvature-based clustering algorithm (CURV; Sections 4.2.3 and 4.3.4)
  - We base the curvature on the Jaccard distance expounded in Section 4.1.1. We also implement the recompacting step for singletons.

We also introduce a secondary level of clustering that we call *hyperclustering*, that can be applied to any of the previous algorithms and is useful for recompacting fragmented clusterings. It is detailed in Section 5.3.2. A one-cluster-per-word baseline (BSL), grouping all nodes of a word graph into a single cluster, will also be used as a benchmark.

The evaluation metrics of which we will make use are:

- Normalized mutual information (NMI, Section 2.2.1.1)
- BCubed (Section 2.2.2)
- TOP2 (Section 5.2.3)

Results will be given in Section 5.4 with respect to each clustering algorithm and to parameters that summarize the structure of the pseudowords, as will be explained in 5.3.1, and there will also be a further in-depth discussion of results in Appendix B.

5.3.1 Pseudoword analysis

In this section we will get more insight into the pseudowords that we compose from our corpus and with which our clustering algorithms will have to deal. In Section 5.2.2.1 we present data about the issue of collapsed pseudowords. Since they are extreme, unmeaningful cases that we will not

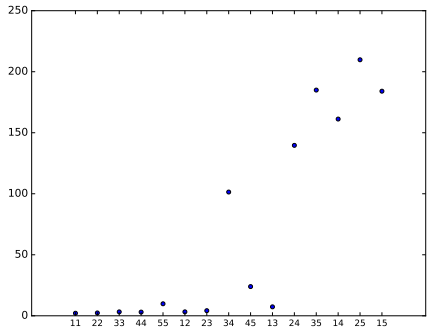
use in our evaluation, the following statistics do not consider them. We will use three main parameters deriving from the node set representation of (5.1) and compare semantic-similarity-based to co-occurrence-based ego graphs. We will denote a generic pseudoword with v_w and its ego graph with G_{vw}.

The first aspect that we will examine is the *balance*, or conversely the *skewedness*, of a pseudoword. We deem it to be represented by the ratio

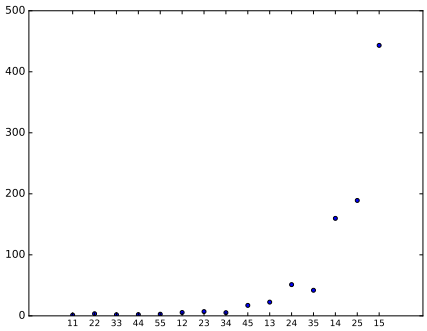
$$\rho = \max\left(\frac{|\beta|}{|\alpha|}, \frac{|\alpha|}{|\beta|}\right).$$

We notice that ρ is always greater than 1. In other words, we consider all nodes that specifically belong to just one of the two components of v_w, i.e. the elements contained in the sets α or β, and measure the ratio of the most represented component to the least represented one (this ratio is well defined because we excluded the case of collapsed pseudowords of Definition 5.2.1). If a pseudoword's ego graph is perfectly balanced, $\frac{|\beta|}{|\alpha|} = \frac{|\alpha|}{|\beta|} = 1$. Otherwise, a value ρ will tell us that for every word of the smaller component G_{vw} contains ρ of the larger one. We take the mean of these ratios for each of the 15 different frequency classes, and show the results in Figure 5.3: once for semantic similarities and once for co-occurrences, first ordered from less to more unbalanced frequency class (e.g.,14 is more unbalanced than 22) and then reordered from smaller to greater values of the ratio.

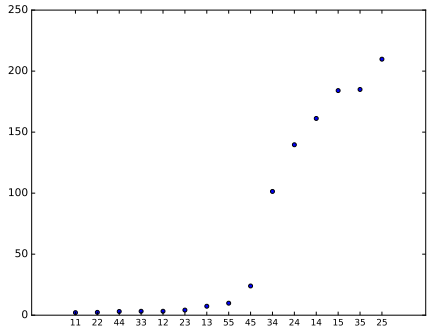
We evidence the general, expected trend to have more unbalanced words when quite different frequency classes (FC) get mixed. Conversely, combinations of components in the same FC are mostly very balanced. The dominance of some terms in FC 5 makes 55 stick out as the more unbalanced of the most balanced combinations. In any case, the situation for semantic similarities and co-occurrences is nearly the same: In both cases the combinations 25, 15 and 14 are among the most unbalanced ones, with 25 and 15 reaching the respective maxima. The latter combination, despite having been deprived of many collapsed pseudowords, still has extremely skewed pseudowords. We notice that already a ratio of 100, in a network of around 500 elements (also counting sets γ and δ for simplicity), implies that the weakest component is represented by about 5 nodes, and a ratio of 2 already means that two thirds of the network belong to the strongest component. So, more than half of our pseudowords are quite unbalanced towards one pseudosense. When this happens, our starting assumption is that a clustering algorithm will be biased towards the dominant term: in



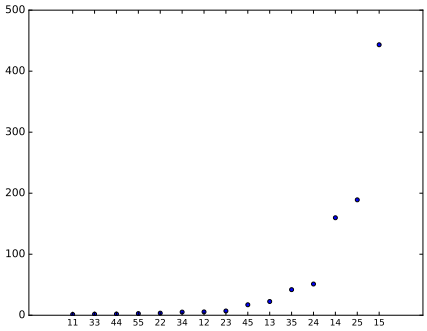
(a) Ratio in similarity ego graphs.



(b) Ratio in co-occurrence ego graphs.



(c) Ratio in similarity ego graphs, ordered from lowest to highest.



(d) Ratio in co-occurrence ego graphs, ordered from lowest to highest.

Figure 5.3: The mean ratio ρ of the most represented component to the least represented component (either $|\alpha|$ / $|\beta|$ or $|\beta|$ / $|\alpha|$; y-axis) in similarity and co-occurrence pseudoword ego graphs, by frequency class combination (x-axis).

practice, we expect it to find tendentially only one big, or even just a single cluster. We recall that we define the TOP2 score in Section 5.2.3 especially to penalize such eventualities, whereas in Section 5.4.1.1 we will see that BCubed seems to be insensible to such skewedness.

We note that co-occurrence-based ego graphs seem to be generally slightly more balanced than the equivalent semantic-similarity-based ones, but also that the gap between the two more unbalanced combinations and the remaining ones is more pronounced than for semantic similarities. For semantic similarities, there are 6 FC combinations (34, 24, 14, 15, 35, 25) with a score over 100, but for co-occurrences there are only 3 (14, 25, 15). However, in the latter case they reach much higher ratios than for semantic similarities. We explain this fact again (cf. Section 5.2.2.1) remarking that co-occurrences are sparser than syntactic relationships, and thus that

LMI scores (see Section 2.2.1.3 and Section 5.2.1) tend to be more evenly distributed and components have a lesser tendency to predominate one over the other. The extreme value of class 15 for co-occurrences is not statistically significative, as it is based on only few (16) pseudowords (see Table 5.1), while the scores for other classes in Figure 5.3d are in line with those seen in Figure 5.3c. The jumps in the higher values on the rightmost side of Figures 5.3c and 5.3d come from the computation of ratio $\frac{|\beta|}{|\alpha|}$ or $\frac{|\alpha|}{|\beta|}$ on a finite, discrete node set.

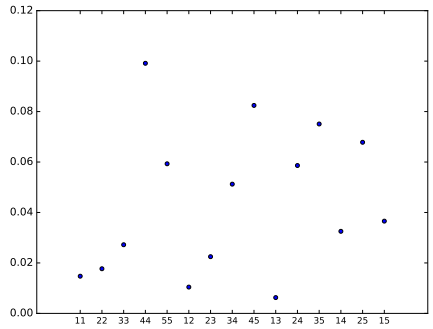
Figure 5.4 shows the mean cardinality of the node set γ per frequency class combination and relative to the total size of the ego graph. More precisely, given a network $G = (V,E)$, we consider the ratio

$$\kappa = \frac{|\gamma|}{|V|}.$$

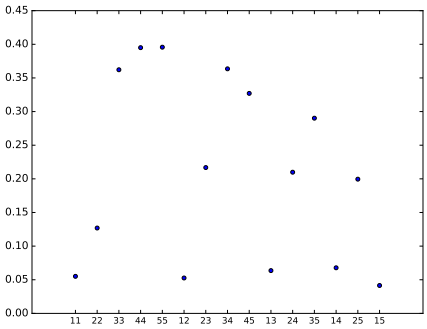
The set γ, in the notation of (5.1), consists of the terms that are common to the distributional thesauri of v and w and its size measures the *overlap* of v and w. The more similar the meanings of the two components, the larger we expect the overlap to be. In the case of the pseudoword *beer_tequila*, for example, 305 nodes, more than half of the total 492 nodes in its semantic-similarity-based ego graph, are found in the distributional thesauri of both *beer* and *tequila*, for a κ score of 0.62; the same number rises to 332 on 484 for co-occurrences (and the pseudoword collapses, so that the resulting score $\kappa = 0.69$ has not been considered for the statistics shown). In Figures 5.4a and 5.4b a pattern is clearly visible: the higher the mean frequency class and the greater the smaller member of the frequency class combination, the more terms the two components have in common (with the exception of 44). Figures 5.4c and 5.4d show a roughly linear increase.

Again, as for ρ, co-occurrence ego graphs present higher values, due to the fact that syntactical dependence relations are more specific than mere co-occurrences. Two terms might share many common contexts even if they fulfill different semantic roles: co-occurrences imply a generally less strict bond than semantic similarities.

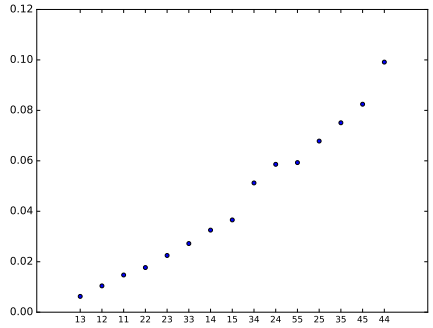
The generic trend of higher-frequency combinations to have a relatively bigger γ is directly related to the frequency of the single components. With a simplistic reasoning, the more frequent a term, the richer its distributional thesaurus, and the richer two distributional thesauri, the more probable that their elements will overlap at a higher degree. Very unbalanced words are likely to have relatively smaller γ's: the unrelatedness of the two components is both the cause of the prevalence of the most frequent one and of a negligible overlap.



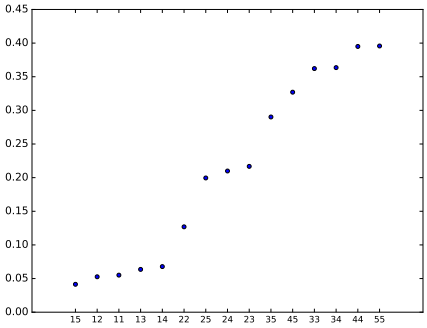
(a) Relative size of γ in similarity ego graphs.



(b) Relative size of γ in co-occurrence ego graphs.



(c) Relative size of γ in similarity ego graphs, ordered from lowest to highest.

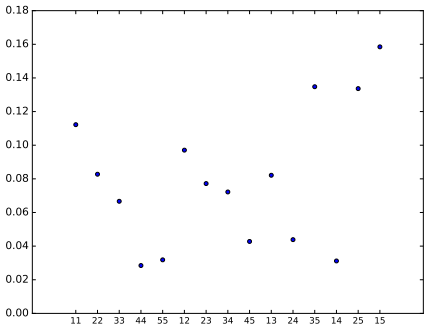


(d) Relative size of γ in co-occurrence ego graphs, ordered from lowest to highest.

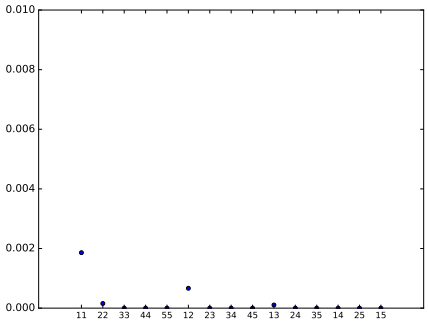
Figure 5.4: Mean relative overlap κ given by the number of nodes common to both components $|\gamma|$ with respect to the total number of nodes (y-axis), by frequency class combination (x-axis).

Finally, we show how the node set δ behaves relatively to the total size of G_{vw} in Figure 5.5, i.e. the ratio $\frac{|\delta|}{|V|}$. The set δ represents the *unknown* terms in the distributional thesaurus of v_w, i.e. those that appear neither in the reduced distributional thesaurus of v nor in that of w, and that we can not therefore relate to either. The ratios are overall smaller, but comparable to those obtained with κ for γ.

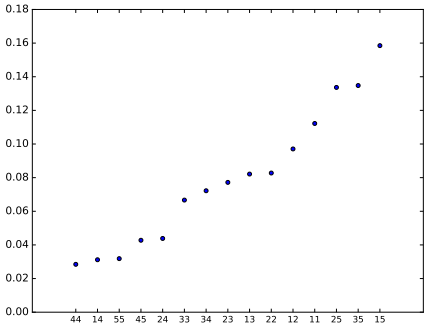
No particular rationale appears from the graphs, apart from a slight negative correlation to the graphs of γ: word combinations with many overlapping elements apparently let fewer unrelated terms slip in. It looks sensible that a greater overlap between two components is mirrored by a network with many common terms, whereas two unrelated components may lead to some stray term to be highlighted and have a combined LMI score that allows it to appear in G_{vw}. If γ represents relatedness, we might



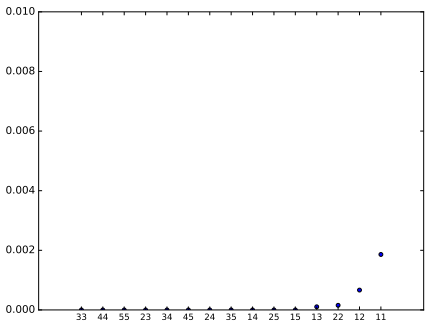
(a) Relative size of δ in similarity ego graphs.



(b) Relative size of δ in co-occurrence ego graphs.



(c) Relative size of δ in similarity ego graphs, ordered from lowest to highest.



(d) Relative size of δ in co-occurrence ego graphs, ordered from lowest to highest.

Figure 5.5: Mean ratios of nodes $|\delta|$, appearing only in the pseudoword v_w's ego graph but in neither of the distributional thesauri of v or w, relative to the total number of nodes (y-axis), by frequency class combination (x-axis).

consider δ as an indication of unrelatedness, with some caution. This is evident for semantic similarities, and even for co-occurrences, although in the latter case this phenomenon is nearly non-existent. Co-occurrence-based networks seem to be more stable: as already stated, more words share more contexts with more evenly distributed frequencies, so that it is particularly improbable that an external term gains a greater score than another term already present in v's or w's distributional thesaurus. Again: Co-occurrences imply a generally less strict bond than syntactical dependencies.

The analysis we conducted in Section 5.2.2.1 and in this section on collapsed and regular pseudowords, based on the decomposition of the node set V of the ego graphs as in (5.1), reveal that our pseudoword data set represents a good variety of artificial words that simulate real homonymy. As

expected, we evince that there are two main aspects which shape a pseudoword's structure: the interaction between different frequency classes, represented by the balance parameter $\rho = \max\left(\frac{|\alpha|}{|\beta|}, \frac{|\beta|}{|\alpha|}\right)$, and the correlation of its two components, represented by the relative overlap $\kappa = \frac{|\gamma|}{|V|}$. We can visualize these two parameters as the axes of a graph that summarizes the salient characteristics of each pseudoword, in both cases of semantic-similarity and of co-occurrence-based ego graphs, shown in Figure 5.6.

The nearer to the origin $(0,0)$, the more desirable the structure of a pseudoword: both components are balanced and the overlap is small, so that it is probably easier to distinguish and separate the two pseudosenses.

5.3.1.1 Examples of parameter computations

We will exemplify the computations of the pseudoword parameters seen in the previous section with two very different pseudowords: the already mentioned *beer_tequila* and *barque_pennywhistle*. The word *beer* is of frequency class 5, *tequila* of frequency class 4, and both *barque* and *pennywhistle* belong to frequency class 1. For the first couple we expect a large overlap, while the second couple looks unrelated. We will first examine their parameters in the semantic-similarity-based data set.

The pseudoword *beer_tequila* is unbalanced in favor of *beer*: the set α contains 139 words like

{`Heineken, tofu, sirloin, soft-drink, fajita,...`},

mostly about beer brands, food and beverages, while β, for *tequila*, has only 15, with a more restricted range:

{`Beam, vermouth, Burgundy, Sauvignon,...`}.

The balance parameter is then

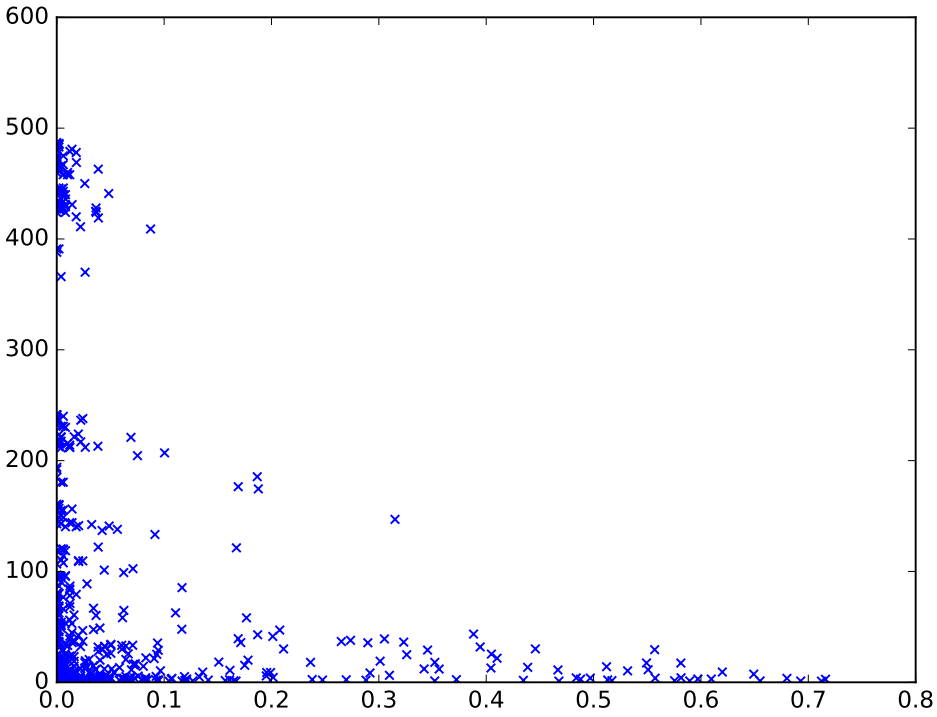
$$\rho = \max\left(\frac{139}{15}, \frac{15}{139}\right) = \max(0.11, 9.27) = 9.27,$$

very far from 1 and demonstrating the skewness of this pseudoword.

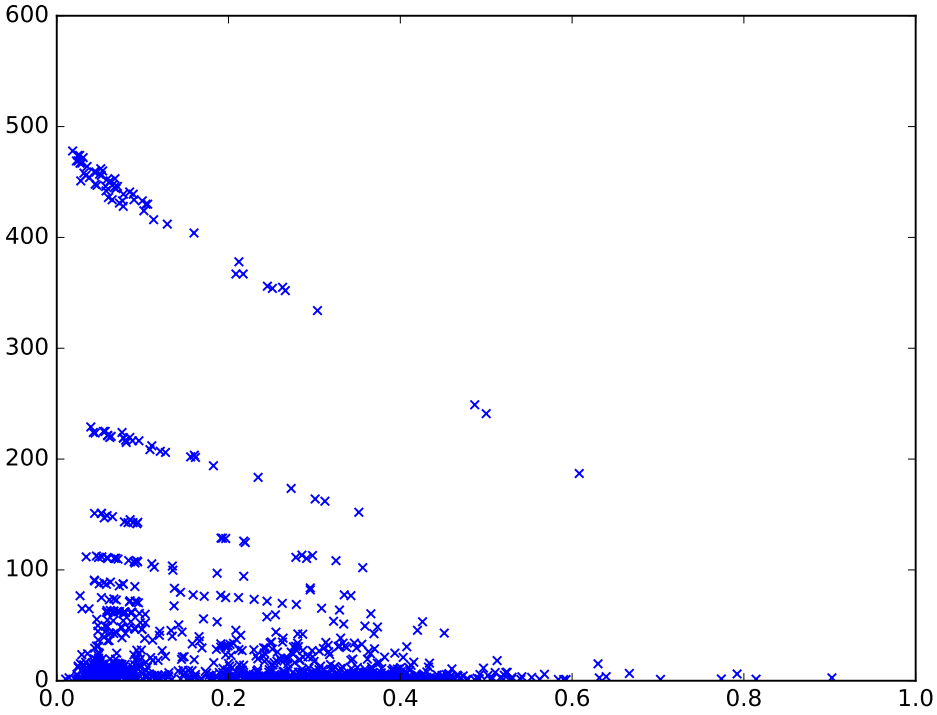
As already mentioned in Section 5.3.1, the set γ is very large here, with cardinality 305, for a high relative overlap $\kappa = \frac{305}{492} \cong 0.62$. The set γ contains terms such as

{`broth, tapa, syrup, roast, lobster, bourbon,...`}.

The set δ of new words has 33 elements, including terms not directly related to food or beverages, like



(a) Word distribution for similarity-based ego graphs.



(b) Word distribution for co-occurrence-based ego graphs.

Figure 5.6: Comparison of relative overlap κ (x-axis) against balance ratio ρ (y-axis) for each pseudoword in both cases of similarity and co-occurrence-based ego graphs.

{detergent, petroleum, shampoo, julep,...}.

The behaviour of this pseudoword changes in the co-occurrence-based data set, where the high frequency class of *beer* and the very similar senses of the two components make it collapse. Out of 485 nodes, the common terms in γ are 332, more than for the semantic-similarity-based ego graph, for an overlap of $\kappa = 0.68$, but $|\beta| = 0$ and $|\delta| = 0$, so that this ego graph is a near-replica of the first-order distributional thesaurus of *beer*. We notice that the terms in α and the common terms in γ approximately repeat the same themes as in the semantic-similarity-based ego graph, as respectively in

{Oktoberfest, Schlitz, license, belgian-style,...}

and

{produce, gallon, drunk, distilled, go, lunch,...}.

The pseudoword *barque_pennywhistle* is more balanced than *beer_tequila* in the semantic-similarity-based data set, a fact due to the same, low frequency class of both components: we have 37 words for barque, among which

{ship, yacht, plane, Fock, sailboat, galleon,...},

and 96 for pennywhistle, among which

{kazoo, organ, horn, tuba, concertina, solo,...},

for a balance parameter $\rho = 2.60$. There is no overlap: the set γ of words common to both distributional thesauri is empty, so that $\kappa = 0$, as could be expected from words referring to two such dissimilar objects. Still, we have 11 terms in δ. Terms like *bell* or the imitative *twang* might be put in relation both with the nautical sphere (the ship's bell, the sound of a tight rope) and with sounds and musical instruments. Other terms like *poetry* are more difficult to interpret.

Passing to co-occurrences does not change the picture much. The ego graph is even more balanced now: 109 nodes for *barque* against 108 for *pennywhistle*, obtaining $\rho = 1.01$. For *barque* we have e.g.

{participate, 179-foot, brig, aircraft, Sodano,...},

and for *pennywhistle*

{musical, jazz, most, drum, pipe, balladeer,...}.

We note two things. First, the word *most*, normally considered a stop word (see Sections 2.1.8 and 4.3.1), testifies the more noisy nature of co-occurrence-based ego graphs (we also recall from Section 5.2.1 that our co-occurrence ego graphs can be composed of any kinds of word classes); second, the nature of co-occurrences is shown by the apparently out-of-place *Sodano*, the name of a former Cardinal Secretary State of the Vatican, which appears because of the *barque of St. Peter*, a metaphor for the Catholic Church.

The remaining 6 nodes of the few total 223 nodes (due to the low frequency classes of both components) of the network belong to γ, for a vanishing relative overlap of $\kappa = 0.03$, in line with $\kappa = 0$ for semantic-similarity-based ego graphs. The common words are

$$\{\texttt{same, way, time, own, more, include}\},$$

which can be considered stop words (e.g. *same*) or very generic ones (e.g. *time*).

5.3.2 Hyperclustering

Each clustering algorithm has its own position on the scale of granularity: some tend to produce fine-grained clusterings with many clusters, independently from the structure of the clustered set, while others tend to coarse-grained clusterings with few clusters. The spectrum of granularity on a set V has two natural extremes: on one side the partition $\{V\}$, where no distinctions are made, and on the other side the partition $\{v_1,v_2,\ldots,v_{|V|}\}$, where each element of V constitutes its own cluster. If a clustering is strictly defined as a partition, all possible clusterings of V are represented by a lattice, with the partial ordering given by partition refinement (see Section 2.1.9.1). We denote this lattice with $\mathcal{C}(V)$ and notice that it behaves quite differently from the power set $\mathcal{P}(V)$ of all subsets of V. We can see $\mathcal{C}(V)$ as a hierarchy, with its supremum $\{V\}$ at its top and the infimum $\{v_1,v_2,\ldots,v_{|V|}\}$ at its bottom.

The aim of hyperclustering is to define a process by which the same algorithm that produced a clustering $\mathcal{C}$ reuses it to build a sequence of coarser clusterings. Starting from $\mathcal{C}$, if $\leq$ represents the ordering by refinement, we want to define an operator Hyp such that

$$\mathcal{C} \leq \mathrm{Hyp}(\mathcal{C}).$$

The iterated application of Hyp will move the clustering upwards in the hierarchy of $\mathcal{C}(V)$, up to a maximal element that may or may not be the supremum. We want Hyp to recompact $\mathcal{C}$, preserving its clusters as building blocks that are combined to form new, bigger clusters. This way, we

can exploit the natural higher precision of smaller clusters to improve the overall recall of $\mathcal{C}$ without losing information about the already found associations. With what may resemble a pun, the operator Hyp "clusters the clusters of a clustering".

To perform a clustering on $\mathcal{C} = \{C_1,C_2,\ldots,C_n\}$ we first have to represent the clusters C_i as nodes of a graph and to define edges connecting them. We will call the resulting graph the *hypergraph*[13] $H = (W,F)$ of $\mathcal{C}$. For the node set we simply define $W = \mathcal{C}$. The edges and their weights depend on the structure of the graph $G = (V,E)$ of which $\mathcal{C}$ is a clustering. We set

$$(C_i,C_j) \in F \quad \Leftrightarrow \quad \exists (v,w) \in E \quad \text{s.t. } v \in C_i, w \in C_j.$$

In other words, two nodes of H are linked if there exists an edge in G linking any two elements of their two corresponding node subsets. This condition is by itself quite weak, especially if we think about the distance graphs defined in Section 4.1.3: A word graph over a large corpus will have many connections, so that H will most probably be very dense too. Therefore, we want to give a weighted structure to H. To do this, we consider all edges connecting C_i to C_j (we could see it as taking the bipartite subgraph of G induced by C_i and C_j) and take their mean weight. Formally, we define the interconnecting edge set

$$I(C_i,C_j) = \{(v,w) \in E \mid v \in C_i, w \in C_j\}, \tag{5.9}$$

and subsequently the weight mapping

$$\mu(C_i,C_j) = \frac{1}{\left|I(C_i,C_j)\right|} \sum_{(v,w)\in I(C_i,C_j)} p(v,w),$$

where $p : E \times E \longrightarrow \mathbb{R}^+$ is the weight mapping associated to G. The new weight mapping of H

$$\mu : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{R}^+$$

preserves weights either as similarities or distances, if they are respectively similarities or distances for G.

We express the clustering $\mathcal{C}$ over $G = (V,E)$ (more precisely over its node set V) obtained through algorithm $\mathfrak{A}$ as $\mathfrak{A}(G)$. We also denote the hypergraph relative to $\mathcal{C}$ as $H(\mathcal{C})$ to put in evidence the clustering from which it originates. Then, we define the hyperclustering of $\mathcal{C}$ as

$$\mathrm{Hyp}(\mathcal{C}) = \mathfrak{A}(H(\mathcal{C})) = \mathfrak{A}(H(\mathfrak{A}(G))). \tag{5.10}$$

[13]Despite some similarities, our definition of hypergraph is different than the common graph-theoretical concept that goes by the same name, namely that of a graph $G = (V,E)$ whose edges can be generic subsets of V. See [Berge and Minieka, 1973] for more details about the subject.

The operator $\mathrm{Hyp}(\cdot) = \mathfrak{A}(H(\cdot))$ defines the hyperclustering $\mathrm{Hyp}(\mathcal{C}) = \{Q_1,\ldots,Q_m\}$, which directly induces a new clustering over G. Namely, the sets defined as

$$K_i = \bigcup_{C \in Q_i} C \quad \forall Q_i \in \mathrm{Hyp}(\mathcal{C})$$

form a new partition $\mathcal{K} = \{K_1,\ldots,K_m\}$ of V, which is by definition a coarsening of $\mathcal{C}$. To ease notation, we identify $\mathcal{K}$ with $\mathrm{Hyp}(\mathcal{C})$, and can therefore write $\mathcal{C} \leq \mathrm{Hyp}(\mathcal{C})$.

Hyperclustering is a recursive process: we can use the same clustering algorithm to cluster over and over the hyperclustering that it obtained in the previous step. Of course, we could write the rightmost side of (5.10) as

$$\mathfrak{B}(H(\mathfrak{A}(G))),$$

i.e. using an algorithm different than the first one to perform the hyperclustering step. However, sticking to the same one guarantees the coherence of the associations made at a higher level with respect to the initial ones. Since we do not have to stop at the first level of hyperclustering, we can write

$$\mathrm{Hyp}^2(\mathcal{C}) = \mathrm{Hyp}(\mathrm{Hyp}(\mathcal{C})) = \mathfrak{A}(H(\mathfrak{A}(H(\mathfrak{A}(G)))))$$

and more generally

$$\begin{aligned} \mathrm{Hyp}^{n+1}(\mathcal{C}) &= \mathrm{Hyp}(\mathrm{Hyp}^n(\mathcal{C})) \\ \mathrm{Hyp}^0(\mathcal{C}) &= \mathcal{C} \end{aligned}$$

for any $n \in \mathbb{N}$. We have naturally

$$\mathrm{Hyp}^{n+1}(\mathcal{C}) \geq \mathrm{Hyp}^n(\mathcal{C}).$$

The sequence

$$\{\mathrm{Hyp}^1(\mathcal{C}), \mathrm{Hyp}^2(\mathcal{C}), \ldots\}$$

has a limit. The coarsest graph-based clustering over G obtainable by iterating Hyp over G coincides with the set of connected components of G. We note that the hypergraph of a connected component of G is itself connected, and that the number of connected components of H (see Section 2.1.2) is the same as G.

Indeed, let $G' = (V',E')$ be a connected component of G. If

$$\mathcal{C} = \{C_1,C_2,\ldots,C_n\}$$

is a partition of V', if $n = 1$ the hypergraph will have just one node and be trivially connected. If $n > 1$, for any cluster C_i there will exist a node $w \notin C_i$. For any two nodes $v \in C_i$ and $w \in C_j$, $i \neq j$, by definition of connectedness (see Section 2.1.2) there exists a path $u_0 = v, u_1, \ldots, u_t, u_{t+1} = w$ of length t between v and w. Each u_k lies in a given cluster, and the path traverses at least two different clusters, C_i and C_j. If k is such that $u_k \in C_r$ and $u_{k+1} \in C_s$ with $r \neq s$, then $I(C_r, C_s)$ contains (u_k, u_{k+1}) and $H(\mathcal{C})$ has the edge (C_r, C_s). Repeating this for every such couple u_k, u_{k+1} in the path, we find that $H(\mathcal{C})$ also contains a path from C_i to C_j. Repeating this for any couple of clusters, we proved that $H(\mathcal{C})$ is also connected.

5.3.2.1 Hypergraphs with two nodes

If a clustering has only two clusters, its hypergraph will have two nodes. This case is rather anomalous. As discussed in the previous Section 5.3.2, if and only if both nodes belong to the same connected component the hypergraph will also have an edge between them. If this is the case, we have two choices: either merge the two clusters or leave them separated. This choice is not particularly significative, as their merge is the supremum of the partition lattice, a trivial clustering. With only two nodes and one edge, the algorithms CW and MM will find one cluster, while MCL will oscillate. Instead, the three algorithms GP, AGG and CURV that we introduce respectively in Sections 4.2.1, 4.2.2 and 4.2.3 all make use of weighted Jaccard distance (Section 4.1.1). When they are iterated on such a hypergraph, the Jaccard distance between the two nodes will forcibly be 0. Respectively for each algorithm, the edge will be counted as a gangplank and GP will keep the nodes separated; any radius σ greater than 0 will make AGG merge the two nodes, and if we are using a percentile to determine σ, since 0 is not a valid value we can arbitrarily set a greater one and get the same result; for CURV there will be a null curvature and the two nodes will be kept separated.

As hyperclusters are already condensed clusters, we will not perform any recompacting step (as described e.g. by Algorithm 3) during hyperclustering.

5.4 Results

In the following section we will present and compare the main results on the two pseudoword ego graph data sets, in relation to the algorithms and the evaluation measures laid out in Section 5.3 and to the hyperclustering detailed in Section 5.3.2. We will first sum up all the mean scores

for each frequency class combination in the double Tables from 5.5 to 5.8, along with the mean number of clusters in Tables 5.9 and 5.10, where the highest scores are always highlighted. All scores, apart from the numbers of clusters, are shown in percentages, ranging from a minimum of 0 to a maximum of 100. In Appendix B we will also show and comment some additional charts portraying how algorithms and evaluation scores behave with respect to the balance and relative overlap parameters ρ and κ discussed in Section 5.3.1.

Throughout this section we will refer to a clustering obtained directly from an ego graph as the *basic clustering* (and consequently will speak of *basic clusters*), opposed to its hyperclustering (and consequently hyperclusters), and will use the following notation for algorithms and evaluation measures:

MCL2: Markov cluster algorithm, expansion 2, inflation 2

MCL14: Markov cluster algorithm, expansion 2, inflation 1.4

CW: Chinese Whispers

MM: MaxMax

GP: gangplank clustering algorithm

AGG97: aggregative clustering algorithm, $\sigma = 0.97$

AGG75P: aggregative clustering algorithm, $\sigma =$ 75th percentile of set Δ

CURV: curvature-based clustering algorithm

BSL: one-cluster-per-word baseline

BC-F: BCubed F-score (harmonic mean of BCubed precision and recall)

NMI: normalized mutual information

TOP2: TOP2 score, valuing the two best clusters

NOR: basic clustering

HYP: hyperclustering

5.4.1 Overall mean scores and numbers of clusters

Firstly, we display the overall scores achieved by each clustering algorithm in both cases of semantic-similarity and co-occurrence-based ego graphs. The numbers in Tables 5.2 and 5.3 represent the mean scores over all pseudowords together with their 95% confidence interval. The best

mean scores for each evaluation measure are highlighted. As we point out in Section 5.3.1, we do not consider collapsed pseudowords for any score computation in our evaluation framework. We write side by side the scores achieved by a basic clustering and its corresponding hyperclustering. The highest scores are highlighted.

	BC-F		NMI		TOP2	
	NOR	HYP	NOR	HYP	NOR	HYP
MCL2	67.8 ± 1.2	$\underline{76.5} \pm 0.9$	39.4 ± 1.6	27.5 ± 1.7	67.1 ± 1.5	56.2 ± 1.7
MCL14	93.0 ± 0.6	$\mathbf{91.0} \pm 0.7$	53.0 ± 2.6	36.3 ± 2.8	72.4 ± 1.8	61.2 ± 1.9
CW	$\mathbf{94.7} \pm 0.5$	85.1 ± 0.7	$\mathbf{53.2} \pm 2.7$	0.5 ± 0.4	$\mathbf{73.9} \pm 1.6$	40.5 ± 0.5
MM	18.8 ± 0.5	$\underline{75.9} \pm 0.8$	27.3 ± 0.9	$\underline{\mathbf{37.9}} \pm 1.8$	39.7 ± 0.8	$\underline{\mathbf{67.2}} \pm 1.5$
AGG97	68.8 ± 1.1	$\underline{85.1} \pm 0.7$	42.6 ± 1.8	0.2 ± 0.3	67.6 ± 1.3	41.2 ± 0.4
AGG75P	58.8 ± 1.2	$\underline{81.6} \pm 0.7$	39.9 ± 1.8	14.2 ± 1.7	66.1 ± 1.4	41.2 ± 1.6
CURV	70.2 ± 1.0	$\underline{70.3} \pm 1.0$	4.9 ± 0.3	4.7 ± 0.3	34.6 ± 1.0	34.2 ± 1.0
GP	55.0 ± 1.2	$\underline{78.4} \pm 0.8$	30.4 ± 1.4	$\underline{35.9} \pm 2.0$	58.6 ± 1.2	$\underline{64.5} \pm 1.4$
BSL	85.1 ± 0.7	-	0.0 ± 0	-	41.1 ± 0.4	-

Table 5.2: Overall mean scores over all pseudowords in the semantic-similarity-based ego graph data set, for all clustering algorithms, both for basic clusterings and their respective hyperclusterings. For each evaluation measure, the best score (respectively for normal clustering and hyperclustering) achieved by an algorithm is boldfaced. Underlined values show an improvement from the normal clustering to the hyperclustering. The 95% confidence interval is also reported for each mean value.

Similarly, Table 5.4 shows how many clusters were found by each clustering algorithm on average, and compares these numbers with the reduced sizes of the respective hyperclusterings.

The two tables already reveal the tendencies of each evaluation measure, and in particular the opposite extremes of BCubed and NMI. They become particularly apparent when comparing the scores of basic clusterings to those of hyperclusterings, using Table 5.4 as an additonal clue. We notice the correlation between the very low, nearly minimal mean size of clusterings produced by Chinese Whispers (oscillating between 1 and 2 for semantic similarities and slightly finer for co-occurrences) and very high BCubed scores on both data sets. Conversely, MaxMax is by far the finest, most fragmented system, only matched by MCL with parameters (2,2) in the case of co-occurrence-based ego graphs, reaching in many cases 40, 50 or more miniclusters, and achieving the lowest BCubed scores. In the tables from 5.5 to 5.10 this behaviour will be examined more in detail. From a more general point of view, we see that this situation is almost reversed when considering the NMI score: now, MaxMax becomes one of the best

	BC-F		NMI		TOP2	
	NOR	HYP	NOR	HYP	NOR	HYP
MCL2	35.3 ± 0.7	$\underline{45.7} \pm 1.0$	9.4 ± 0.3	$\mathbf{8.5} \pm 0.3$	33.4 ± 0.5	$\underline{34.6} \pm 0.7$
MCL14	69.1 ± 0.9	$\underline{77.2} \pm 0.7$	5.4 ± 0.3	4.0 ± 0.2	39.3 ± 0.7	35.6 ± 0.9
CW	88.7 ± 0.5	$\underline{\mathbf{90.3}} \pm 0.4$	4.1 ± 0.4	0.1 ± 0.0	25.6 ± 1.1	$\underline{34.9} \pm 0.7$
MM	35.2 ± 0.7	$\underline{51.0} \pm 0.7$	$\mathbf{11.1} \pm 0.4$	5.8 ± 0.3	34.2 ± 0.6	$\underline{36.7} \pm 0.7$
AGG97	50.3 ± 0.8	$\underline{81.5} \pm 0.7$	$\mathbf{11.1} \pm 0.5$	3.9 ± 0.4	41.7 ± 0.6	39.9 ± 0.8
AGG75P	55.9 ± 0.7	$\underline{82.6} \pm 0.6$	10.1 ± 0.5	3.8 ± 0.4	$\mathbf{42.8} \pm 0.7$	38.0 ± 0.9
CURV	77.5 ± 0.9	77.5 ± 0.9	4.8 ± 0.3	4.5 ± 0.3	35.8 ± 1.0	35.3 ± 1.0
GP	58.2 ± 2.0	$\underline{76.9} \pm 1.0$	4.2 ± 0.4	2.1 ± 0.3	35.4 ± 0.5	$\underline{\mathbf{40.6}} \pm 0.5$
BSL	$\mathbf{90.5} \pm 0.4$	-	0.0 ± 0	-	38.8 ± 0.5	-

Table 5.3: Overall mean scores over all pseudowords in the co-occurrence-based ego graph data set, for all clustering algorithms, both for basic clusterings and their respective hyperclusterings. For each evaluation measure, the best score (respectively for normal clustering and hyperclustering) achieved by an algorithm is boldfaced. Underlined values show an improvement from the normal clustering to the hyperclustering.

performers, and the best for hyperclustering on semantic-similarity ego graphs and for co-occurrences. With regard to NMI, it seems illuminating how our trivial one-cluster-per-word baseline BSL always produces a null score, whereas its BCubed scores are among the highest ones.

Despite the biases we will put in evidence in Section 5.4.1.1, Chinese Whispers emerges as the best system for semantic-similarity-based ego graphs. Its high TOP2 score reflects the fact that, even if it often conflates all elements into one single cluster, its average two clusters represent the two pseudosenses quite faithfully. This is also mirrored in the drop of scores for its hyperclustering: once the peak has been reached, performances can only decline, and more so if we mostly obtain a single hypercluster. The MCL and aggregative variants also perform analogously.

For co-occurrences, the picture is not so distinct and we do not have a clear winner, although the aggregative clustering algorithm with variable radius based on the 75th percentile and the gangplank clustering algorithm seem to perform slightly better than the others. Aided by the more dispersive nature of co-occurrence word graphs, reflected in the analyses of Sections 5.2.2, 5.2.2.1 and 5.3.1, and by the bias discussed in 5.4.1.1, our trivial baseline outclasses all other algorithms in terms of BCubed measure, and has an average TOP2 score. Incidentally, we notice that similar TOP2 scores are due to its bounded nature, in that it evaluates just two clusters and is capped to $1/2$ for maximally skewed clusterings.

	Similarities		Co-occurrences	
	NOR	HYP	NOR	HYP
MCL2	18.7 ± 0.8	7.8 ± 0.3	50.2 ± 0.9	38.8 ± 0.6
MCL14	2.9 ± 0.1	1.6 ± 0.0	14.1 ± 0.3	9.0 ± 0.3
CW	2.0 ± 0.1	1.1 ± 0.0	4.0 ± 0.1	1.1 ± 0.0
MM	43.8 ± 0.8	5.0 ± 0.1	42.9 ± 0.6	8.3 ± 0.2
AGG97	5.4 ± 0.2	1.0 ± 0.0	7.9 ± 0.2	1.9 ± 0.0
AGG75P	6.8 ± 0.2	1.0 ± 0.0	6.5 ± 0.2	1.8 ± 0.0
CURV	10.4 ± 0.3	9.7 ± 0.3	8.7 ± 0.2	8.1 ± 0.2
GP	7.5 ± 0.2	2.5 ± 0.0	6.3 ± 0.3	2.0 ± 0.1
BSL	1	1	1	1

Table 5.4: Overall mean numbers of clusters over all pseudowords for all clustering algorithms, comparing basic clusterings and their respective hyperclusterings. The 95% confidence interval is also shown for each mean value.

In the case of co-occurrences we see that hyperclustering can indeed help better focus a basic clustering, as in all cases for BCubed measure and in half of the cases for TOP2 the scores improve, independently from the fine-grainedness of an algorithm. This principally tells us two facts: it backs up the impression that co-occurrence clustering is a harder task to tackle, and it reveals that all systems we selected essentially produce skewed clusterings consisting of few bigger agglomerations and a host of smaller clusters. This might be tied to the small-world nature of word graphs (see Section 2.1.7).

We will delve deeper into the single clustering algorithms in Section 5.4.2, observing their detailed behaviours for frequency class combinations and pseudoword parameters.

5.4.1.1 Measure biases

Given the previous observations, we claim to have gathered further evidence of the biases of BCubed measures and NMI.

BCubed measures (see Section 2.2.2), due to their nature as averages over all single clustered elements, stress the similarity between the *internal* structures, i.e. the distribution of elements inside each cluster, of two clusterings and disregard their *external* structures, i.e. their respective sizes and the distribution of cardinalities among clusters. The fact that many pseudowords in both our data sets are very unbalanced (in the sense explored in Section 5.3.1) and that e.g. CW tends to produce coarse clusterings originates extremely high BCubed scores for this algorithm,

as this measure does not give enough relevance to the smaller one of the two ground truth clusters. In fact, if the algorithm manages to find only one cluster, the BCubed score is the same as for the one-cluster-per-word trivial baseline, which itself corresponds to the value $\frac{\max(|\alpha|,|\beta|)}{|\alpha|+|\beta|}$ (in the notation of identity (5.1)); the more skewed the pseudoword, the higher this value. The same considerations are valid for MCL with parameters (2,1.4) and less pronouncedly for the aggregative clustering algorithm with radius $\sigma = 0.97$. We might argue that if a pseudoword is nearly collapsed onto one single component, a high BCubed score effectively recognizes this and rewards the algorithm for not being too fragmentary. Still, we want an algorithm to isolate both components, and we deem that TOP2 is better suited to evaluate this in the case of clear-cut homonymy.

We exemplify the bias of BCubed by taking the pseudoword *lightbulb_wedlock* as an example in the case of the semantic-similarity-based ego graph data set. Its balance parameter ρ (see Section 5.3.1) is 6.2, so it is quite unbalanced in favour of *lightbulb*, which has 403 exclusive elements in α compared to the 65 for *wedlock* in β, which is still a not negligible number. The relative overlap κ is nearly zero, because there is only one word common to their respective distributional thesauri. With such premises, CW fails to tell the difference and produces only one cluster; nonetheless, it obtains a very high BCubed score of $0.86 = \frac{403}{468} = \frac{|\alpha|}{|\alpha|+|\beta|}$. Conversely, TOP2 assesses a score of 0.45, smaller than $\frac{1}{2}$, as defined in Section 5.2.3.

On the other hand, NMI tendentially penalizes a small number of clusters too heavily[14], and it seems sometimes quite difficult to interpret, as its variations are too brisk; in general, though, NMI seems more adherent to the desired kind of evaluation (as discussed in Section 5.2.3) than BCubed. Still, this briskness does not take account of the skewness of a pseudoword at all. We see this by the example of two pseudowords, in the case of semantic-similarity-based ego graphs, with very different balance parameters: *paycheck_reptile*, fairly balanced with $\rho = 2.0$ (160 terms for *paycheck*, 322 for *reptile*), and *artistry_bufflehead*, with a very high $\rho = 35.62$ (413 terms for *artistry* and only 13 for *bufflehead*). We will consider the algorithm AGG75P and its hyperclustering. For *paycheck_reptile*, 2 good clusters are found, for an excellent NMI score of 0.96; for *artistry_bufflehead*, the clustering is quite fragmented and yields 9 clusters that score a NMI of 0.28. Now, both hyperclusterings are greatly condensed to one single hypercluster. Then, for both the NMI score is exactly 0, as is always the case when one single cluster is found. However, the TOP2 score behaves differently: in the first case we obtain 0.39, while in the second one 0.48. Despite

[14]This bias is also discussed at length in [Li et al., 2014].

being both under the threshold of $\frac{1}{2}$ (see again Section 5.2.3), the higher, second one mirrors the intuition that it is "less wrong" to assign a single cluster to a very unbalanced sense distribution (like for *artistry_bufflehead*) than to a more uniform one (like for *paycheck_reptile*).

Moreover, in a global comparison like those of Tables 5.2, 5.3 and 5.4, we remark e.g. that the two versions of the aggregative clustering algorithm (again in the case of semantic-similarity-based ego graphs), though both having hyperclusterings which gravitate around a single cluster, have their NMI scores for hyperclustering separated by a huge gap: 0.2 against 14.2. This is not really justified by a huge difference in their hyperclustering quality, but comes instead from the fact that, as we claim, normalized mutual information is too much influenced by small differences in size between the compared clusterings and too susceptible to outliers.

5.4.2 Detailed scores for frequency class combinations

In the following Tables 5.5-5.10, each square shows the mean scores of each algorithm for the respective combination of frequency classes. We remember that frequency classes are sorted from 1, least frequent, to 5, most frequent (see Section 5.2.1). Their combinations will be denoted by the juxtaposition of two numbers: so, e.g. 34 represents all the pseudowords with one component in the third and the other in the fourth frequency class. Given the observations about collapsed pseudowords in Section 5.2.2.1 and the pseudoword analysis in Section 5.3.1, we can regard the left-upper to middle region (11, 12, 22,..., but also 33 and so on) of each table as referring to the more balanced pseudoword ego graphs, while conversely the rightmost side (15, 25, 35,..., but also 55) is characterized by a greater unbalance. From time to time we will speak of *balanced* and *unbalanced regions* of the table.

Tied to this rough subdivision, as a first general consideration we notice that BCubed scores generally improve for the hyperclusterings, especially on the unbalanced regions of the tables. We expect this from the bias towards the baseline discussed in Section 5.4.1.1. The fewer the clusters and the more unbalanced a pseudoword, the more the clustering aligns to our trivial baseline. Normalized mutual information also follows its already evidenced general trend of scores decreasing from the more balanced (finer clusterings) to the more unbalanced (polarized, coarser clusterings) side of the table. Nonetheless, we notice that all three scores, BC-F, NMI and TOP2 tend to agree, particularly on more balanced pseudowords. Also the TOP2 score tends to drop in the unbalanced regions, albeit not as sharply as NMI. This is a consequence of the more polarized clusterings:

we have lesser significant clusters among which to choose the representative clusters defined by (5.4) and their purity and completeness scores are worse than for balanced pseudowords.

Semantic-similarity-based ego graphs In this paragraph we refer to Table 5.5 for the three algorithms taken from literature, to Table 5.6 for our proposed algorithms and to Table 5.9 for the number of found clusters. Among basic clusterings, Chinese Whispers regularly obtains the best scores for most combinations, and is followed closely by the MCL variants, especially the more balanced one with inflation parameter 1.4. Going to hyperclusterings, CW loses its leadership to MCL14, and the general drop in TOP2 scores is more marked for CW than for other systems, notably in the balanced regions, due to the propensity to be reduced to a single cluster. Overall, it is interesting to notice that the three scores are slightly more concordant for basic clusterings than for hyperclusterings. In terms of TOP2 scores, the two MCL variants are comparable to the two aggregative variants and alternatively best each other. We notice that the aggregative clustering algorithm is doing better on the balanced regions when using a variable radius (dependent on the distance distribution) than when using the fixed value 0.97. A similar trend persists for its hyperclustering. This is mirrored by the numbers of clusters they respectively find. AGG97 finds fewer clusters for unbalanced pseudowords dominated by components in the 4th or 5th frequency class: in such ego graphs nodes are mostly tied to the dominant component and closer to each other, so that 0.97 becomes a higher threshold than the 75th percentile over all distance values. The situation is clearly reversed for more balanced pseudowords. The fixed radius also causes AGG97 to flatten on 1 hypercluster, whereas AGG75P often maintains approximately two distinct hyperclusters. The gangplank clustering algorithm appears rather stable, both in terms of clustering sizes and scores, across all regions of the table, and places itself between the two extreme behaviours of CW and MaxMax. Its performances are decidedly improved by hyperclustering, especially TOP2 scores in the balanced region, and its stability persists in the reduced number of clusters. This behaviour looks positive for our task of homonymy detection with a fixed number of pseudosenses, but consequently suffers the skewedness of unbalanced pseudowords. A similar pattern is observed for the curvature-based clustering algorithm CURV, which preserves nearly the same clustering size in its hyperclustering, and also maintains constantly low TOP2 scores. At the same time, it shows a trend to find more clusters when high-frequency components are involved. We can explain this with the fact that in such cases the ego graph is more polarized around some very frequent words with separated positive-curvature regions around them. Finally,

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
MCL2	85.4	71.1	84.4	71.8	56.3	77.7	52.5	40.6	68.4	77.0	16.6	58.4	58.5	13.2	49.9	
MCL14	92.6	71.9	79.7	89.7	67.6	81.1	87.4	64.0	83.4	**99.0**	**35.9**	**65.5**	98.0	26.3	55.0	
CW	**94.7**	**79.6**	**85.6**	**94.2**	**78.4**	**87.0**	**91.8**	**73.0**	**86.8**	98.9	30.6	63.6	**99.4**	**29.7**	**61.4**	1
MM	37.0	45.5	50.8	26.8	39.7	46.4	17.6	32.5	40.2	17.0	11.2	42.5	13.2	10.3	37.7	
BSL	72.5	0.0	35.5	75.8	0.0	38.2	81.5	0.0	41.6	98.4	0.0	47.9	98.3	0.0	44.1	
			MCL2	65.2	53.3	71.4	52.3	42.8	64.3	76.8	21.1	63.8	59.5	9.5	42.2	
			MCL14	88.2	63.5	80.4	85.4	58.3	**79.0**	**98.2**	**43.2**	**68.4**	97.8	**12.9**	38.5	
			CW	**92.4**	**74.5**	**86.8**	**87.5**	**59.6**	78.8	98.2	27.2	60.9	**98.5**	10.9	**49.6**	2
			MM	22.7	38.9	40.8	16.3	33.6	34.8	16.9	13.1	45.3	12.5	9.0	28.6	
			BSL	73.8	0.0	37.2	76.4	0.0	38.0	97.4	0.0	46.4	98.0	0.0	**43.7**	
						MCL2	46.0	38.5	56.8	78.1	27.8	65.7	61.2	10.6	46.0	
						MCL14	83.5	**49.9**	**75.0**	**97.7**	**41.8**	**66.1**	97.3	**13.2**	44.1	
						CW	**84.3**	43.4	69.9	97.5	25.8	59.3	**97.7**	9.1	**49.4**	3
						MM	12.7	31.0	28.4	17.8	15.4	43.1	12.4	9.6	29.3	
						BSL	76.6	0.0	38.1	**96.5**	0.0	45.4	**97.5**	0.0	**43.3**	
									MCL2	84.9	68.9	83.9	75.8	53.1	77.3	
									MCL14	93.9	69.3	78.0	96.3	68.9	79.1	
									CW	**96.0**	**81.9**	**86.4**	**96.8**	**69.5**	**80.5**	4
									MM	21.4	40.0	41.7	18.2	33.1	39.6	
									BSL	77.2	0.0	36.6	**82.8**	0.0	39.6	
												MCL2	75.2	56.5	81.4	
												MCL14	96.6	**78.9**	**86.5**	
												CW	**96.9**	76.9	85.5	5
												MM	17.6	35.7	38.8	
												BSL	81.0	0.0	40.2	

(a) Scores of the basic clusterings.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
MCL2	79.2	38.0	62.5	75.1	37.4	66.3	66.8	30.8	60.5	84.3	11.9	42.7	78.0	**13.3**	45.8	
MCL14	**91.2**	**66.2**	76.1	**84.8**	36.8	59.1	**83.7**	20.6	53.0	**98.9**	**31.9**	**65.3**	98.3	9.0	**48.3**	
CW	72.5	0.0	35.9	75.7	0.3	37.3	81.5	0.1	37.8	98.4	2.8	49.2	**98.3**	0.0	43.4	1
MM	81.8	61.6	**78.3**	81.3	**55.9**	**78.9**	73.0	**39.8**	**72.9**	69.3	11.5	53.2	71.7	12.6	45.5	
BSL	72.5	0.0	35.5	75.8	0.0	38.2	81.5	0.0	41.6	98.4	0.0	47.9	**98.3**	0.0	44.1	
			MCL2	71.6	38.5	66.3	66.5	31.9	61.4	86.1	15.3	48.0	75.2	8.3	34.1	
			MCL14	81.1	27.8	54.6	**81.4**	26.7	53.5	**98.2**	**41.4**	**68.1**	97.9	**8.9**	39.9	
			CW	73.8	0.4	35.7	76.3	0.4	35.7	97.4	0.0	46.4	**98.0**	0.0	**43.8**	2
			MM	**84.7**	**60.9**	**82.5**	76.9	**46.8**	**76.9**	69.3	13.5	52.1	70.1	7.1	35.8	
			BSL	73.8	0.0	37.2	76.4	0.0	38.0	97.4	0.0	46.4	**98.0**	0.0	43.7	
						MCL2	63.4	29.7	62.1	85.3	20.2	51.8	75.7	**7.7**	39.7	
						MCL14	**79.9**	19.8	50.3	**97.7**	**39.0**	**64.8**	97.5	6.4	42.6	
						CW	76.6	0.1	36.9	96.5	1.2	46.1	**97.5**	1.8	**44.3**	3
						MM	73.7	**40.3**	**72.9**	71.4	18.8	56.9	67.6	6.8	39.9	
						BSL	76.6	0.0	38.1	96.5	0.0	45.4	97.5	0.0	**43.3**	
									MCL2	84.6	50.0	68.6	82.3	35.8	58.6	
									MCL14	**93.8**	67.5	76.9	**94.9**	**63.2**	**77.2**	
									CW	77.2	0.0	36.6	82.8	0.0	39.9	4
									MM	84.4	**69.8**	**84.9**	80.9	53.4	76.8	
									BSL	77.2	0.0	36.6	82.8	0.0	39.6	
												MCL2	77.4	35.3	65.6	
												MCL14	**95.4**	**74.9**	**83.9**	
												CW	81.0	0.0	40.2	5
												MM	84.4	60.2	81.5	
												BSL	81.0	0.0	40.2	

(b) Scores of the hyperclusterings.

Table 5.5: Mean scores per frequency class combination over the semantic-similarity-based ego graph data set of the three clustering algorithms from literature, for basic clustering and hyperclustering. The best values are highlighted.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
AGG75P	**76.1**	63.6	75.8	71.8	**61.3**	**77.4**	60.5	**47.5**	**73.5**	44.3	14.4	55.8	43.0	**13.7**	**51.5**	
AGG97	73.4	**65.1**	**77.5**	64.0	56.2	73.1	55.0	46.0	70.1	77.5	**24.6**	**62.3**	58.4	12.9	52.9	
CURV	56.9	8.8	43.3	68.5	5.6	38.9	73.2	4.9	38.2	69.1	3.6	29.3	83.5	4.5	20.6	1
GP	70.6	57.5	76.4	62.3	46.4	71.7	50.8	34.9	62.4	49.8	8.4	43.2	46.9	7.5	36.9	
BSL	72.5	0.0	35.5	**75.8**	0.0	38.2	**81.5**	0.0	41.6	**98.4**	0.0	47.9	**98.3**	0.0	**44.1**	
			AGG75P	73.7	**63.3**	**78.7**	64.8	**53.2**	**74.3**	45.1	17.0	59.5	40.3	**9.4**	38.4	
			AGG97	62.0	55.6	70.9	58.2	49.0	69.8	77.5	**27.5**	**65.0**	63.4	7.9	42.7	
			CURV	70.6	5.4	39.9	71.8	5.2	38.1	67.5	2.8	25.2	74.8	3.3	23.9	2
			GP	60.4	47.3	72.7	55.5	40.4	67.0	51.6	9.7	45.2	48.1	6.8	35.7	
			BSL	**73.8**	0.0	37.2	**76.4**	0.0	38.0	**97.4**	0.0	46.4	**98.0**	0.0	**43.7**	
						AGG75P	64.0	**49.6**	**72.2**	45.9	20.4	57.6	42.6	**10.3**	41.5	
						AGG97	56.8	45.1	67.5	76.6	**29.8**	**64.1**	63.6	9.6	**46.1**	
						CURV	70.0	5.6	38.0	76.4	4.1	30.1	74.8	3.1	23.5	3
						GP	53.2	36.1	65.8	46.3	13.3	47.9	43.4	7.8	36.5	
						BSL	**76.6**	0.0	38.1	**96.5**	0.0	45.4	**97.5**	0.0	43.3	
									AGG75P	70.7	53.5	71.1	63.2	47.4	69.9	
									AGG97	**86.4**	**70.0**	**80.2**	80.7	**57.4**	**75.9**	
									CURV	63.0	7.8	43.3	65.9	5.5	38.0	4
									GP	69.2	48.2	69.5	59.8	37.8	64.3	
									BSL	77.2	0.0	36.6	**82.8**	0.0	39.6	
												AGG75P	70.2	55.3	75.9	
												AGG97	**81.4**	**65.3**	**81.7**	
												CURV	69.4	4.3	40.7	5
												GP	59.4	43.7	70.1	
												BSL	81.0	0.0	40.2	

(a) Scores of the basic clusterings.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
AGG75P	75.6	21.9	53.4	76.7	11.6	42.9	**81.9**	20.7	44.1	89.4	**20.8**	44.2	91.8	**10.7**	27.6	
AGG97	72.5	0.0	35.5	75.8	0.0	38.2	81.5	0.0	41.2	98.4	2.8	**49.2**	**98.3**	0.0	**44.1**	
CURV	56.9	8.7	43.4	68.6	5.3	38.2	73.3	4.4	36.9	69.1	3.6	29.4	83.7	4.5	20.7	1
GP	**88.1**	**70.1**	**83.9**	**83.3**	**56.1**	**78.7**	74.1	**39.8**	**68.6**	76.0	8.4	46.8	74.3	7.1	39.9	
BSL	72.5	0.0	35.5	75.8	0.0	38.2	81.5	0.0	41.6	**98.4**	0.0	47.9	**98.3**	0.0	**44.1**	
			AGG75P	75.4	17.1	49.1	74.3	11.1	43.1	88.9	**22.3**	43.2	85.5	1.9	21.5	
			AGG97	73.8	0.0	37.2	76.4	0.1	37.7	**97.4**	0.0	46.4	**98.0**	0.0	43.7	
			CURV	70.8	4.9	39.2	71.8	4.5	37.6	67.5	2.8	25.2	75.0	3.3	24.6	2
			GP	**82.1**	**57.3**	**81.3**	**78.1**	**47.6**	**75.7**	77.1	8.9	**46.9**	77.2	**5.6**	39.2	
			BSL	73.8	0.0	37.2	76.4	0.0	38.0	**97.4**	0.0	46.4	**98.0**	0.0	**43.7**	
						AGG75P	**77.3**	11.3	45.3	89.4	**16.8**	40.5	88.7	5.1	27.5	
						AGG97	76.6	0.0	38.1	96.5	0.0	45.4	97.5	0.0	43.3	
						CURV	70.4	5.0	37.7	76.5	3.9	29.3	74.9	3.0	23.0	3
						GP	71.9	**40.2**	**72.7**	73.9	15.2	**51.0**	72.7	**8.1**	42.2	
						BSL	76.6	0.0	38.1	**96.5**	0.0	45.4	**97.5**	0.0	43.3	
									AGG75P	77.3	17.8	47.9	79.5	10.1	38.8	
									AGG97	77.2	0.0	36.6	82.8	0.0	39.6	
									CURV	63.1	7.8	43.4	65.9	5.4	37.8	4
									GP	**89.0**	**65.3**	**79.4**	80.5	**46.1**	**70.9**	
									BSL	77.2	0.0	36.6	**82.8**	0.0	39.6	
												AGG75P	76.2	8.6	41.0	
												AGG97	**81.0**	0.0	40.2	
												CURV	69.4	4.3	40.7	5
												GP	80.2	**52.5**	**75.5**	
												BSL	**81.0**	0.0	40.2	

(b) Scores of the hyperclusterings.

Table 5.6: Mean scores per frequency class combination over the semantic-similarity-based ego graph data set of our three proposed clustering algorithms, for basic clustering and hyperclustering. The best values are highlighted.

the most surprising behaviour is that of MaxMax. Its basic clustering is by far the worst system together with CURV: it tends to score quite low, confronted with the other algorithms, for all three evaluation measures. This is surely connected to its extreme fragmentation, as it very often finds above 40 clusters. Even if the precision of each is high, the recall is at its lowest, and TOP2 can not find two clusters that are representative enough. The scenario totally changes when we introduce hyperclustering: then, the number of clusters becomes sensible and MaxMax often achieves the best scores, especially for NMI and TOP2. This means that MM adapts particularly well to hyperclustering and that the highly precise smaller building blocks are effectively combined in meaningful bigger entities.

As a final comment, we might argue that the criterion for identifying monsemous terms explained in Section 5.2.1 making use of the Chinese Whispers clustering algorithm creates a bias towards it in our evaluation framework. While this may be partially true, we add the look-up in WordNet for this reason, and other clustering algorithms still achieve comparable or better results than CW. We will also see that the picture is fairly different for co-occurrence-based ego graphs.

Co-occurrence-based ego graphs In this paragraph we refer to Table 5.7 for the three algorithms taken from literature, to Table 5.8 for our proposed algorithms and to Table 5.10 for the number of found clusters. At a first glance, scores for co-occurrence-based ego graphs are dominated by the trivial baseline performance. Further, there is a bigger discrepancy here between BC-F and TOP2 scores. We note that CW, while being mostly aligned to the baseline (mostly due to its small number of clusters) and thus having the second-best BC-F scores, is more often than not outclassed by the other algorithms in terms of TOP2. The reason is that its clusterings consists of one catch-all cluster that covers around 95% of the nodes in an ego graph, while the remaining terms are split amongst a handful of micro clusters, and the already discussed bias of BCubed measures does not penalize this clustering structure. Otherwise, all other systems seem to settle inside a relatively homogenous range of values, with the aggregative variants usually rising slightly above the others, with a prevalence of AGG75P. As far as TOP2 scores go, we notice a generalized expected tendency of scores decreasing progressively from the balanced to the more unbalanced regions. Only the gangplank clustering algorithm seems to improve a bit when more frequent components are involved. For the rest, most individual considerations for semantic-similarity-based ego graphs remain still valid for co-occurrences; we just note that scores are everywhere lower, as discussed in Section 5.4.1. After hyperclustering, the picture remains substantially unvaried, after we register a further general lowering of all

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
MCL2	42.0	13.2	**43.1**	37.6	10.8	38.0	39.4	6.2	34.2	35.3	3.6	30.3	37.6	3.7	40.3	
MCL14	61.3	6.3	41.1	63.2	5.0	**45.8**	74.4	4.0	46.4	70.9	2.2	30.2	76.8	**5.1**	37.1	
MM	49.6	**17.1**	42.2	40.7	**13.4**	39.7	38.3	**7.3**	39.2	34.7	**3.7**	28.0	36.9	3.7	32.7	1
CW	69.9	7.1	38.3	80.5	4.4	34.4	92.5	3.2	25.9	95.8	0.3	7.3	98.7	0.1	12.8	
BSL	**71.9**	0.0	34.4	**83.5**	0.0	42.0	**94.4**	0.0	**46.7**	**98.9**	0.0	**47.9**	**99.7**	0.0	**48.9**	
			MCL2	34.1	11.5	35.8	34.9	10.0	33.9	34.2	7.9	31.0	36.9	4.9	29.8	
			MCL14	60.5	4.8	43.3	67.5	5.0	**40.5**	65.8	4.6	38.8	77.6	4.6	34.0	
			MM	35.0	**13.8**	35.6	34.6	**11.8**	33.5	34.2	**8.7**	34.5	36.2	5.1	30.3	2
			CW	81.8	3.1	32.5	87.1	3.4	30.4	92.7	2.8	20.2	97.0	**5.7**	22.6	
			BSL	**83.0**	0.0	**38.2**	**88.3**	0.0	36.8	**94.7**	0.0	**41.9**	**98.0**	0.0	**44.0**	
						MCL2	34.0	11.0	32.9	32.0	11.2	30.1	35.4	7.4	30.6	
						MCL14	69.6	5.4	**34.0**	66.4	6.4	**37.1**	77.2	6.8	**43.1**	
						MM	32.7	**13.7**	27.9	31.7	**13.2**	30.6	33.4	**8.1**	34.5	3
						CW	85.6	3.3	25.1	88.4	3.3	23.2	94.6	5.7	24.4	
						BSL	**86.7**	0.0	28.3	**89.6**	0.0	32.9	**95.8**	0.0	40.3	
									MCL2	31.4	15.1	32.8	33.2	11.8	34.9	
									MCL14	59.2	8.5	**35.1**	71.9	7.7	**39.8**	
									MM	31.1	**18.5**	32.9	31.6	**14.3**	35.6	4
									CW	84.7	5.0	26.7	89.1	7.7	31.5	
									BSL	**86.8**	0.0	27.3	**90.5**	0.0	35.5	
												MCL2	33.4	16.1	31.9	
												MCL14	73.0	7.8	**33.7**	
												MM	30.5	**19.2**	32.4	5
												CW	85.1	4.3	27.2	
												BSL	**86.8**	0.0	27.6	

(a) Scores of the basic clusterings.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
MCL2	51.7	**12.5**	44.3	46.9	**9.9**	40.1	48.1	**5.6**	37.3	45.4	**3.0**	28.1	53.8	**3.7**	33.9	
MCL14	66.1	5.0	39.3	71.4	4.0	45.0	82.0	3.3	41.6	81.9	1.6	22.5	88.1	3.3	21.8	
MM	64.3	9.2	**49.0**	56.3	6.9	**47.2**	53.1	3.7	40.8	49.9	2.4	25.5	57.1	2.4	25.9	1
CW	71.7	0.1	34.5	83.0	0.2	41.6	94.1	0.1	37.1	98.3	0.1	30.1	99.5	0.0	42.9	
BSL	**71.9**	0.0	34.4	**83.5**	0.0	42.0	**94.4**	0.0	**46.7**	**98.9**	0.0	**47.9**	**99.7**	0.0	**48.9**	
			MCL2	42.2	**10.4**	38.2	45.4	**9.0**	35.6	43.6	**6.8**	33.4	51.5	**4.3**	28.7	
			MCL14	68.9	3.5	42.4	75.6	3.8	38.4	77.0	3.4	33.3	86.3	3.2	28.7	
			MM	49.5	6.2	**42.9**	48.1	5.5	**38.5**	49.3	4.7	33.8	55.3	2.9	27.8	2
			CW	83.0	0.1	37.2	88.3	0.0	35.5	94.4	0.3	35.3	97.8	0.2	39.7	
			BSL	**83.0**	0.0	38.2	**88.3**	0.0	36.8	**94.7**	0.0	**41.9**	**98.0**	0.0	**44.0**	
						MCL2	45.2	**10.1**	31.3	40.1	**9.7**	31.8	49.3	**6.7**	34.4	
						MCL14	74.3	3.7	32.6	74.5	4.6	**35.7**	84.7	4.7	34.9	
						MM	46.4	6.2	**33.2**	45.9	6.6	35.2	52.5	4.0	34.2	3
						CW	86.6	0.1	28.0	89.5	0.1	32.2	95.6	0.1	36.4	
						BSL	**86.7**	0.0	28.3	**89.6**	0.0	32.9	**95.8**	0.0	**40.3**	
									MCL2	39.1	**14.1**	34.8	43.7	**11.2**	35.6	
									MCL14	70.6	6.2	33.5	77.7	5.9	**37.5**	
									MM	46.4	9.8	**39.5**	48.7	8.5	36.6	4
									CW	86.8	0.0	27.5	90.4	0.0	35.3	
									BSL	**86.8**	0.0	27.3	**90.5**	0.0	35.5	
												MCL2	47.1	**14.9**	33.5	
												MCL14	76.9	5.0	33.3	
												MM	48.7	12.2	**37.4**	5
												CW	86.7	0.0	27.7	
												BSL	**86.8**	0.0	27.6	

(b) Scores of the hyperclusterings.

Table 5.7: Mean scores per frequency class combination over the co-occurrence-based ego graph data set of the three clustering algorithms from literature, for basic clustering and hyperclustering. The best values are highlighted.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
AGG75P	53.1	13.1	**55.1**	51.5	**10.9**	**50.3**	55.7	**6.6**	44.9	66.9	4.6	40.5	55.9	**3.7**	36.3	
AGG97	33.5	**19.4**	43.0	43.0	12.4	45.3	59.5	**6.6**	**46.9**	59.1	**4.8**	39.7	43.6	3.4	31.2	
CURV	53.1	6.8	45.1	68.6	5.6	46.4	80.7	4.7	43.2	92.0	1.8	13.9	91.0	3.5	18.5	1
GP	48.1	5.3	40.9	45.0	5.0	37.1	27.7	4.5	28.0	57.3	2.5	34.7	73.7	0.7	37.9	
BSL	**71.9**	0.0	34.4	**83.5**	0.0	42.0	**94.4**	0.0	46.7	**98.9**	0.0	**47.9**	**99.7**	0.0	**48.9**	
			AGG75P	50.9	8.4	**47.3**	50.9	**8.0**	43.0	63.9	9.0	42.0	56.0	5.4	35.2	
			AGG97	51.8	**8.9**	46.4	58.0	7.5	**45.1**	56.4	**9.5**	**42.8**	45.2	**6.0**	34.1	
			CURV	71.9	4.6	42.8	78.1	3.9	39.1	82.5	6.0	37.3	84.1	5.1	29.3	2
			GP	49.6	5.7	36.9	29.2	7.2	31.0	62.7	5.1	38.2	86.0	0.7	41.1	
			BSL	**83.0**	0.0	38.2	**88.3**	0.0	36.8	**94.7**	0.0	41.9	**98.0**	0.0	**44.0**	
						AGG75P	50.0	**8.0**	38.5	60.9	12.2	39.6	53.6	7.3	34.5	
						AGG97	55.7	6.4	**38.7**	54.8	**12.4**	**40.5**	45.3	**8.8**	34.6	
						CURV	75.7	3.9	32.9	78.4	4.7	34.5	79.4	5.0	34.0	3
						GP	38.5	5.8	28.5	63.5	6.2	36.6	88.3	0.9	39.9	
						BSL	**86.7**	0.0	28.3	**89.6**	0.0	32.9	**95.8**	0.0	**40.3**	
									AGG75P	59.3	18.4	**42.6**	52.9	17.2	**45.4**	
									AGG97	52.5	**19.2**	40.9	41.2	**19.5**	43.5	
									CURV	76.0	5.2	32.3	74.0	5.5	38.2	4
									GP	60.7	7.1	34.5	81.0	2.5	36.7	
									BSL	**86.8**	0.0	27.3	**90.5**	0.0	35.5	
												AGG75P	47.4	22.2	**45.2**	
												AGG97	34.7	**26.1**	41.6	
												CURV	68.5	6.0	36.3	5
												GP	81.9	1.4	29.5	
												BSL	**86.8**	0.0	27.6	

(a) Scores of the basic clusterings.

	1			2			3			4			5			
	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	BC-F	NMI	TOP2	
AGG75P	68.3	4.2	44.0	77.8	4.1	50.4	91.8	3.4	47.4	89.2	3.0	31.0	90.1	**2.7**	35.7	
AGG97	58.7	1.7	**51.4**	75.0	4.4	**53.3**	91.1	**4.6**	**49.2**	87.2	**3.1**	31.9	93.7	1.9	38.5	
CURV	53.1	**6.7**	45.1	68.6	**5.5**	46.5	80.9	4.5	43.1	92.1	1.7	13.9	91.1	1.8	13.5	1
GP	59.1	2.7	47.5	67.8	2.3	46.4	69.7	2.2	44.1	79.2	1.2	40.6	87.9	0.8	43.7	
BSL	**71.9**	0.0	34.4	**83.5**	0.0	42.0	**94.4**	0.0	46.7	**98.9**	0.0	**47.9**	**99.7**	0.0	**48.9**	
			AGG75P	79.1	2.7	41.8	86.0	1.4	36.9	85.1	3.2	30.7	87.1	1.6	32.0	
			AGG97	79.0	2.7	42.5	85.5	1.8	37.9	84.7	3.3	31.4	86.2	1.7	34.7	
			CURV	71.9	**4.4**	42.9	78.2	**3.5**	39.1	82.5	**5.0**	35.2	84.2	**4.0**	25.9	2
			GP	70.5	2.5	**43.3**	66.2	2.8	**42.2**	79.5	2.3	40.3	92.0	0.4	42.1	
			BSL	**83.0**	0.0	38.2	**88.3**	0.0	36.8	**94.7**	0.0	**41.9**	**98.0**	0.0	**44.0**	
						AGG75P	84.5	2.0	31.0	83.0	4.7	33.6	81.8	2.4	33.2	
						AGG97	84.3	2.2	31.0	81.0	**6.7**	37.0	82.4	2.0	34.5	
						CURV	75.7	**3.7**	32.9	78.6	4.4	34.2	79.4	**4.7**	33.7	3
						GP	69.4	2.2	**35.7**	78.1	3.9	**37.7**	92.0	0.5	39.7	
						BSL	**86.7**	0.0	28.3	**89.6**	0.0	32.9	**95.8**	0.0	**40.3**	
									AGG75P	79.8	**13.4**	44.7	75.7	**5.6**	38.6	
									AGG97	76.2	12.8	**45.5**	76.2	5.2	**41.2**	
									CURV	76.7	4.8	33.2	74.0	5.4	38.2	4
									GP	75.6	4.8	36.5	85.8	0.8	36.6	
									BSL	**86.8**	0.0	27.3	**90.5**	0.0	35.5	
												AGG75P	72.2	**6.8**	**39.7**	
												AGG97	74.1	4.6	39.1	
												CURV	68.5	6.0	36.4	5
												GP	83.5	1.4	29.5	
												BSL	**86.8**	0.0	27.6	

(b) Scores of the hyperclusterings.

Table 5.8: Mean scores per frequency class combination over the co-occurrence-based ego graph data set of our three proposed clustering algorithms, for basic clustering and hyperclustering. The best values are highlighted.

	1		2		3		4		5		
	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	
MCL2	7.7	3.9	16.9	8.3	29.5	10.8	9.9	4.7	28.7	12.2	
MCL14	2.3	1.8	3.6	1.7	5.0	1.7	1.4	1.4	2.8	1.5	
CW	2.0	1.0	2.7	1.2	3.0	1.3	1.3	1.0	1.5	1.1	1
MM	20.0	3.4	36.7	4.8	52.3	5.8	36.7	5.3	52.4	6.1	
BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
		MCL2	22.1	10.5	29.4	11.9	9.7	4.4	24.2	10.4	
		MCL14	3.8	1.7	4.9	1.9	1.6	1.5	2.2	1.5	
		CW	3.0	1.2	2.9	1.3	1.3	1.0	1.2	1.0	2
		MM	45.8	4.5	55.0	5.4	36.8	5.2	50.4	5.8	
		BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
				MCL2	31.8	11.4	9.3	4.4	24.4	10.1	
				MCL14	4.6	1.7	1.6	1.5	2.1	1.4	
				CW	2.6	1.2	1.3	1.0	1.2	1.1	3
				MM	59.7	5.5	35.5	4.8	49.7	6.3	
				BSL	1.0	1.0	1.0	1.0	1.0	1.0	
						MCL2	6.6	3.2	14.1	6.4	
						MCL14	1.7	1.7	2.1	1.8	
						CW	1.9	1.0	1.8	1.1	4
						MM	35.5	3.5	44.3	4.5	
						BSL	1.0	1.0	1.0	1.0	
								MCL2	15.8	8.0	
								MCL14	2.1	1.8	
								CW	1.8	1.0	5
								MM	46.4	4.2	
								BSL	1.0	1.0	

(a) Number of clusters for the three algorithms taken from literature.

	1		2		3		4		5		
	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	
AGG75P	4.4	1.6	5.8	1.5	7.6	2.0	8.3	2.2	9.5	2.2	
AGG97	4.9	1.0	7.4	1.0	9.2	1.0	2.5	1.0	5.8	1.0	
CURV	8.7	8.5	7.8	7.2	10.5	9.1	11.7	11.5	13.0	11.9	1
GP	5.4	2.2	7.3	2.5	8.7	2.8	7.3	2.3	7.5	2.4	
BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
		AGG75P	5.7	1.6	6.9	1.8	8.0	2.1	8.9	2.0	
		AGG97	8.5	1.0	8.7	1.0	2.6	1.0	4.6	1.0	
		CURV	6.7	6.0	8.9	7.7	12.0	11.5	14.5	13.8	2
		GP	7.8	2.7	8.0	2.7	6.7	2.2	7.5	2.2	
		BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
				AGG75P	6.7	1.7	7.9	2.1	8.7	2.0	
				AGG97	8.1	1.0	2.8	1.0	4.7	1.0	
				CURV	9.9	8.4	11.0	10.5	13.1	12.4	3
				GP	8.6	2.7	7.6	2.4	8.3	2.6	
				BSL	1.0	1.0	1.0	1.0	1.0	1.0	
						AGG75P	3.6	1.4	5.2	1.6	
						AGG97	2.3	1.0	2.8	1.0	
						CURV	7.6	7.5	10.9	10.5	4
						GP	6.1	2.0	7.1	2.3	
						BSL	1.0	1.0	1.0	1.0	
								AGG75P	4.4	1.6	
								AGG97	2.8	1.0	
								CURV	11.0	10.6	5
								GP	7.2	2.3	
								BSL	1.0	1.0	

(b) Number of clusters for our three novel algorithms.

Table 5.9: Comparisons of mean number of found clusters on the semantic-similarity-based ego graph data set, for the basic clustering (NOR) and its first hyperclustering (HYP), for all discussed algorithms, in the same notation of Tables 5.5 and 5.6.

the scores. The only algorithms gaining something, not always consistently, are CW, especially when less-frequent components are involved, GP, MM and AGG97. We notice that systems that were cluster-wise more stable across frequency class combinations for semantic similarities, like GP, now tend to vary more their clustering sizes, and to produce less clusters in the unbalanced regions. This mirrors the more pronounced character of collapsed pseudowords in the case of co-occurrences, already detected in Section 5.2.2.1 and further examined in Appendix A: ego graphs are tendentially more balanced, but unbalances are starker than for semantic similarities. Finally, looking at the number of clusters, we notice the extremely high numbers for MCL2, which surpasses MaxMax as the finest clustering algorithm, and in particular how they behave after hyperclustering: while MM falls back to more reasonable clustering sizes, those of MCL2 are still very high. We want to see a sort of paradigmatic difference in the clustering approach: whereas MCL *isolates* denser regions, MaxMax *congregates* the nodes in certain types of subgraphs. Such difference is accentuated by the hyperclustering: MaxMax goes on condensing its clustering, while MCL retains its divisive nature.

5.4.3 Example of Clusterings

We briefly want to show the differences between the basic clusterings of our six systems (MCL, CW, MM, AGG, GP, CURV) and their variants on the ego graph of a same pseudoword, in order to give a small direct insight into their nature. We will also briefly look at the effects of hyperclustering. As our example, we chose the semantic-similarity-based ego graph of *euonymus_carryall*, a combination of two words belonging to the lowest frequency class 1. Its network is connected and consists of 366 nodes and has a density of 0.344, quite below the global mean (which is 0.45); this is due to the rarity of such words. The network is balanced, with a ratio ρ (see Section 5.3.1) between *euonymus* and *carryall* of 1.09. The baseline has here a BCubed F-score of 0.67, a TOP2 score of 0.31 and of course a NMI of exactly 0.

Chinese Whispers finds two clusters, which seem to correspond on one hand to the sense of *plant*, as in

{`nandina, hemlock, reed, iris,...`},

and on the other hand to the sense of *transportable container*, as in

{`can, receptacle, scarf, compartment,...`}.

As a consequence, all scores rate very high, respectively 0.97 for BCubed F-score, 0.90 for NMI and 0.99 for TOP2.

	1		2		3		4		5		
	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	
MCL2	23.0	17.4	43.4	33.5	57.1	44.1	53.5	40.9	46.9	36.6	
MCL14	8.9	6.2	14.3	9.8	15.5	10.0	14.6	9.5	11.6	7.0	
MM	16.9	2.7	36.8	7.0	48.9	10.2	46.1	9.3	39.7	7.8	1
CW	4.5	1.1	4.9	1.1	4.6	1.2	4.5	1.6	2.8	1.1	
BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
		MCL2	53.0	41.7	57.2	44.1	51.1	39.4	45.4	35.5	
		MCL14	14.9	9.3	15.1	9.3	14.4	8.8	12.2	7.8	
		MM	47.3	9.1	49.5	10.1	44.7	8.9	38.7	7.5	2
		CW	4.3	1.1	4.0	1.0	3.9	1.3	3.0	1.2	
		BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
				MCL2	59.0	46.4	54.2	41.6	46.6	36.1	
				MCL14	15.3	9.9	14.9	9.3	12.4	7.7	
				MM	50.0	10.2	46.0	8.8	39.2	7.3	3
				CW	3.8	1.0	3.7	1.1	3.1	1.1	
				BSL	1.0	1.0	1.0	1.0	1.0	1.0	
						MCL2	51.9	39.9	48.4	37.6	
						MCL14	14.4	8.5	14.3	9.6	
						MM	46.4	8.0	40.5	7.6	4
						CW	3.5	1.0	3.5	1.0	
						BSL	1.0	1.0	1.0	1.0	
								MCL2	49.5	37.7	
								MCL14	14.4	9.6	
								MM	41.2	7.3	5
								CW	3.9	1.0	
								BSL	1.0	1.0	

(a) Number of clusters for the three algorithms taken from literature.

	1		2		3		4		5		
	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	NOR	HYP	
AGG75P	3.6	1.6	6.7	1.8	8.9	1.7	6.4	1.9	5.1	1.7	
AGG97	9.7	1.9	9.9	2.0	8.5	2.0	7.5	1.9	6.7	1.5	
CURV	9.7	9.6	11.1	10.7	8.9	8.1	8.3	7.5	7.2	6.6	1
GP	5.1	2.0	7.4	2.2	12.0	2.8	7.2	2.1	4.6	1.6	
BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
		AGG75P	7.4	1.8	8.6	1.7	6.7	1.9	5.2	1.7	
		AGG97	7.8	1.9	7.4	1.9	7.5	1.9	7.0	1.7	
		CURV	10.0	9.7	7.9	7.4	8.8	7.9	8.1	7.4	2
		GP	7.1	2.2	11.3	2.9	6.0	1.9	2.8	1.3	
		BSL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
				AGG75P	8.6	1.8	6.7	1.8	5.2	1.8	
				AGG97	7.1	1.9	7.3	1.9	7.1	1.8	
				CURV	7.8	7.3	8.3	7.7	7.9	7.5	3
				GP	9.6	2.5	5.2	1.8	2.1	1.2	
				BSL	1.0	1.0	1.0	1.0	1.0	1.0	
						AGG75P	5.8	2.0	4.9	1.8	
						AGG97	7.7	1.9	7.9	1.8	
						CURV	7.5	6.7	8.7	8.3	4
						GP	5.2	1.8	2.4	1.2	
						BSL	1.0	1.0	1.0	1.0	
								AGG75P	4.6	1.8	
								AGG97	8.2	1.7	
								CURV	8.1	7.7	5
								GP	1.7	1.2	
								BSL	1.0	1.0	

(b) Number of clusters for our three novel algorithms.

Table 5.10: Comparisons of mean number of found clusters on the co-occurrence-based ego graph data set, for the basic clustering (NOR) and its first hyperclustering (HYP), for all discussed algorithms, in the same notation of Tables 5.5 and 5.6.

The gangplank algorithm gives us 8 clusters. Their precision is quite high and it is possible to distinguish *carryall*-type clusters like

{apartment,package,sheeting,canopy},

and *euonymus*-type one, although the distinction between two clusters of the same type is not always clear, like for the plant-related terms of

{clethra,tree,spiraea} and
{loosestrife,gilt,multiflora,bugger,leucothoe,...}.

The gangplank scores are: BCubed F-score 0.70, NMI 0.61, TOP2 0.78.

After hyperclustering, we get 3 more clear-cut clusters, similar to those obtained by CW. One is just a micro cluster of three terms with no relevance. Scores improve consequently: BC-F 0.94, NMI 0.80, TOP2 0.90.

Here we want very briefly to consider the effects of the minimum cut described in Section 2.1.3 and in Section 4.3.2.1 on the gangplank clustering algorithm. The basic clustering has only 5 instead of the 8 of the regular version, and the scores are slightly inferior: BCubed F-score 0.70, NMI 0.55 and TOP2 0.71. The smaller clustering size is due to the more homogenous nature of the two "halves" of the minimum cut. Nonetheless, gangplanks will tendentially always split a graph; in this case, we have three clusters pertaining to *euonymus*:

{fern,cranberry,birch,shrubbery,cypress,bougainvillea,...},
{yam,apple,rose,tree,leaf,baguette},
{lawn,abelium,multiflora,buckthorn,clethra,...}.

and two to *carryall*:

{gilt,PDA,diamond,apartment,style,Cruiser,duffle,...},
{restraint,pouch,gear,rucksack,binder,flask,...}.

We deem the smaller recall coming from their relative balance the cause of worse scores with respect to the regular algorithm.

The hyperclustering is compressed in two clusters which clearly and mostly correctly correspond to the two pseudosenses, and the scores improve accordingly but, since its quality depends on the quality of its building blocks, they remain still slightly inferior than for the regular gangplank version: BCubed F-score 0.90, NMI 0.73 and TOP2 0.88.

We could conclude this parenthesis observing that for our task, the simulation of homonymy detection for disemous words, the inclusion of the minimum cut as a step prior to the actual clustering will mostly not improve performances significantly. Applied to the particular case of ego

graphs of disemous pseudowords, this variant forces an algorithm to produce at least two distinct clusters (the two connected components resulting from the cut). If, in the best scenario, the minimum cut already identifies the two ground truth clusters α and β, the clustering algorithm should not split the two connected components any further. In a sense, the minimum cut variant puts to a test the fragmentary nature of the given clustering algorithm. If a clustering algorithm has a penchant for splitting a set, it will obtain probably more precise clusters on the two halves of the cut, with the risk of them being smaller and with less recall, while a coarse-grained clustering algorithm like CW will not need it. In a less bipolarized situation (a word with more than two possible senses), however, a minimum cut might be useful (although computationally more expensive) as a sort of pre-processing step to improve the purity of very blurred clusters.

The Markov cluster algorithm with parameters (2,2) returns a more fragmented result, with 11 clusters. Some of them appear to be very specific, like

{bike,humvee, skateboard, car, Cruiser},

together with a pair of singletons like

{wheel} or
{look}.

We can still separate *euonymus* from *carryall*, but recall penalizes the scores and we have a BCubed F-score of 0.73, a NMI of 0.63 and a TOP2 of 0.84: in this case higher than GP, but worse than GP's hyperclustering.

When using 1.4 as the inflation parameter, though, MCL nearly acts as the hyperclustering of the version with inflation parameter 2. We retrieve just 2 clusters where the two pseudosenses are clearly distinct, and we obtain extremely high scores: BCubed F-score 0.97, NMI 0.9, TOP2 0.92. Its hyperclustering stays unvaried.

MaxMax takes the tendency of the more fragmented MCL version even further and produces 21 clusters, some of them consisting only of two or three terms, like

{pack, equip} and
{style, lash, look},

and overall with very restricted meaning areas. The biggest cluster comprises 56 elements. Despite the very high precision, its scores are the lowest ones between basic clusterings, with a BCubed F-score of 0.34, a NMI of 0.48 and a TOP2 of 0.47. In denser networks its clustering tends to be even more dispersed. However, the hyperclustering causes here a major improvement. The clustering are condensed in 3 clusters, with the *carryall* sense split among one bigger and one smaller cluster:

{humvee,bike,car,cart,Cruiser,skateboard,...},
{lawn,eraser,moss,brush,reed,grass,wedge,...} and
{eyeglass,overalls,cap,parka,sweater,...}.

This new clustering reaches a BCubed F-score of 0.62, a NMI of 0.37 and a TOP2 of 0.66. It is remarkable how normalized mutual information unpredictably decreases, despite the evident overall better look. MaxMax' hyperclustering is capable of achieving even better improvements than for *euonymus_carryall*.

Between the two aggregative clustering variants, one with variable radius (AGG75P) and the other with a fixed radius of 0.97 (AGG97), the latter has a more dispersed clustering, with 6 clusters against the 3 found by the former. This is explained by the fact that the ego graph of *euonymus_carryall* is very balanced, so that distances are more homogenous, producing a very high 75th percentile. The clusters of AGG97 are quite balanced and the TOP2 score is still quite good: 0.72, compared to a BCubed F-score of 0.63 and a NMI of 0.67. Precision is good, but the pseudosenses are split among the clusters. AGG75P is more focused, with only one smaller spurious cluster pertaining to *carryall*:

{humvee,bike,car,cart,Cruiser,skateboard,...},
{lawn,eraser,moss,brush,reed,grass,wedge,...} and
{eyeglass,overalls,cap,parka,sweater,...}.

This variant thus obtains indeed much better results: BCubed F-score 0.93, NMI 0.82, TOP2 0.90.

The two respective hyperclusterings also behave differently: AGG97 merges everything in one big cluster, because of its high fixed radius, while AGG75P, by its own definition, will always find at least two different clusters in a hypergraph with at least three nodes and different weights on its edges.

The curvature-based clustering algorithm identifies 9 clusters. Their distribution is very skewed: there are two bigger clusters with respectively 283 and 29 terms:

{equip,apple,twig,style,lunchbox,toothpick,pachysandra,...},
{nandina,brake,wheel,rosebush,poppy,...}.

The rest is subdivided among the remaining miniclusters. The clusters appear quite confused, and scores are consequently: BCubed F-score 0.61, NMI 0.06, TOP2 0.42. The scores are substantially unvaried for the hyperclustering, since the only change is that two minor clusters are merged, namely:

{cultivar,gizmo,daylily,weed,boxwood,hibiscus} to
{rucksack,restraint,leaf,brooch,tree,bulb,button-downs}.

5.4.4 Conclusions

Even if we observe some clustering algorithms excelling at some evaluation measure more than others, in the end we are effectively not able to proclaim one of them as the "overall best WSI clustering algorithm". However, the analysis on both the semantic-similarity and on the co-occurrence-based ego graph data sets of the algorithms' behaviours in Section 5.4.2 (and in Appendix B), and the insight into the nature of their clusterings given in Section 5.4.3, allow us to make some final considerations about the functioning and the desired properties of a clustering algorithm for the specific task we have chosen to study.

First, we want again to address the possible bias towards Chinese Whispers in our evaluation framework, already discussed in Section 5.4.2. We have observed that CW is on average the most coarse-grained among our systems. We acknowledge that coarse-grainedness is an advantage in our evaluation framework: In fact, in the ideal case we want an algorithm to produce exactly two clusters, and the TOP2 score penalizes outputs which deviate too much from this ideal number. Therefore, it is true that Chinese Whispers manages to achieve very good TOP2 scores, albeit at the expense of stability: the variance of its scores, as shown e.g. in Figure B.2 in Appendix B, is the highest in our evaluation framework. In Appendix B we refer to this fact as the *swim or sink* property of coarse algorithms: very often they ignore smaller or not well-represented differences and conflate all elements into just one cluster.

On the other hand, more fine-grained clustering algorithms might be more sensitive to minor subregions in a word graph and be stabler in terms of score variance, but their lack of focus and more dispersed clusterings give them inferior average scores; the best example for this is MaxMax. Considering the semantic-similarity-based ego graphs, the curvature-based clustering algorithm is also stably floating around low scores. However,

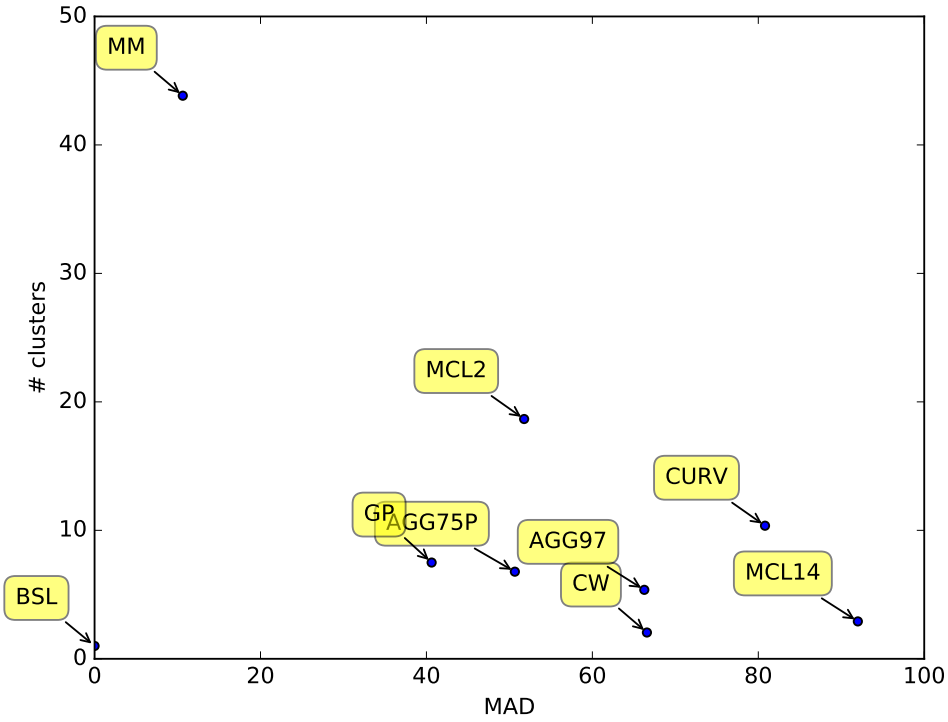
we recognize somewhat different splitting behaviours that we want to summarize using the *mean absolute deviation* (MAD), a measure of dispersion (see Section 2.1.9.2), together with the mean number of clusters (Table 5.4) in a clustering. We apply the MAD to the cardinalities of the clusters in a given clustering, and for each algorithm we take the mean of the MAD scores of all its clusterings on either data set[15]. The mean absolute deviation tells us how unbalanced the distribution of elements in a clustering is with respect to a hypothetical uniform clustering where all clusters are of equal size. The higher the MAD, the more skewed the clustering: this means that there will be few very big clusters counterposed to many smaller clusters. In Table 5.11 we show the MAD values for our algorithms on both data sets. The mean number of clusters, instead, tells us how fragmented a clustering is. Figure 5.7 graphically shows the position of our algorithms with respect to these two statistics, which we will comment comparing them to the respective TOP2 scores.

	Similarities	Co-occurrences
MCL2	51.8	13.0
MCL14	92.0	54.6
CW	66.6	160.5
MM	10.6	12.4
AGG97	66.3	59.4
AGG75P	50.7	69.6
CURV	80.8	90.8
GP	40.6	12.2
BSL	0	0

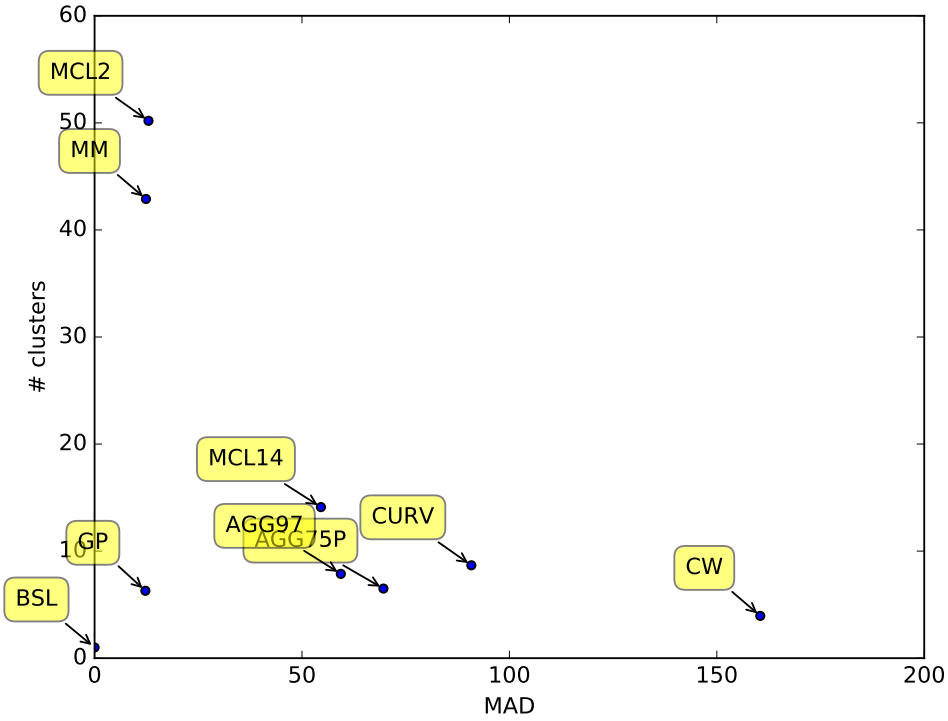
Table 5.11: Average values of the mean absolute deviations for each clustering algorithm over both ego graph data sets.

We observe for example that CURV, though only relatively fine-grained, has quite high mean absolute deviation values, much higher than Max-Max, which however suffers from an extremely high fragmentation. On the contrary, GP is more balanced, because, although having a higher MAD, it is less fine-grained, which lends to its clustering more compactness than the two other aforementioned systems. However, at the same time, It is exactly this same compactness that is detrimental to its TOP2 scores, since the recall of its representative clusters will be relatively low: We know that in most cases of homonymy one term is prevalent, as discussed in Section 1.1.1.

[15]We could have normalized the MAD score with respect to number of total clustered elements. However, since the order of our ego graphs is nearly constant, we left the absolute mean deviations. The same goes for the mean number of clusters.



(a) Similarity-based data set



(b) Co-occurrence-based data set

Figure 5.7: Distribution of the clustering algorithms with respect to mean absolute deviation (MAD, x-axis) on the cluster size distribution in their clusterings and mean number of produced clusters (y-axis).

We claim that the best clustering shape that can be obtained by an algorithm for the task of homonymy detection, and more generally of Word Sense Induction, is one that admits a pronounced, even if not exasperated, degree of skewness without being too much fragmented, i.e. with too many clusters. In the charts of Figure 5.7, this would correspond to a rather undefined region towards their bottom center-right quarter, where the best systems tend to concentrate (compare Tables 5.2 and 5.3), especially in Figure 5.7b (AGG75P, AGG97, MCL14 and CURV). As a matter of fact, CW's performances suffer on the co-occurrence-based ego graph data set (Figure 5.7b) because its good shape seen on the semantic-similarity-based data set is lost, or, in other terms, because while its mean clustering size remains stably low, its MAD value gets too high.

An interesting factor emerges from hyperclustering: we might call it the *scalability*, or conversely *rigidity*, of a clustering algorithm. Namely, we already noticed how MaxMax's hyperclustering markedly improves with respect to its basic clustering. At the same time, we observe that MaxMax's hyperclustering greatly rescales the size of the basic clustering. On the contrary, the basic clusterings of MCL2 and especially of the curvature-based algorithm are largely unaffected by hyperclustering. This hints to different properties of the algorithms, particularly when noting that MCL2 is as fragmented as MM on the co-occurrence-based data set (cf. Table 5.10). In the notation of Section 5.3.2, we express this ratio as

$$\zeta = \frac{|\mathcal{C}|}{|\mathrm{Hyp}(\mathcal{C})|} \geq 1 \tag{5.11}$$

and summarize its values for each algorithm in Table 5.12.

	Similarities	Co-occurrences
MCL2	2.5	1.3
MCL14	1.9	1.7
CW	1.8	3.5
MM	9.3	5.6
AGG97	5.3	4.3
AGG75P	3.8	3.9
CURV	**1.1**	**1.1**
GP	3.1	2.6

Table 5.12: Mean ratios of the number of clusters in a basic clustering to the number in its hyperclustering, for each algorithm, over both ego graph data sets. The lower ratios for each category are highlighted.

We already mentioned this behaviour in Section 5.4.2. We can say that the closer to 1 is ζ, the greater the *invariance* of an algorithm with respect to hyperclustering. This invariance is surely tied to coarseness, albeit it does not directly depend on it. We might claim that clustering algorithms with low invariance and a relatively great fragmentation, like MM or GP, benefit the most from hyperclustering. This is due to fact that the algorithm is capable of *rescaling* itself and efficiently use its relatively small and limited basic clusters as building blocks for more significant clusters.

Conversely, we recognize a generic turning point in terms of coarseness beyond which hyperclustering ceases to be useful: When a clustering is already coarse enough, i.e. close to the supremum of the partition lattice (see Section 5.3.2), the hyperclustering precipitates this coarseness, lowering its quality and the associated scores. Chinese Whispers is a clear example thereof: on the semantic-similarity-based ego graph data set (compare Table 5.9) its hyperclustering very often, if not always coincides with the trivial baseline, and indeed its scores drastically reduce, especially for the more balanced pseudowords.

The type of an ego graph, either based on semantic similarities or on co-occurrences, influences the values of the mean absolute deviation and to a much lesser extent the ratio ζ defined by (5.11): different clustering algorithms react differently to first-order or second-order relations. As we have stressed many times throughout the analysis of the previous sections, co-occurrence-based word graphs represent more unpredictable, generally less significant connections between words than semantic-similarity-based word graphs. They are also more subject to noise in the form of words with little relevance to WSI (cf. Table 4.1) or even punctuation, depending on the pre-processing that was performed on the original raw text. Co-occurrence word graphs in our data set are sensibly denser than their semantic-similarity-based counterparts and their small-world nature (see Section 2.1.7) is more pronounced, and this is a cause of generally more fragmented and more blurred clusters, corresponding to overall lower scores. In such a setting, we can distinguish a disadvantage for clustering algorithms that rely on the abstract concept of *information flow*, like the MCL variants or CW, compared to others that are based on distance, like the AGG variants, or that just check some local properties, like MM or CURV, especially for the more balanced pseudowords (compare Tables 5.7 and 5.8). We see a reason for this in the fact that co-occurrences have a local essence, and that we can not as easily associate them semantic reasonings as with semantic similarities, as an information flow implies, but we rather observe that the emerging syntactical relations (see also Section 1.1.1 and Section 2.1.8) possess a different nature.

Chapter 6

Conclusions

The goal of the present dissertation is to describe various graph-based approaches to Word Sense Induction and to carry out a comparison between them.

In the first part of the dissertation, consisting of Chapters 1, 2 and 3, the task of Word Sense Induction (WSI) as the unsupervised counterpart of Word Sense Disambiguation is laid out in its theoretical foundations. The problem of distinguishing the possible, different senses that a given word can assume in different contexts originates from intrinsic properties of language and human thinking; the issues that we encounter in this task have sparked large and still-lasting debates in various fields, primarily linguistics and philosophy, with ramifications also in the broader spectrum of cognitive sciences, like psychology or even neuroscience. The notions of *word* and *sense* are complex, and there is not always consensus about their definitions. Different perspectives about such matters have given rise to different approaches to WSI even in the area of Computer Science, where some kind of compromise has to be reached between the theoretical discussion and the actual implementation of automated systems that identify the possible meanings of string tokens in a text. In this regard, in Chapter 1 we make a more subtle distinction between two kinds of Word Sense Induction tasks: Word Sense Induction proper, concerned with the detection and implicit definition of the senses of a word, and Word Sense Discrimination, oriented towards the labelling of a word's occurrence with one out of some given senses. In both cases, such operations need and are based on the modelling of word contexts. We recognized two main paradigms in literature when coming to this: models using vector spaces to represent semantics, and graph-based models. We deem graphs to be particularly suited to the direct representation of relationships between words and to the interpretation of a word as an atomic unit which is part of many different local substructures (semantic, syntactical, or of other kinds), and in

this work we focus on graph-based approaches to WSI. We remark that, independently from its concrete implementation, Word Sense Induction principally relies on clustering a data set, with the assumption that each cluster implicitly defines a sense of the examined word.

Chapter 3 more concretely investigates how graph-based approaches to WSI have been implemented in literature so far, by the examples of five systems: the Markov cluster algorithm [van Dongen, 2000] (MCL), originally developed in the field of biology but of general purpose; the approach by [Dorow and Widdows, 2003], based on the detection of particular syntactical patterns that put words into reciprocal relation and making use of MCL to perform the clustering step; Chinese Whispers [Biemann, 2006], which simulates the flow of information through the graph to determine the "sense class" of each node; HyperLex [Véronis, 2004], which defines a significance measure to evaluate the co-occurrence of words and exploits the semantic dominance of some nodes over the others; and finally MaxMax [Hope and Keller, 2013], identifying a word's possible senses with directed subgraphs possessing particular properties. We established two core principles that are common to most of the aforementioned clustering algorithms, and to graph-based WSI approaches in general: first, after the global document has been modelled as a word graph, the senses of a given word are induced with respect to a local subgraph, which acts as a neighbourhood of that word; second, the detection of the sense clusters hinges on the assumption that there are "denser" regions in the subgraph, semantically interpretable as groups of terms revolving around a common meaning. This expectation is justified by the nature of word graphs. Since these are structures that model a natural phenomenon like language, their characteristics are much different from those of random graphs, and have actually been shown to have small-world properties and also to be scale-free (see Section 2.1.7). Besides the different angles from which the examined approaches try to deal with this fact, we notice a general tendency of focusing on the word graph building step rather than on the final clustering and sense-inducing step. Further, many algorithms rely on parameters and thresholds, which are by definition arbitrary constants that characterize the execution and determine the outcome of an algorithm. We see a basic contradiction: the notion of tuning a parameter to obtain optimal results implies reasonings that stray from the unsupervised paradigm of Word Sense Induction and involve external knowledge. The same holds for the concept of thresholds, which are used to shape the results of a clustering by setting arbitrary or heuristically determined limits. What we are arguing here is that such parameters and thresholds represent supervised elements that are imposed on the structure of the data to be clustered, whereas we would expect a pure unsupervised approach to discover patterns by tapping exclusively from such structure. Instead, very often the

introduction of parameters can be seen just as an unintended shift from an *a priori* supervised set up to an *a posteriori* supervised refinement. These considerations are the basic motivations that brought us to present, in the second part of this work, definitions and instruments which allow to create a cohesive framework for WSI where text pre-processing is kept to a minimum, and clustering algorithms let a natural partition of the word graph arise just by exploiting its structure, not by forcing one on it. Also,we have the goal to cope with the already mentioned peculiar small-world structure of word-graphs, which in our opinion greatly impacts the outcomes of clustering algorithms.

The main contribution of Chapter 4, which presents and greatly expands the material of [Cecchini et al., 2015] and [Cecchini and Fersini, 2015], is the definition and exploration of mathematical instruments that we deem useful for the analysis of graphs, especially in the field of WSI. The *weighted Jaccard distance* (together with its more straightforward unweighted counterpart) of Section 4.1.1 is an adaptation and a generalization of the Jaccard index to the node set of an (undirected) weighted graph. Our aim is to obtain a second-order relation that represents the distance between the contexts of two words on the basis of first-order relations, e.g. simple co-occurrences with their frequencies, which are possibly modelled by a word graph. To this end we treat closed first-degree neighbourhoods of the nodes as multisets, to take into account the weighted structure of the graph beyond its mere topology. The weighted Jaccard distance enables us to abstract a word metric space from a word graph and to perform the clustering and sense-inducing step from there, deducing both syntagmatic and paradigmatic information from the text. We have provided many examples and minor results that prove the desirable properties of our weighted Jaccard distance. In addition, we define *gangplank edges*, that is, edges that represent a weak, random connections in a word graph and that are thus used to outline denser regions of the graph. Finally, the observation that weighted and unweighted Jaccard distances delineate different structure levels of the word graph leads us to give a novel synthetic definition of *curvature* for a weighted graph, adapting the classical geometrical notion to the case of a discrete space. Our distance-based curvature allows to reveal denser subregions of the graph. We also discuss the implementation and time complexities of all defined instruments.

The weighted Jaccard distance, gangplank edges and the definition of curvature are at the base of the three novel clustering algorithms for WSI introduced in this dissertation. We describe and discuss both their generic functioning and their principles, and their actual implementation in our Word Sense Induction framework, going from the raw text pre-processing

step to the induction and later the discrimination of word senses. The different parts of this pipeline and the involved techniques (like the computation of Jaccard distances) form a series of modules that can be combined in a flexible way or included in already existing WSI systems.

The three algorithms are the *gangplank clustering algorithm*, an *aggregative clustering algorithm* and a *curvature-based clustering algorithm*. The first one exploits the second-order relations obtained by the weighted Jaccard distance on a word graph and employs the gangplank edges to identify the borders between subgraphs possessed of significant intraconnections and thereby defining word senses. The aggregative approach is a variant of a k-medoid clustering algorithm on the metric word space derived from a word graph by means of the weighted Jaccard distance, and applies a prior initialization step to assess the number of clusters and to ascertain the medoids that will act as their seeds. The curvature-based approach identifies the senses of a word with the subgraphs of that word's local word graph induced by node couples with positive (spherical) curvature. The logic which is common to all these three methods is that they focus on the properties of edges or more generally node couples, which correspond either to first-order or second-order relations between words; such properties are used to detect regions of the word graph that are denoted by strong, significant internal connections.

The contribution of Chapter 5 is to provide two novel data sets of word ego graphs based on *pseudowords*. A pseudoword is the artificial conflation of two existing words: this construct simulates a case of homonymy and polysemy in a controlled setting and allows to perform evaluation and comparisons of different graph-based WSI clustering algorithms with perfect knowledge of the ground truth. Each one of our two data sets consists of 1225 word ego graphs, each relative to one of the possible combinations of 50 selected monosemous words and thus acting as the word graph of a disemous (i.e. with exactly two senses) word. One data set represents relations between words by means of semantic similarities computed through the lexicographer's mutual information (Section 2.2.1.3), and the other one by means of simple sentence co-occurrences. The monosemous components of the pseudowords and the contexts needed to create the ego graphs are extracted from a large database containing more than 100 million sentences in English. To our knowledge, the pseudoword data sets presented here are the largest ones of their kind.

We use the two data sets as the backbone of our proposed *pseudoword evaluation framework*. In this context, we perform a thorough analysis of the properties of the pseudowords, figuring out the dynamics that dictate the structure of a pseudoword ego graph and how these interact with the mechanisms of a clustering algorithm. Specifically, on our data sets we make a reciprocal comparisons of the results of three algorithms from

literature (MCL, Chinese Whispers and MaxMax) and of our three proposed algorithms (gangplanks, aggregative and curvature-based), and in the process we devise two new instruments: a *hyperclustering* step (Section 5.3.2), which recursively condenses an already existing clustering and helps recompact a fragmented one; and the TOP2 evaluation score (Section 5.2.3), which we use alongside NMI and BCubed measures and which we developed expressively to evaluate the task of homonymy detection for disemous words.

The investigation of how the parameters of a pseudoword affect an algorithm's outcomes (Section 5.3.1 and Appendix B), the comparison of the scores obtained by different evaluation metrics together with the detection of their biases (see Section 5.4.1.1), the sizes of the clusterings and the trends put in evidence by the hyperclustering step (Sections 5.4.2 and 5.4.4), the influence of the type of a word graph (based on semantic similarities or co-occurrences) on the output of an algorithm - all these factors, preceded by the comprehensive description of the task and the definition of novel concepts and instruments to tackle it, concur to give a deeper insight into the functioning and even on the pitfalls of graph-based Word Sense Induction. Namely, in Section 5.4 we highlight and isolate the elements that determine how the results of an algorithm look like, discuss their properties and behaviours in relation to the word graph features and establish the pro and contra of each algorithm.

Most notably, different clustering algorithms react differently to first-order or second-order relations. In particular, the data set analysis in Sections 5.2.2.1 and 5.3.1 and in Appendix A shows how co-occurrence-based word graphs represent more unpredictable, generally less significant connections between words than word graphs based on semantic similarity. Co-occurrences are also more subject to noise in the form of words with little relevance to WSI or even punctuation, depending on the pre-processing that was performed on the original raw text (cf. Section 4.3.1). At the same time, co-occurrence word graphs in our data set are sensibly denser than their semantic-similarity-based counterparts and their small-world nature (cf. Section 2.1.7) is more pronounced, this being a cause of generally more fragmented and more blurred clusters, corresponding to overall lower scores. In such a setting, we can distinguish a disadvantage for clustering algorithms that rely on the abstract concept of *information flow*, like the MCL variants or Chinese Whispers (cf. Sections **??**), compared to others that are based on the concept of distance, like the AGG variants (cf. Section 4.2.2), or that just check some local properties, like MM (cf. Section 3.5) or CURV (cf. Section 4.2.3), especially for the more balanced pseudowords. We see a reason for this in the fact that co-occurrences have a local essence, and that we can not as easily associate them semantic reasonings (like those implied by an information flow) as for similarities, but we rather

observe that the emerging syntactical relations possess a different nature and are in a complementary syntagmatic versus paradigmatic relation (see Section 1.1.1 and Section 2.1.8), mirrored by the complementary relation between Word Sense Discrimination and Word Sense Induction. Even if they are often used as synonyms (as e.g. in [Navigli, 2009]), this dissertation puts in evidence the need to approach these tasks with different instruments.

As a final remark, we want to highlight the paramount importance of text pre-processing in the outcome of WSI clustering algorithms. This determinant step is often not given enough relevance, even if it greatly influences text modelizations, a fact that indirectly emerges from our work. In the light of this, besides believing that this connections needs further investigation, we think that it might be useful for WSI evaluation campaigns or frameworks to be performed not only on a shared data set, but also on the same given type of structure.

We envision different possible future developments and hooks for the material presented in this work. The first, obvious one is to expand the pseudoword data set, acquiring many more components to combine and refining the division in frequency classes, besides also considering additional weighting schemes for the ego graphs. In connection to this, we could perform a more detailed, graph theoretical analysis of the characteristics of the pseudoword ego graphs, with the aim to further investigate the representation of raw text and word contexts. Another immediate direction is the extension of our examined and proposed algorithms to the task of Word sense Discrimination, of which we only mention an implementation. Finally, from a more technical point of view, taking inspiration from hyperclustering as a recursive and self-contained way to produce new clusterings from existing ones, we conceive of exploring the implementation of *consensus clustering* (see e.g. [Goder and Filkov, 2008, Ghaemi et al., 2009]) for the algorithms we have taken into consideration, as a way to let the best features of each emerge. Of course, this will involve the further tuning of the synthetic distance-based notion of curvature for a graph and the extensive study of its theoretical and practical implications.

Appendix A

Analysis of collapsed pseudowords

Here we present some further comments and data about collapsed pseudowords, expanding the discussion of Section 5.2.2.1. As noted there, while some of the dominant or dominated components appear in pseudowords of both data sets, no pseudoword collapses both for semantic-similarity-based and co-occurrence-based ego-networks.

Each one of the component words listed in Section 5.3 contributes to 49 different pseudowords. Still, some components have a greater tendency to originate a collapsed pseudoword. In the case of similarity-based ego-networks, we observe that all 50 components occur in at least one collapsed pseudoword, from a maximum of 27 times for `courage` (frequency class 5) to a minimum of 1 for `heartache` (FC 4), `lightbulb` (FC 3) and `tequila` (FC 4). On the contrary, in co-occurrence-based pseudoword ego-networks only 39 component words are involved in a collapsed pseudoword: beer (FC 5) is the most present one (26 times). In the following tables we summarize the number of times each component participates in a collapsed pseudoword (Table A.1) and the presence of each frequency class (FC) in collapsed pseudowords (Table A.2).

Table A.2 confirms the observations about collapsed pseudowords of Section 5.2.2.1: the components that most destabilize a pseudoword's ego-network are mostly of the highest frequency class, and the victims of this process are the least frequent terms. Of course, only half (rounded down) of the involved components dominate their respective pseudoword ego-networks. Table A.3 highlights these components together with the ratio of dominated pseudowords to generic pseudowords they appear in, and their respective frequency classes.

Nr. collapsed pseudowords	Participating components: Similarities	Participating components: Co-occurrences
27	courage(5)	-
26	-	beer(5)
24	ailment(5)	clothing(5)
20	beer(5), psychologist(5)	-
19	clothing(5), credibility(5)	-
18	pillow(4), tofu(4)	-
17	southerly(2)	-
15	reliability(4)	-
14	dioxide(5), treasurer(5)	-
13	pneumococcus(1), ugandan(3)	-
12	ackee(1), bullfight(3), dimwit(2), hyperthyroidism(2)	ailment(5), barque(1), euonymus(1), tautog(1)
11	barman(2), bobolink(1), carryall(1), shawl(4), tautog(1)	courage(5), credibility(5), leatherette(1), psychologist(5)
10	barque(1), garment(5)	bobolink, bufflehead(1), carryall(1), garment(5), paycheck(5), pneumococcus(1)
9	pennywhistle(1)	catsup(2), dioxide(5), treasurer(5)
8	euonymus(1), leatherette(1), showtime(3), tailwind(3), tranquilliser(2)	tranquilliser(2)
7	artistry(4), bufflehead(1)	hyperthyroidism(2), reliability(4)
6	astrologer(3), catsup(2), philanderer(2)	-
5	curio(3), grosgrain(2), heartbreak(4), yellowcake(2)	-
4	marshmallow(4), wedlock(3)	grosgrain(2), pennywhistle(1)
3	flamboyance(3), hairpiece(3)	reptile(4)
2	afro(2), paycheck(5), reptile(4)	ackee(1), afro(2), barman(2), bullfight(3), curio(3), hairpiece(3), showtime(3), southerly(2), tailwind(3), ugandan(3)
1	heartache(4), lightbulb(3), tequila(4)	dimwit(2), flamboyance(3), philanderer(2), tequila(4)

Table A.1: Components involved in collapsed pseudowords by number of occurrences. Frequency classes are indicated for each component in parentheses.

FC of a component	Number of collapsed pseudowords	
	Similarities	Co-occurrences
5	23	22
4	6	2
3	4	2
2	5	3
1	12	10

Table A.2: Number of collapsed pseudowords with a component of a given frequency class.

Component (FC)	Ratio of collapsed pseudowords	
	Similarities	Co-occurrences
`courage` (5)	**0.55**	0.22
`ailment` (5)	0.49	0.24
`psychologist` (5)	0.41	0.22
`beer` (5)	0.41	**0.53**
`credibility` (5)	0.39	0.22
`clothing` (5)	0.39	0.49
`pillow` (4)	0.37	-
`tofu` (4)	0.37	-
`reliability` (4)	0.31	0.14
`dioxide` (5)	0.29	0.18
`treasurer` (5)	0.29	0.18
`shawl` (4)	0.22	-
`garment` (5)	0.20	0.20
`artistry` (4)	0.14	-
`heartbreak` (4)	0.08	-
`paycheck` (5)	0.04	0.20
`reptile` (4)	0.04	0.06
`tequila` (4)	0.02	-

Table A.3: Dominant components of collapsed pseudowords and percentages of pseudowords dominated by them. The components are ordered by their percentages in the semantic-similarity-based data set.

Indeed, all 10 terms in frequency class 5 appear as highly frequent dominant words both for similarities and co-occurrences; in both cases the six most predominant terms belong to that class. For similarity ego-networks, the only terms of frequency class 4 not appearing in the list are `heartache` and `marshmallow`. As collapsed pseudowords are rarer for co-occurrence-based ego-networks, in that case only 2 terms of frequency class 4 cause a collapse, which are maybe surprisingly not `pillow` or `tofu`, the strongest FC 4 terms for similarity-based ego-networks.

Appendix B

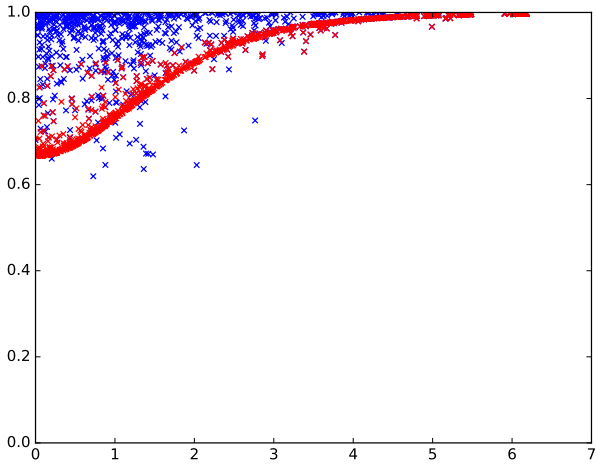
Pseudoword evaluation results with respect to word parameters

In this appendix we will comment the behaviours of the chosen clustering algorithms with respect to the parameters which define a pseudoword introduced in Section 5.3.1. We will not show every single result for every combination of data sets, algorithms, evaluation measures and parameters, as this would be too dispersive. Instead, we will focus on some selected cases to support the evidence already gathered with the analysis of the preceding sections.

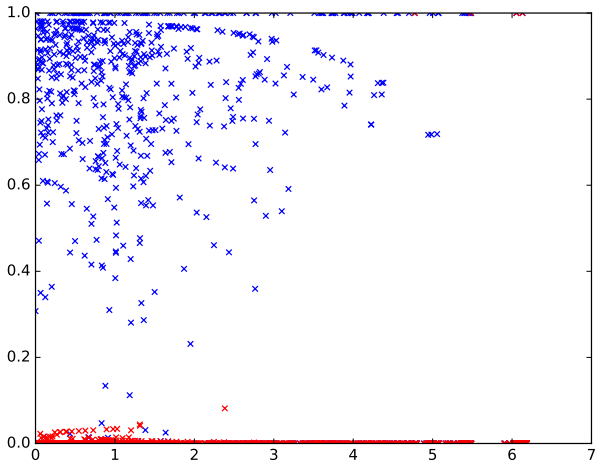
The following charts give a visual confirmation of the trends analyzed in Section 5.4.2. Tables B.1-B.2 and B.3-B.4 report, once in the form of a "point cloud" and once in the form of lines with error bars (showing means and variances over a 15-fold partition of the data set), the scores of respectively Chinese Whispers and MaxMax on the similarity-based ego-network data set, with respect to the balance parameter

$$\rho = \max(\frac{\alpha}{\beta}, \frac{\beta}{\alpha})$$

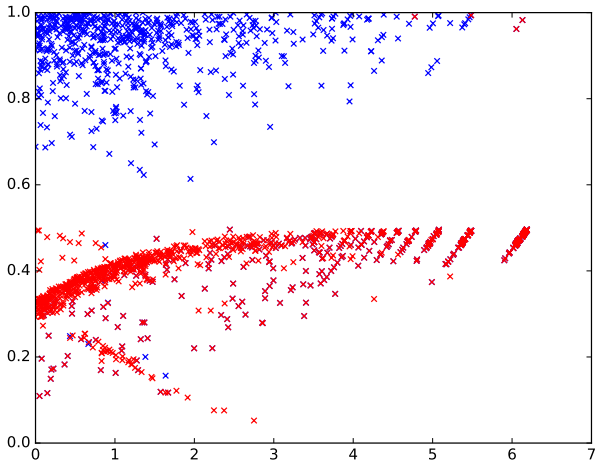
introduced in Section 5.3.1. Here we actually take the values $\log(\rho)$ to have a better scale in the graphs. Tables B.5-B.6 use the same parameter, but refer to the aggregative clustering algorithm with variable radius (at the 75th percentile of distance values; see Section 4.2.2) on the co-occurrence-based data set.



(a) BCubed F-measure

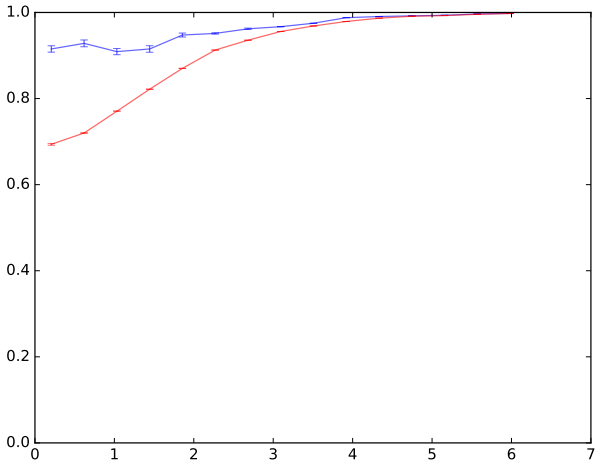


(b) NMI

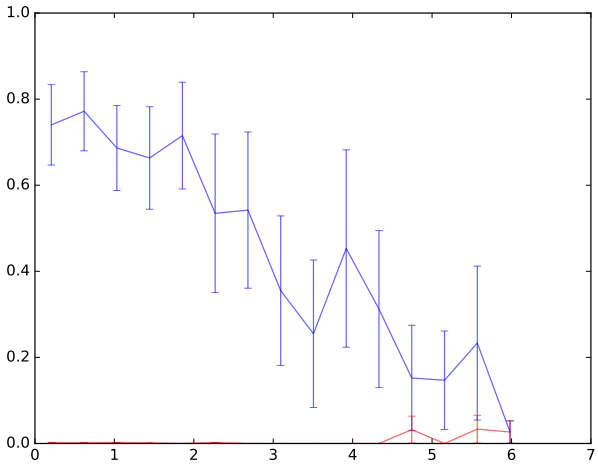


(c) TOP2 score

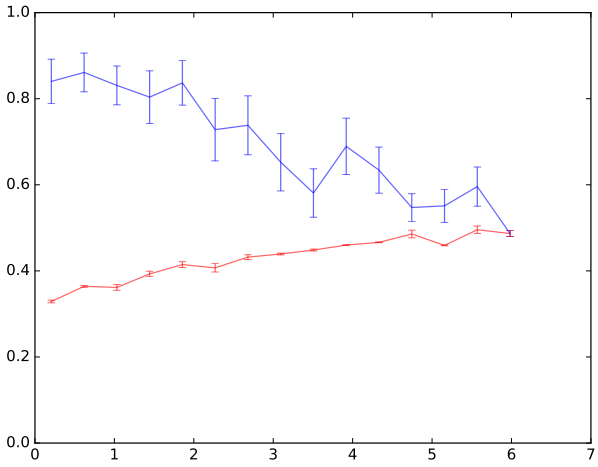
Figure B.1: Chinese Whispers scores (y-axis) as a scatter plot with respect to the logarithmized balance parameter ρ of the pseudowords (x-axis) on the similarity-based ego-network data set. The basic clustering is in blue, the hyperclustering in red.



(a) BCubed F-score

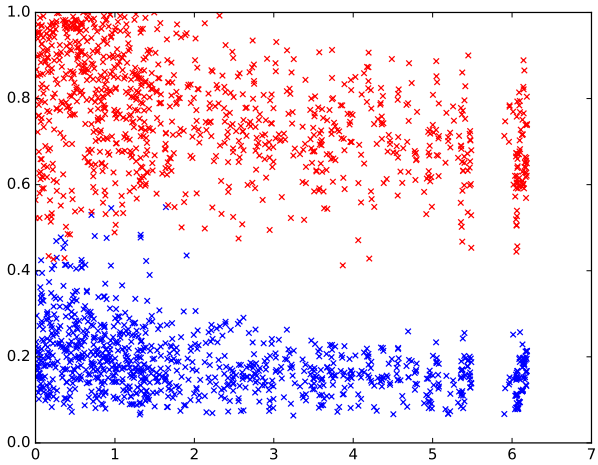


(b) NMI

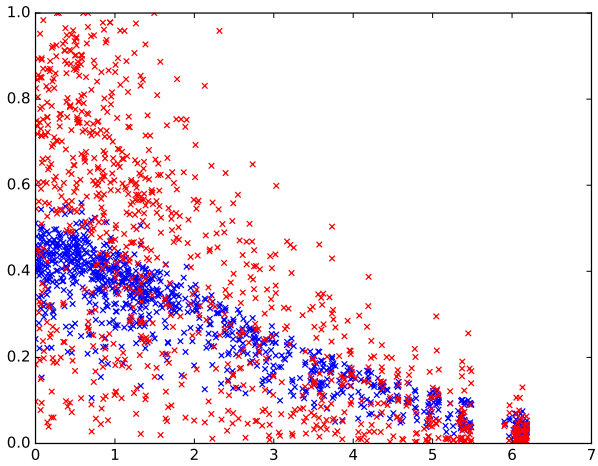


(c) TOP2 score

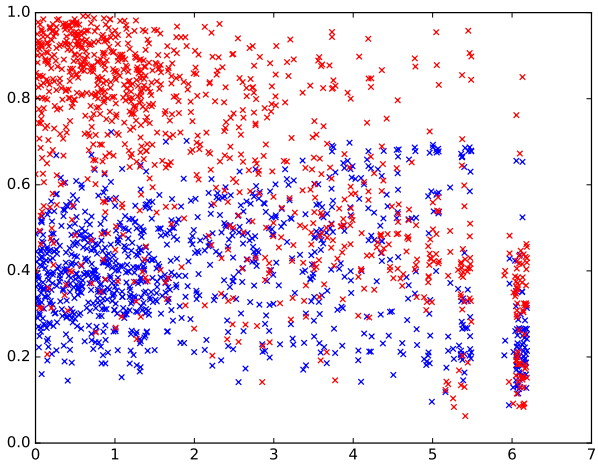
Figure B.2: Chinese Whispers scores (y-axis) as error bars with respect to the logarithmized balance ρ of the pseudowords (x-axis) on the similarity-based ego-network data set.



(a) BCubed F-score

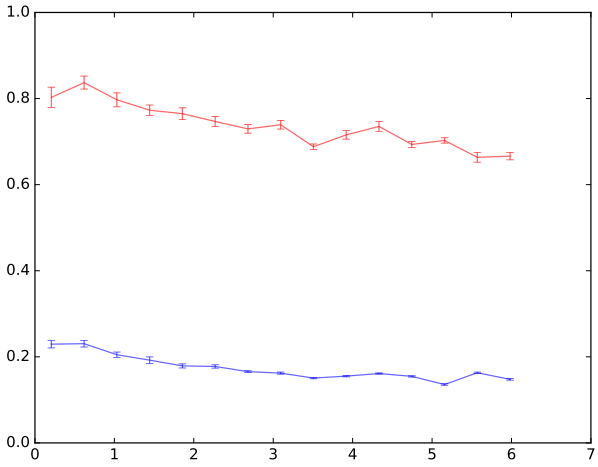


(b) NMI

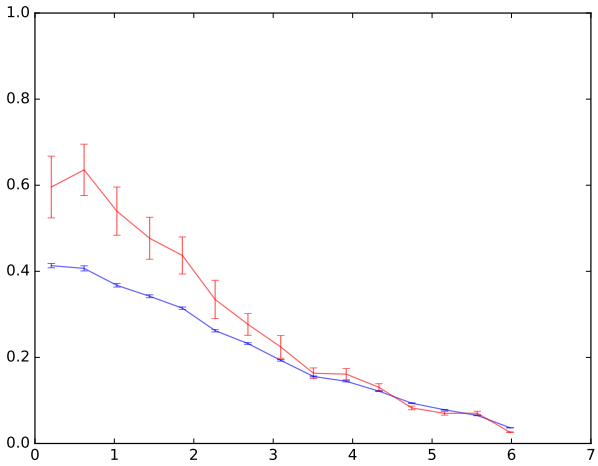


(c) TOP2 score

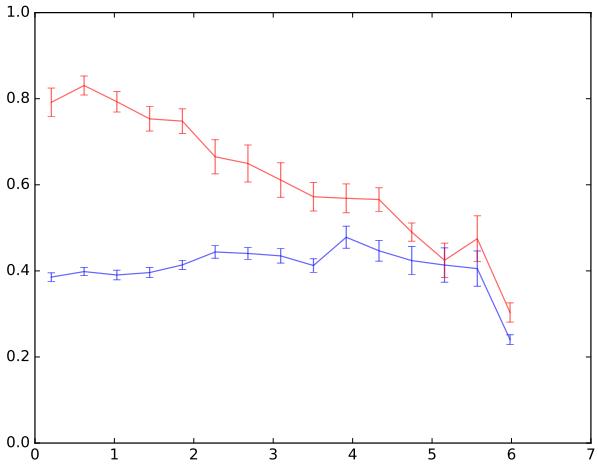
Figure B.3: MaxMax scores (y-axis) as a scatter plot with respect to the logarithmized balance ρ of the pseudowords (x-axis) on the similarity-based ego-network data set. The basic clustering is in blue, the hyperclustering in red.



(a) BCubed F-score



(b) NMI



(c) TOP2 score

Figure B.4: MaxMax scores (y-axis) as error bars with respect to the logarithmized balance ρ of the pseudowords (x-axis) on the similarity-based ego-network data set.

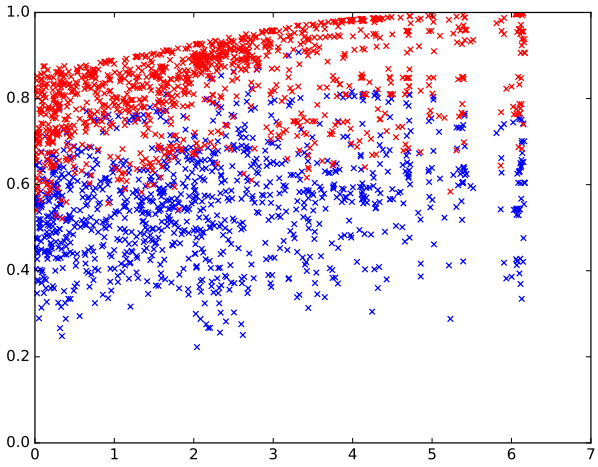
The greater ρ, the more unbalanced a pseudoword; this means that points and error bars on the left side of the charts refer to more balanced words than on the right side. For higher values of ρ pseudowords get quite sparser (also because of collapsed pseudowords), and we observe more variability on the right sides of the charts as well. This parallels the subdivisions of the Tables in Section 5.4.2 in balanced and unbalanced regions and the distribution shown in Table 5.3.

Tables B.1a-B.2a and less markedly B.5a-B.6a, relative to the BCubed F-score, are the only ones where we see an increase in the score as ρ also grows. However, due to the fact that the hyperclustering of Chinese Whispers is nearly always reduced to one single cluster, in B.1a-B.2a this trend actually corresponds to the trivial one-cluster-per-word baseline and follows the structural insensibility of BCubed measures discussed in Section 5.4.1.1. Otherwise, with respect to NMI and TOP2, we again acknowledge a general drop in performances tied to the unbalancedness between the components of a pseudoword on both data sets, while it is more evident for similarity-based ego-networks. A reason for this is that co-occurrence-based ego-networks are *noisier*: co-occurrences represent generally less significant bonds between words than semantic similarities, so that clustering algorithms are already influenced by this sort of inconsistence and the effect of unbalancedness has lesser impact on their overall performances.

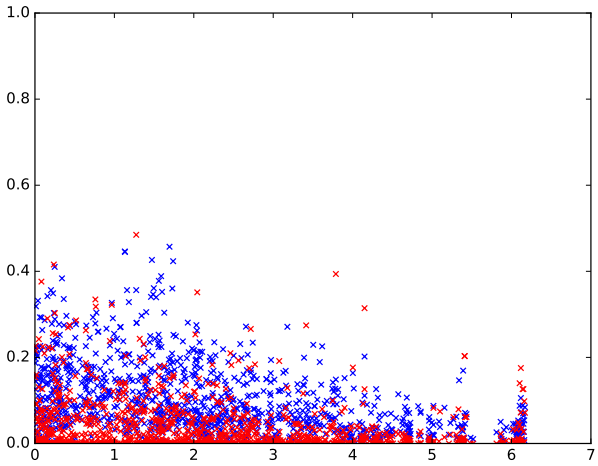
We notice that scores for basic clusterings and hyperclusterings are likely to converge as ρ grows, even if they diverge for lower values. This is because unbalanced pseudowords attract unbalanced, coarser clusterings, and so hyperclusterings often do not add a perceptible enough level of condensation. Finally, we note that Chinese Whispers, and tendentially coarse-grained clustering algorithms like MCL with parameters (2,1.4), show a greater variance in their scores than other systems. We could call this a *swim or sink* attitude of an algorithm: In the task we have chosen, many evaluation measures would reward very coarse clusterings, and we define the TOP2 score (see Section 5.2.3) exactly to lessen this effect and penalize trivial clusterings. This leads to more oscillating scores, where in one case a very coarse clustering is in line with the skewed composition of an ego-network, but in another case nearly coincides with the trivial baseline and is therefore penalized.

Tables B.7-B.8 and B.9-B.10 show the scores of respectively the gangplank and the curvature-based clustering algorithm, the former on the similarity-based and the latter on the co-occurrence-based data set, with respect to the relative overlap ratio

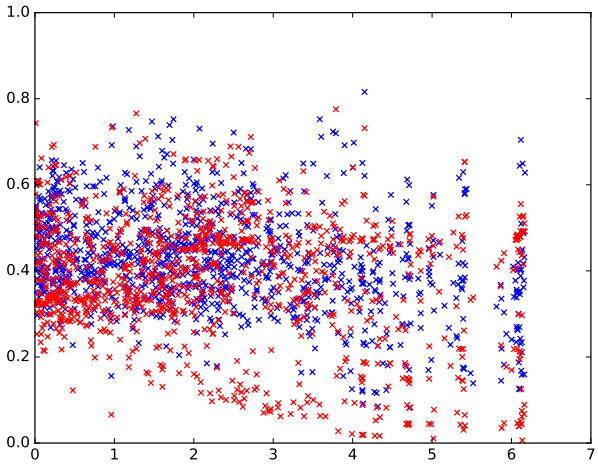
$$\kappa = \frac{|\gamma|}{|V|}$$



(a) BCubed F-score

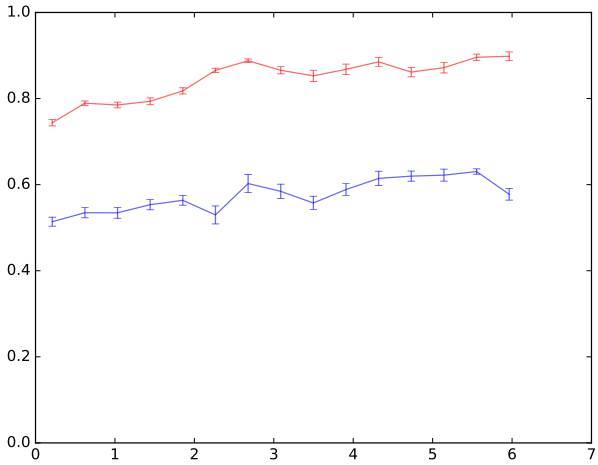


(b) NMI

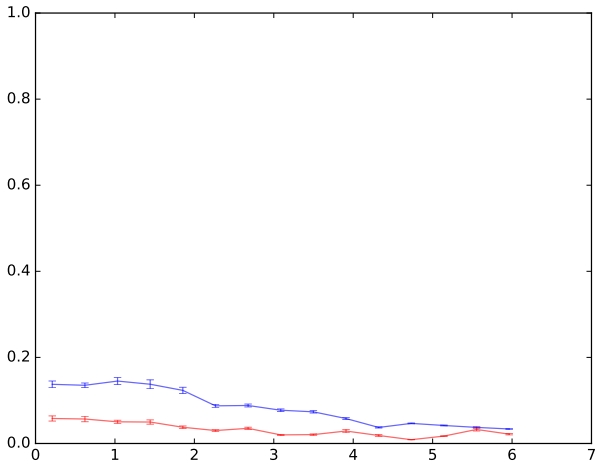


(c) TOP2 score

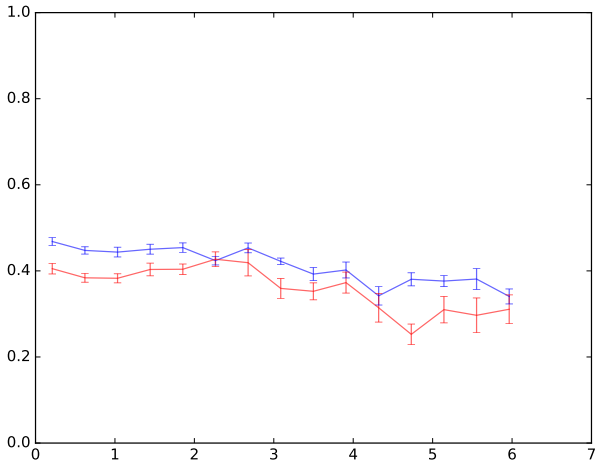
Figure B.5: Scores (y-axis) for the aggregative clustering with variable radius as a scatter plot with respect to the logarithmized balance ρ of the pseudowords (x-axis) on the co-occurrence-based ego-network data set. The basic clustering is in blue, the hyperclustering in red.



(a) BCubed F-score



(b) NMI

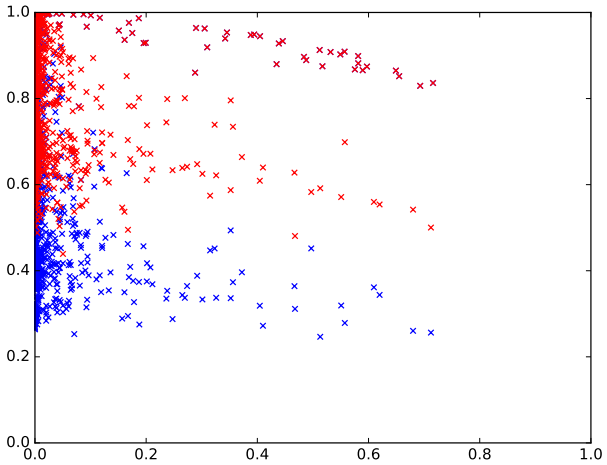


(c) TOP2 score

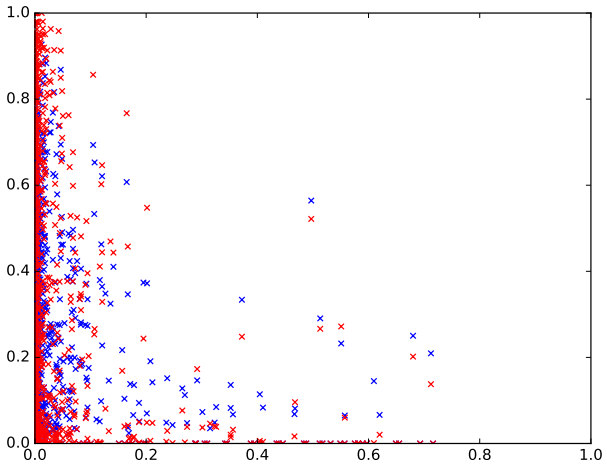
Figure B.6: Scores (y-axis) for the aggregative clustering with variable radius as error bars with respect to the logarithmized balance ρ of the pseudowords (x-axis) on the co-occurrence-based ego-network data set.

introduced in Section 5.3.1. The higher κ, the more common lexicon (based on the reduced distributional thesauri) is shared by the two components of a pseudoword. Analogously to ρ, pseudowords in both our data sets tend to have rather small overlaps, so that their distribution for higher values of κ is rather sparse, and sparser than for ρ. This behaviour is apparent from the denser concentration of points in the left half of the charts in B.7 and B.9. As already discussed, this concentration is slightly less pronounced in B.9, because in our co-occurrence data set words overlap more, due to frequent common lexical elements.

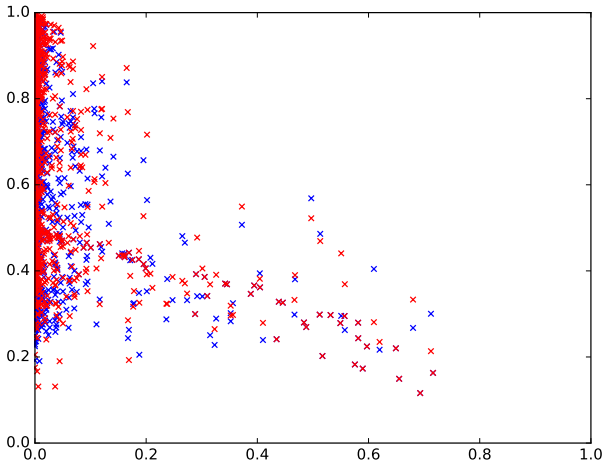
The general trend is comparable to what happened for ρ: all scores decrease as the relative overlap increases. The only possible exception can be again by part of the BCubed F-score, which rewards trivial, monocluster clusterings. All in all, the picture is nearly the same as for the balance parameter. The more irregular curve of the error bar plots and the rightmost spikes, like in B.10c, are not significantly interpretable and are caused by the marked sparsity of pseudowords in those regions. Still, in terms of TOP2 score we can claim that the greater the overlap, the more the basic clustering concords with its hyperclustering, up to the extreme case of the curvature-based clustering algorithm, where there is small to no difference. We look for the reason of this again in the more blurred structure of pseudowords with similar components, for which clusterings tend to be coarser and therefore the hyperclustering does not alter the pre-existing situation too much.



(a) BCubed F-score

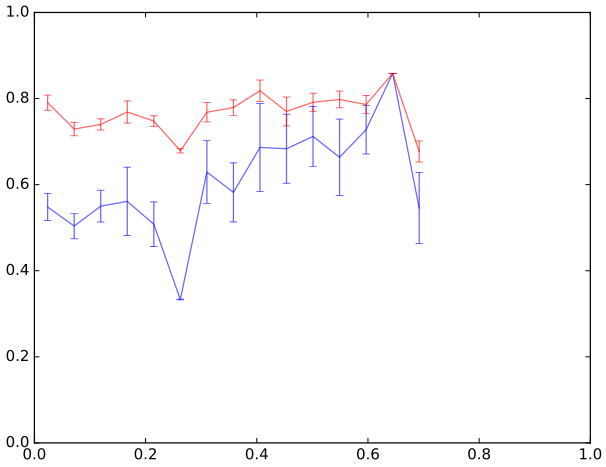


(b) NMI

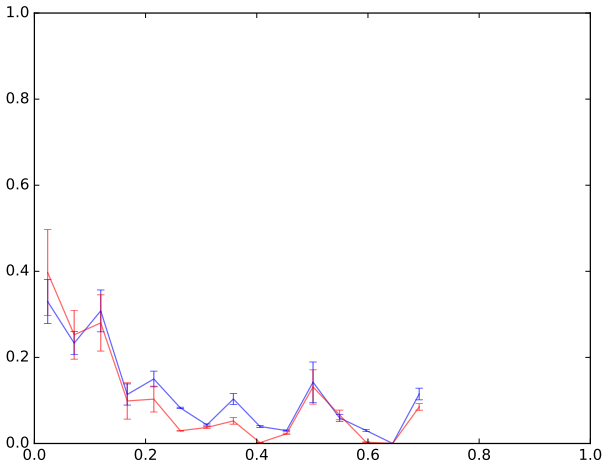


(c) TOP2 score

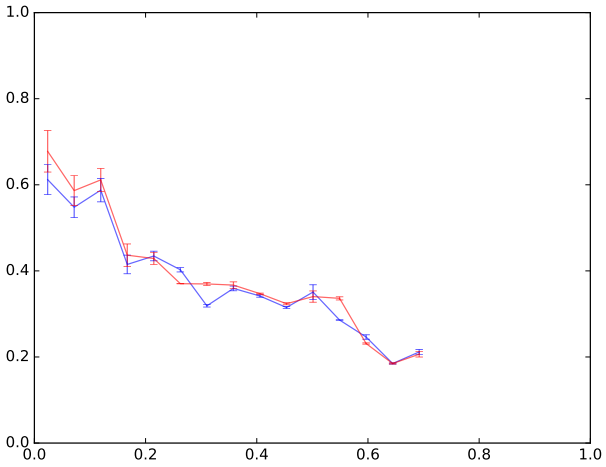
Figure B.7: Scores (y-axis) for the gangplank clustering as a scatter plot with respect to the relative overlap κ of the pseudowords (x-axis) on the similarity-based ego-network data set. The basic clustering is in blue, the hyperclustering in red.



(a) BCubed F-score

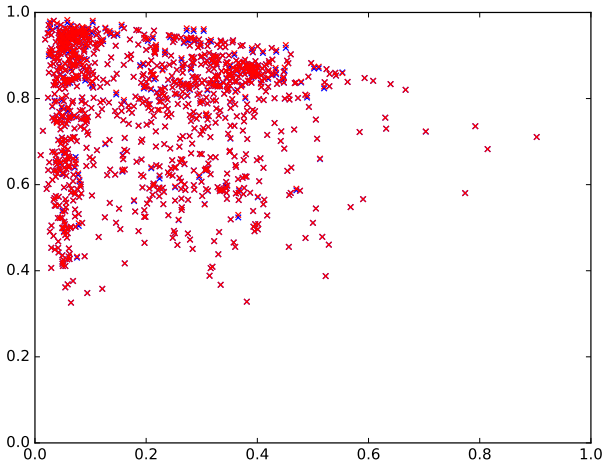


(b) NMI

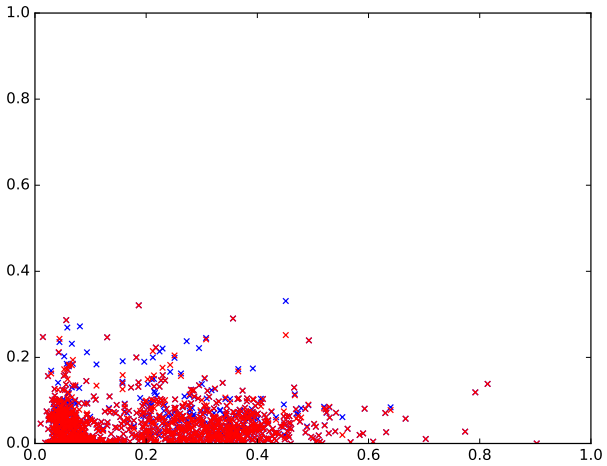


(c) TOP2 score

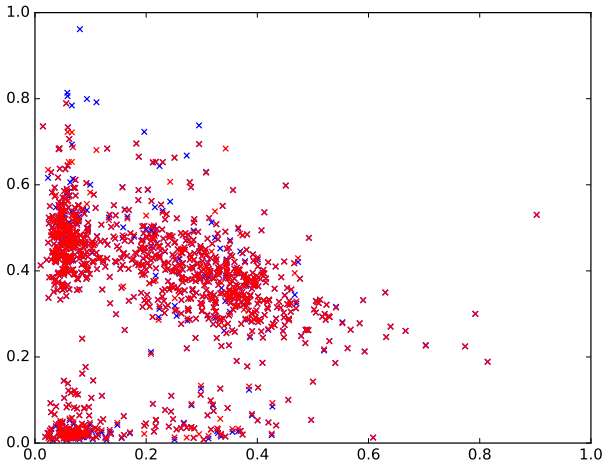
Figure B.8: Scores (y-axis) for the gangplank clustering as error bars with respect to the relative overlap κ of the pseudowords (x-axis) on the similarity-based ego-network data set.



(a) BCubed F-score

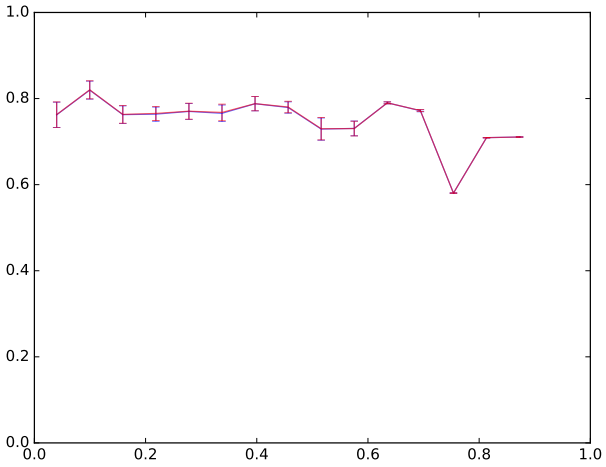


(b) NMI

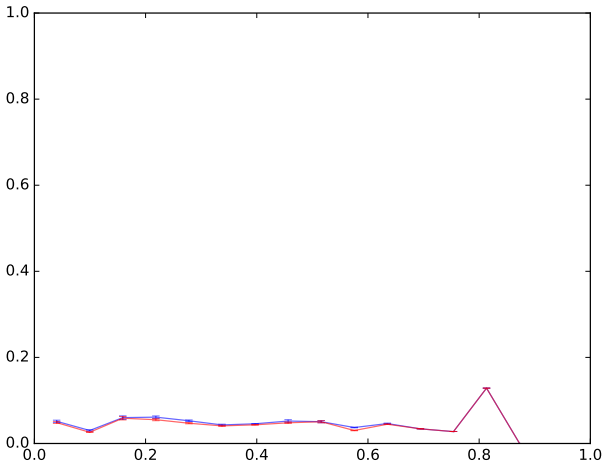


(c) TOP2 score

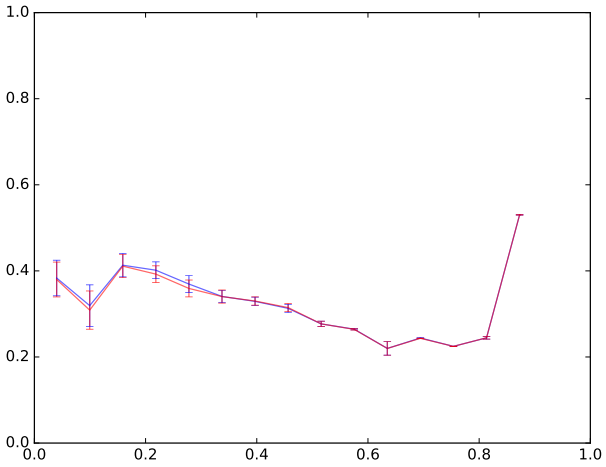
Figure B.9: Scores (y-axis) for the curvature-based clustering as a scatter plot with respect to the relative overlap κ of the pseudowords (x-axis) on the co-occurrence-based ego-network data set. The basic clustering is in blue, the hyperclustering in red.



(a) BCubed F-score



(b) NMI



(c) TOP2 score

Figure B.10: Scores (y-axis) for the curvature-based clustering as error bars with respect to the relative overlap κ of the pseudowords (x-axis) on the co-occurrence-based ego-network data set.

Bibliography

[Ábrego et al., 2009] Ábrego, B. M., Fernández-Merchant, S., Neubauer, M. G., and Watkins, W. (2009). Sum of squares of degrees in a graph. *Journal of Inequalities in Pure and Applied Mathematics*, 10(64):369–378.

[Agirre and Edmonds, 2007] Agirre, E. and Edmonds, P., editors (2007). *Word sense disambiguation: Algorithms and applications*, volume 33. Springer.

[Aigner, 2012] Aigner, M. (2012). *Combinatorial theory*, volume 234. Springer.

[Amigó et al., 2009] Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

[Arkhangel'skii et al., 2012] Arkhangel'skii, A., Fedorchuk, V., O'Shea, D., and Pontryagin, L. (2012). *General topology I*, volume 17. Springer.

[Artiles et al., 2009] Artiles, J., Amigó, E., and Gonzalo, J. (2009). The role of named entities in web people search. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, pages 534–542, Edinburgh, Scotland. Association for Computational Linguistics.

[Bagga and Baldwin, 1998] Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *Proceedings of the first international Conference on Language Resources and Evaluation (LREC'98), workshop on linguistic coreference*, pages 563–566, Granada, Spain. European Language Resources Association.

[Baker, 2003] Baker, M. C. (2003). *Lexical categories: Verbs, nouns and adjectives*, volume 102 of *Cambridge Studies in Linguistics*. Cambridge University Press.

[Barabási and Albert, 1999] Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

[Başkaya and Jurgens, 2016] Başkaya, O. and Jurgens, D. (2016). Semi-supervised learning with induced word senses for state of the art word sense disambiguation. *Journal of Artificial Intelligence Research*, 55:1025–1058.

[Bauckhage, 2015] Bauckhage, C. (2015). Numpy/scipy recipes for data science: k-medoids clustering. Technical report, Universität Bonn.

[Bell, 1934] Bell, E. T. (1934). Exponential numbers. *The American Mathematical Monthly*, 41(7):411–419.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Berge and Minieka, 1973] Berge, C. and Minieka, E. (1973). *Graphs and hypergraphs*, volume 7. North-Holland, Amsterdam, Netherlands.

[Bhogal et al., 2007] Bhogal, J., Macfarlane, A., and Smith, P. (2007). A review of ontology based query expansion. *Information processing & management*, 43(4):866–886.

[Biemann, 2006] Biemann, C. (2006). Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 73–80, New York, New York, USA.

[Biemann, 2007] Biemann, C. (2007). *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*. PhD thesis, Universität Leipzig.

[Biemann and Quasthoff, 2009] Biemann, C. and Quasthoff, U. (2009). Networks generated from natural language text. In *Dynamics on and of Complex Networks*, pages 167–185. Springer.

[Biemann and Riedl, 2013] Biemann, C. and Riedl, M. (2013). Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

[Bläser, 2013] Bläser, M. (2013). Fast matrix multiplication. *Theory of Computing, Graduate Surveys*, 5:1–60.

[Bollobás, 1998] Bollobás, B. (1998). Random graphs. In *Modern Graph Theory*, pages 215–252. Springer.

[Bollobás and Riordan, 2006] Bollobás, B. and Riordan, O. (2006). *Percolation*. Cambridge University Press.

[Bomze et al., 1999] Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999). The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer.

[Bordag, 2006] Bordag, S. (2006). Word sense induction: Triplet-based clustering and automatic evaluation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 137–144, Trento, Italy. EACL.

[Boyer and Merzbach, 1989] Boyer, C. B. and Merzbach, U. C. (2011 [1989]). *A history of mathematics*. John Wiley & Sons, Hoboken, New Jersey, USA.

[Brown et al., 1992] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Cameron, 2004] Cameron, P. J. (2004). Automorphisms of graphs. *Topics in algebraic graph theory*, 102:137–155.

[Carnap, 1948] Carnap, R. (1948). *Introduction to semantics*, volume 1042. Harvard University Press, Cambridge, Massachusetts, USA.

[Carpuat and Wu, 2007] Carpuat, M. and Wu, D. (2007). Improving statistical machine translation using word sense disambiguation. In *EMNLP-CoNLL*, volume 7, pages 61–72, Prague, Czech Republic. Association for Computational Linguistics.

[Cecchini and Fersini, 2015] Cecchini, F. M. and Fersini, E. (2015). Word sense discrimination: A gangplank algorithm. In *Proceedings of the Second Italian Conference on Computational Linguistics CLiC-it 2015*, pages 77–81, Trento, Italy.

[Cecchini et al., 2015] Cecchini, F. M., Fersini, E., and Messina, E. (2015). Word sense discrimination on tweets: A graph-based approach. In *KDIR 2015 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, volume 1, pages 138–146, Lisbon, Portugal. IC3K.

[Clark and Clark, 1977] Clark, H. H. and Clark, E. V. (1977). *Psychology and language. An introduction to psycholinguistics*. Harcourt, New York, New York, USA.

[Clarkson, 2006] Clarkson, K. L. (2006). Nearest-neighbor searching and metric space dimensions. In Shakhnarovich, G., Darrell, T., and Indyk, P., editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, Cambridge, Massachusetts, USA.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, Helsinki, Finland. ACM.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (2012 [1991]). *Elements of information theory*. John Wiley & Sons, Hoboken, New Jersey, USA.

[Dantzig and Fulkerson, 2003] Dantzig, G. and Fulkerson, D. R. (2003). On the max flow min cut theorem of networks. *Linear inequalities and related systems*, 38:225–231.

[Das, 2004] Das, K. C. (2004). Maximizing the sum of the squares of the degrees of a graph. *Discrete Mathematics*, 285(1):57–66.

[de Caen, 1998] de Caen, D. (1998). An upper bound on the sum of squares of degrees in a graph. *Discrete Mathematics*, 185(1):245–248.

[De Marneffe et al., 2006] De Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international Conference on Language Resources and Evaluation (LREC'06)*, number 2006, pages 449–454, Genoa, Italy. European Language Resources Association.

[De Saussure, 1916] De Saussure, F. (1995 [1916]). *Cours de linguistique générale*. Payot&Rivage, Paris, France. Critical edition of 1st 1916 edition.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

[Dixon and Massey Jr, 1957] Dixon, W. J. and Massey Jr, F. J. (1957). *Introduction to statistical analysis*. McGraw-Hill, New York, New York, USA.

[Dorow, 2006] Dorow, B. (2006). *A graph model for words and their meanings*. PhD thesis, Universität Stuttgart.

[Dorow and Widdows, 2003] Dorow, B. and Widdows, D. (2003). Discovering corpus-specific word senses. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, volume 2, pages 79–82, Budapest, Hungary. Association for Computational Linguistics.

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2012 [2001]). *Pattern classification*. John Wiley & Sons, Hoboken, New Jersey, USA.

[Dunning, 1993] Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

[Enright et al., 2002] Enright, A. J., Van Dongen, S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic acids research*, 30(7):1575–1584.

[Erdős and Rényi, 1959] Erdős, P. and Rényi, A. (1959). On random graphs I. *Publicationes Mathematicae Debrecen*, 6:290–297.

[Erdős and Rényi, 1963] Erdős, P. and Rényi, A. (1963). Asymmetric graphs. *Acta Mathematica Hungarica*, 14(3-4):295–315.

[Erk and McCarthy, 2009] Erk, K. and McCarthy, D. (2009). Graded word sense assignment. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 440–449, Singapore, Singapore. ACM.

[Esfahanian, 2012] Esfahanian, A.-H. (2012). Connectivity algorithms. In Oellermann, O. R., Beineke, L. W., and Wilson, R. J., editors, *Topics in Structural Graph Theory*, chapter 12, pages 120–133. Cambridge University Press.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*, pages 226–231, Portland, Oregon, USA. AAAI.

[Euclid, 1956] Euclid (1956). *The thirteen books of Euclid's Elements*. Courier Corporation. Translation, introduction and commentary from the original text of IV-III century BC by Heath, Thomas Little and others.

[Even, 2011] Even, S. (2011). *Graph algorithms*. Cambridge University Press.

[Evert, 2004] Evert, S. (2004). *The statistics of word cooccurrences: word pairs and collocations*. PhD thesis, Universität Stuttgart.

[Feld, 1981] Feld, S. L. (1981). The focused organization of social ties. *American journal of sociology*, 86(5):1015–1035.

[Ferrer i Cancho and Solé, 2001] Ferrer i Cancho, R. and Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265.

[Firth, 1957] Firth, J. R. (1957). *Papers in Linguistics*. Oxford University Press.

[Ford Jr and Fulkerson, 1969] Ford Jr, L. R. and Fulkerson, D. R. (2015 [1969]). *Flows in networks*. Princeton University Press, Princeton, New Jersey, USA.

[Freund et al., 1999] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, 14(771-780):1612.

[Gale et al., 1992a] Gale, W. A., Church, K. W., and Yarowsky, D. (1992a). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5-6):415–439.

[Gale et al., 1992b] Gale, W. A., Church, K. W., and Yarowsky, D. (1992b). Work on statistical methods for word sense disambiguation. In *Technical Report of 1992 Fall Symposium - Probabilistic Approaches to Natural Language*, pages 54–60, Cambridge, Massachusetts, USA. AAAI.

[Ganguly et al., 2009] Ganguly, N., Deutsch, A., and Mukherjee, A., editors (2009). *Dynamics on and of complex networks - Applications to biology, computer science, and the social sciences*. Springer.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (2002 [1979]). *Computers and intractability*, volume 29. W. H. Freeman and Company, New York, New York, USA.

[Ghaemi et al., 2009] Ghaemi, R., Sulaiman, M. N., Ibrahim, H., Mustapha, N., et al. (2009). A survey: clustering ensembles techniques. *World Academy of Science, Engineering and Technology*, 50:636–645.

[Goder and Filkov, 2008] Goder, A. and Filkov, V. (2008). Consensus clustering algorithms: Comparison and refinement. In *Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 109–117, San Francisco, California, USA. Society for Industrial and Applied Mathematics.

[Grätzer, 2011] Grätzer, G. (2011). *Lattice theory: foundation*. Springer.

[Grefenstette, 1994] Grefenstette, G. (1994). *Explorations in automatic thesaurus discovery*, volume 278.

[Griffiths and Harris, 1978] Griffiths, P. and Harris, J. (2014 [1978]). *Principles of algebraic geometry*. John Wiley & Sons, Hoboken, New Jersey, USA.

[Hanks, 2000] Hanks, P. (2000). Do word meanings exist? *Computers and the Humanities*, 34(1):205–215.

[Harary, 1969] Harary, F. (1969). *Graph theory*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hatori et al., 2008] Hatori, J., Miyao, Y., and Tsujii, J. (2008). Word sense disambiguation for all words using tree-structured conditional random fields. In *COLING (Posters)*, pages 43–46, Manchester, UK.

[Haynes et al., 1998] Haynes, T. W., Hedetniemi, S., and Slater, P. (1998). *Fundamentals of domination in graphs*. CRC Press, Boca Raton, Florida, USA.

[Hopcroft and Tarjan, 1973] Hopcroft, J. and Tarjan, R. (1973). Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378.

[Hope and Keller, 2013] Hope, D. and Keller, B. (2013). MaxMax: a graph-based soft clustering algorithm applied to word sense induction. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 368–381, Samos, Greece.

[Hovy et al., 2006] Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York, New York, USA. Association for Computational Linguistics.

[Hromic, 2015] Hromic, H. (2015). python-bcubed. `https://github.com/hhromic/python-bcubed`.

[Hung et al., 2005] Hung, J. C., Wang, C.-S., Yang, C.-Y., Chiu, M.-S., and Yee, G. (2005). Applying word sense disambiguation to question answering system for e-learning. In *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 1, pages 157–162, Taipei, Taiwan. IEEE.

[Jiamjitvanich and Yatskevich, 2009] Jiamjitvanich, K. and Yatskevich, M. (2009). Reducing polysemy in wordnet. In *Proceedings of the 4th International Conference on Ontology Matching-Volume 551*, pages 260–261, Chantilly, Washington DC, USA. CEUR-WS.

[Joshi et al., 2005] Joshi, M., Pedersen, T., and Maclin, R. (2005). A comparative study of support vector machines applied to the supervised word sense disambiguation problem in the medical domain. In *Proceedings of the Second Indian Conference on Artificial Intelligence (IICAI-05)*, pages 3449–3468, Pune, India.

[Karinthy, 1929] Karinthy, F. (1929). Láncszemek. In *Minden másképpen van*. Athenaeum, Budapest, Hungary.

[Karlberg, 1997] Karlberg, M. (1997). Testing transitivity in graphs. *Social Networks*, 19(4):325–343.

[Kaufman and Rousseeuw, 1987] Kaufman, L. and Rousseeuw, P. (1987). Clustering by means of medoids. In *Statistical data analysis based on the L1 norm and related methods*, pages 405–416. North-Holland, Amsterdam, Netherlands.

[Kilgarriff, 1997] Kilgarriff, A. (1997). I dont believe in word senses. *Computers and the Humanities*, 31(2):91–113.

[Kilgarriff et al., 2004] Kilgarriff, A., Rychlý, P., Smrž, P., and Tugwell, D. (2004). The sketch engine. In *Proceedings of the Eleventh Euralex Conference*, pages 105–116, Lorient, France.

[Kilgarriff, 2007] Kilgarriff, D. A. (2007). Word senses. In *Word sense disambiguation: Algorithms and applications*, volume 33, pages 29–46. Springer.

[Kindermann and Snell, 1980] Kindermann, R. and Snell, L. (1980). *Markov random fields and their applications*, volume 1 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, USA.

[Kiss and Strunk, 2006] Kiss, T. and Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

[Klein, 1971] Klein, D. E. (1971). *A Comprehensive Etymological Dictionary of the English Language*. Elsevier, Amsterdam, Netherlands.

[Klepousniotou, 2002] Klepousniotou, E. (2002). The processing of lexical ambiguity: Homonymy and polysemy in the mental lexicon. *Brain and Language*, 81(1):205–223.

[Korf, 1985] Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109.

[Koschützki et al., 2005] Koschützki, D., Lehmann, K. A., Peeters, L., Richter, S., Tenfelde-Podehl, D., and Zlotowski, O. (2005). Centrality indices. In *Network analysis*, Lecture notes in Computer Science, pages 16–61. Springer.

[Kruskal, 1956] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, Williamstown, Massachusetts, USA.

[Le Gall, 2014] Le Gall, F. (2014). Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303, Kobe, Japan. ACM.

[Lee et al., 2004] Lee, Y. K., Ng, H. T., and Chia, T. K. (2004). Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Senseval-3: third international workshop on the evaluation of systems for the semantic analysis of text*, pages 137–140, Barcelona, Spain.

[Levin, 2015] Levin, M. S. (2015). Combinatorial clustering: Literature review, methods, examples. *Journal of Communications Technology and Electronics*, 60(12):1403–1428.

[Li, 2013] Li, C. (2013). Korean word-sense disambiguation using parallel corpus as additional resource. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP) 2013*, pages 113–118, Hissar, Bulgaria.

[Li et al., 2014] Li, L., Titov, I., and Sporleder, C. (2014). Improved estimation of entropy for evaluation of word sense induction. *Computational Linguistics*, 40(3):671–685.

[Lieber and Stekauer, 2009] Lieber, R. and Stekauer, P. (2009). *The Oxford handbook of compounding*. Oxford University Press.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

[Lyons, 1968] Lyons, J. (1968). *Introduction to theoretical linguistics*. Cambridge University Press.

[Manandhar et al., 2010] Manandhar, S., Klapaftis, I. P., Dligach, D., and Pradhan, S. S. (2010). Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68, Los Angeles, California, USA. Association for Computational Linguistics.

[Mandelbrot, 1977] Mandelbrot, B. B. (1977). *Fractals*. John Wiley & Sons, Hoboken, New Jersey, USA.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press, Cambridge, Massachusetts, USA.

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, USA. ACL.

[Martin and Jurafsky, 2000] Martin, J. H. and Jurafsky, D. (2000). *Speech and language processing*. Pearson, Upper Saddle River, New Jersey, USA.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *Workshop proceedings of the first International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Moro et al., 2014] Moro, A., Raganato, A., and Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

[Murata et al., 2001] Murata, M., Utiyama, M., Uchimoto, K., Ma, Q., and Isahara, H. (2001). Japanese word sense disambiguation using the simple bayes and support vector machine methods. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 135–138, Toulouse, France. Association for Computational Linguistics.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

[Nakov and Hearst, 2003] Nakov, P. I. and Hearst, M. A. (2003). Category-based pseudowords. In *Companion Volume of the Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HTL-NAACL) 2003 - Short Papers*, pages 70–72, Edmonton, Alberta, Canada. Association for Computational Linguistics.

[Navigli, 2009] Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

[Navigli and Crisafulli, 2010] Navigli, R. and Crisafulli, G. (2010). Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 116–126, Massachusetts Institute of technology, Cambridge, Massachusetts, USA. ACL-SIGDAT.

[Navigli et al., 2007] Navigli, R., Litkowski, K. C., and Hargraves, O. (2007). Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35, Prague, Czech Republic. Association for Computational Linguistics.

[Navigli and Ponzetto, 2012] Navigli, R. and Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Norris, 1998] Norris, J. R. (1998). *Markov chains*. Number 2 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge university press.

[Opsahl and Panzarasa, 2009] Opsahl, T. and Panzarasa, P. (2009). Clustering in weighted networks. *Social networks*, 31(2):155–163.

[Otrusina and Smrž, 2010] Otrusina, L. and Smrž, P. (2010). A new approach to pseudoword generation. In *Proceedings of the seventh international Conference on Language Resources and Evaluation (LREC'10)*, pages 1195–1199. European Language Resources Association.

[Owoputi et al., 2013] Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HTL-NAACL)*, pages 380–390, Atlanta, Georgia, USA. Association for Computational Linguistics.

[Palmer et al., 2007] Palmer, M., Dang, H. T., and Fellbaum, C. (2007). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(02):137–163.

[Pantel and Lin, 2002] Pantel, P. and Lin, D. (2002). Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 613–619, Halifax, Nova Scotia, Canada. ACM.

[Parker et al., 2011] Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. (2011). *English Gigaword Fifth Edition*. Linguistic Data Consortium, Philadelphia, Pennsylvania, USA.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

[Pelevina et al., 2016] Pelevina, M., Arefyev, N., Biemann, C., and Panchenko, A. (2016). Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183, Berlin, Germany. Association for Computational Linguistics.

[Pilehvar and Navigli, 2013] Pilehvar, M. T. and Navigli, R. (2013). Paving the way to a large-scale pseudosense-annotated dataset. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HTL-NAACL)*, pages 1100–1109, Atlanta, Georgia, USA. Association for Computational Linguistics.

[Pilehvar and Navigli, 2014] Pilehvar, M. T. and Navigli, R. (2014). A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics*, 40(4):837–881.

[Plaza et al., 2011] Plaza, L., Jimeno-Yepes, A. J., Díaz, A., and Aronson, A. R. (2011). Studying the correlation between different word sense disambiguation methods and summarization effectiveness in biomedical texts. *BMC bioinformatics*, 12(1):355.

[Ravasz and Barabási, 2003] Ravasz, E. and Barabási, A.-L. (2003). Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112.

[Read et al., 2012] Read, J., Dridan, R., Oepen, S., and Solberg, L. J. (2012). Sentence boundary detection: A long solved problem? In *Proceedings of the 24th international Conference on Computational Linguistics (COLING): Posters*, pages 985–994, Mumbai, India.

[Remus and Biemann, 2013] Remus, S. and Biemann, C. (2013). Three knowledge-free methods for automatic lexical chain extraction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HTL-NAACL)*, pages 989–999, Atlanta, Georgia, USA. Association for Computational Linguistics.

[Rezapour et al., 2011] Rezapour, A., Fakhrahmad, S., and Sadreddini, M. (2011). Applying weighted KNN to word sense disambiguation. In *Proceedings of the World Congress on Engineering*, volume 3, pages 1834–1838, London, United Kingdom. International Association of Engineers (IAENG).

[Richter et al., 2006] Richter, M., Quasthoff, U., Hallsteinsdóttir, E., and Biemann, C. (2006). Exploiting the Leipzig Corpora Collection. In *Proceedings of the Fifth Slovenian and First International Language Technologies Conference*, IS-LTC '06, pages 68–73, Ljubljana, Slovenia. Slovenian Language Technologies Society.

[Rosch, 2005] Rosch, E. (2005). Principles of categorization. *Etnolingwistyka. Problemy języka i kultury*, (17):11–35.

[Rosenberg and Hirschberg, 2007] Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. SIGDAT-SIGNLL.

[Ruohonen, 2013] Ruohonen, K. (2013). *Graph Theory*. Tampereen teknillinen yliopisto. Originally titled Graafiteoria, lecture notes translated by Tamminen, J., Lee, K.-C. and Piché, R.

[Scalise and Vogel, 2010] Scalise, S. and Vogel, I., editors (2010). *Cross-disciplinary issues in compounding*, volume 311 of *Current Issues in Linguistic Theory*. John Benjamins Publishing Company, Amsterdam, Netherlands.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-of speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, pages 44–49, Manchester, United Kingdom. Association for Computational Linguistics.

[Schult and Swart, 2008] Schult, D. A. and Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, pages 11–16, Pasadena, California, USA.

[Schütze, 1992] Schütze, H. (1992). Dimensions of meaning. In *Proceedings of Supercomputing'92*, pages 787–796, Minneapolis, Minnesota, USA. ACM/IEEE.

[Schütze, 1998] Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123.

[Shannon, 2001] Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.

[Stevenson, 2010] Stevenson, A., editor (2010). *Oxford dictionary of English*. Oxford University Press.

[Strehl, 2002] Strehl, A. (2002). *Relationship-based clustering and cluster ensembles for high-dimensional data mining*. PhD thesis, The University of Texas at Austin.

[Strehl and Ghosh, 2002] Strehl, A. and Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

[Székely et al., 1992] Székely, L. A., Clark, L., and Entringer, R. C. (1992). An inequality for degree sequences. *Discrete mathematics*, 103(3):293–300.

[Tahbaz-Salehi and Jadbabaie, 2007] Tahbaz-Salehi, A. and Jadbabaie, A. (2007). Small world phenomenon, rapidly mixing markov chains, and average consensus algorithms. In *Proceedings of 2007 46th IEEE Conference on Decision and Control*, pages 276–281, New Orleans, Louisiana, USA. IEEE.

[Tanimoto, 1958] Tanimoto, T. T. (1958). An elementary mathematical theory of classification and prediction. Technical report.

[Towell and Voorhees, 1998] Towell, G. and Voorhees, E. M. (1998). Disambiguating highly ambiguous words. *Computational Linguistics*, 24(1):125–145.

[Turney and Pantel, 2010] Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

[van Dongen, 2000] van Dongen, S. (2000). *Graph Clustering by Flow Simulation*. PhD thesis, Universiteit Utrecht.

[Vapnik, 1995] Vapnik, V. (2013 [1995]). *The nature of statistical learning theory*. Springer.

[Véronis, 1998] Véronis, J. (1998). A study of polysemy judgements and inter-annotator agreement. In *Web-Based Proceedings of Herstmonceux 1998 Workshop of Senseval-1*, Herstmonceux, United Kingdom. Online at `http://www.itri.brighton.ac.uk/events/senseval/ARCHIVE/PROCEEDINGS/interannotator.ps`.

[Véronis, 2004] Véronis, J. (2004). Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252.

[Villani, 2016] Villani, C. (2016). Synthetic theory of Ricci curvature bounds. *Japanese Journal of Mathematics*, 11(2):219–263.

[Vinh et al., 2009] Vinh, N. X., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080, Montreal, Québec, Canada.

[Vinh et al., 2010] Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

[Weaver, 1949] Weaver, W. (1955 [1949]). Translation. *Machine translation of languages*, 14:15–23.

[Widdows, 2003] Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HTL-NAACL)*, volume 1, pages 197–204, Edmonton, Alberta, Canada. Association for Computational Linguistics.

[Widdows and Dorow, 2002] Widdows, D. and Dorow, B. (2002). A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational Linguistics*, volume 1, pages 1–7, Taipei, Taiwan. Association for Computational Linguistics.

[Wiriyathammabhum et al., 2012] Wiriyathammabhum, P., Kijsirikul, B., Takamura, H., and Okumura, M. (2012). Applying deep belief networks to word sense disambiguation. *CoRR*, abs/1207.0396.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, California, USA.

[Yarowsky, 2000] Yarowsky, D. (2000). Hierarchical decision lists for word sense disambiguation. *Computers and the Humanities*, 34(1-2):179–186.

[Yip, 2000] Yip, P.-C. (2000). *The Chinese lexicon: A comprehensive survey*. Routledge, London, United Kingdom.

[Zipf, 1935] Zipf, G. K. (1935). *The psycho-biology of language*. Houghton Mifflin, Boston, Massachusetts, USA.

Questa storia è iniziata una nebbiosa e gelida alba di gennaio di quattro anni fa fra le risaie di Vercelli. Oppure, ancora prima in una piccola aula in via Festa del Perdono a un esame di glottologia. I due catalizzatori del mio dottorato sono stati il professor G. Ferrari (ho ancora il libro…) e la professoressa L. Biondi, che devo ringraziare di cuore per la loro gentilezza e disponibilità, e per aver fatto iniziare il ping pong che mi ha fatto capitare in Bicocca.

Un ringraziamento speciale va innanzitutto alla professoressa E. Messina che mi ha accolto nel suo laboratorio. Presto la quiete meditativa del MIND 2 ha lasciato il posto ai compagni d'avventura. Grazie a Macca per tutte le dritte e i pranzi puntuali, e grazie alle ragazze, Debora/Μέλισσα e Pikākśī/Κουκώπη (sorry, in the end it was a futile battle against LaTeX to display devanāgarī correctly) per avermi sopportato così eroicamente, specie nelle concitate fasi finali (e anche in trasferta a Lisbona). A questo punto, sono disposto a chiudere un occhio su qualche cartolina mancante! E come avrei potuto farcela senza le strigliate e la guida della Betta, che ci ha tenuti tutti in riga? La riuscita del mio dottorato la devo in buona parte a lei.

Sul versante germanico, voglio esprimere la mia gratitudine all'ex capitale del Granducato d'Assia per avermi accolto così bene in un altro gelido giorno di gennaio. Ringrazio tutto il gruppo dell'LT, Dr. Alexander, Benjamin, Dr. Bonaventura (e il kebab del giovedì), Eugen (und die wichtigen Kuchenpausen), Gerold, Dr. Martin, Steffen, Sarah, Seid. E un grazie sentito al professor C. Biemann, l'altra pietra angolare di questo dottorato. E infine un pensiero anche ai giocatori di Magic del Paladin's Place, ai Lilien e all'atmosfera surreale della primavera tedesca.

Ma non si vive di sola università! Visto che nelle altre tesi non ho messo nessun ringraziamento, qui devo recuperare. Un grazie disordinato, enorme e multiforme va a tutti gli amici e le amiche: gli svizzeri, i nerd, i matematici, gli erasmi, i norvegesi, i giocatori, perché no i trekker georgiani, e chi più ne ha più ne metta, chi vedo di più e chi vedo di meno, a cui ho provato mille volte a spiegare cosa stessi combinando fra ennellepì e vuesseì, ma non so mica se è stato più facile che con le proiezioni di superfici algebriche.

E poi dovrò anche ringraziare quei due lì che mi sono stati sempre, ma proprio sempre accanto, in tutti i momenti, tristi e divertenti. Le parole non bastano, quindi la smetto subito.

E infine…

Дандаа гүйдэг галт туулайхандаа баярлалаа! Бид эртний чулуун хотод танилцснаас хойш чи зүрхэнд минь оршсоор байна.

E ora, avanti!