SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Department of
Computer Science, Systems and Communication

PhD program in Computer Science   Cycle XXXII

Corpus-based Comparison of Distributional Models of Language and Knowledge Graphs

Bianchi Federico
746914

Tutor: Prof. Enza Messina

Supervisor: Prof. Matteo Palmonari

Coordinator: Prof. Leonardo Mariani

ACADEMIC YEAR 2018/2019

When you find yourself, in a morning, averse to rise, have this thought at hand: I arise to the proper business of a man: And shall I be averse to set about that work for which I was born, and for which I was brought into the universe? Have I this constitution and furniture of soul granted me by nature, that I may lie among bed-clothes and keep myself warm?

— Marcus Aurelius Antoninus Augustus, Meditations

Study hard and all things can be accomplished; give up and you will amount to nothing.

— Yamaoka Tesshu

Dedicated to all the people that have helped me during these years of study. I love you all.

ABSTRACT

One of the main goals of artificial intelligence is understanding how intelligent agent acts. Language is one of the most important media of communication, and studying theories that can account for the meaning of natural language expressions is a crucial task in artificial intelligence.

Distributional semantics states that the meaning of natural language expressions can be derived from the context in which the expressions appear. This theory has been implemented by algorithms that generate vector representations of natural language expressions that represent similar natural language expressions with similar vectors.

In the last years, several cognitive scientists have shown that these representations are correlated with associative learning and they capture cognitive biases and stereotypes as they are encoded in text corpora. If language is encoding important aspects of cognition and our associative knowledge, and language usage change across the contexts, the comparison of language usage in different contexts may reveal important associative knowledge patterns. Thus, if we want to reveal these patterns, we need ways to compare distributional representations that are generated from different text corpora. For example, using these algorithms on textual documents from different periods will generate different representations: since language evolves during time, finding a way to compare words that have shifted over time is a valuable task for artificial intelligence (e.g., the word "Amazon" has changed its prevalent meaning during the last years).

In this thesis, we introduce a corpus-based comparative model that allows us to compare representations of different sources generated under the distributional semantic theory. We propose a model that is both effective and efficient, and we show that it can also deal with entity names and not just words, overcoming some problems that follow from the ambiguity of natural language. Eventually, we combine these methods with logical approaches. We show that we can do logical reasoning on these representations and make comparisons based on logical constructs.

PUBLICATIONS

Publications releated to this research work:

- Di Carlo, V., **Bianchi, F**. and Palmonari, M. (2019). **Training Temporal Word Embeddings with a Compass**. AAAI.
- Hitzler, P., **Bianchi, F**., Ebrahimi, M., and Sarker, M. K. (2019) **Neural-symbolic integration and the Semantic Web**. Semantic Web, 1-9.
- **Bianchi, F**., Palmonari, M., and Nozza, D. (2018). **Towards Encoding Time in Text-Based Entity Embeddings**. In ISWC.
- **Bianchi, F**., Palmonari, M., Cremaschi, M., and Fersini, E. (2017). **Actively Learning to Rank Semantic Associations for Personalized Contextual Exploration of Knowledge Graphs**. In ESWC (pp. 120-135). Springer, Cham.
- **Bianchi, F**. and Hitzler, P. (2019). **On the Capabilities of Logic Tensor Networks for Deductive Reasoning**. AAAI-MAKE Spring Symposium.
- **Bianchi, F**., Palmonari, M., Hitzler, P., and Serafini, L. (2019). **Complementing Logical Reasoning with Sub-Symbolic Commonsense**. In International Joint Conference on Rules and Reasoning.
- Vimercati M., **Bianchi, F**., Soto, M., and Palmonari, M. (2019). **Mapping Lexical Knowledge to Distributed Representations for Ontology Concept Invention**. In AIXIA.
- **Bianchi, F**., Soto, M. Palmonari, M., and Cutrona, V. (2018). **Type Vector Representations from Text: An empirical analysis**. in DL4KG@ESWC.
- **Bianchi, F**. and Palmonari, M. (2017). **Joint Learning of Entity and Type Embeddings for Analogical Reasoning with Entities**. In Proceedings of the NLP4AI@AIXIA.
- Ebrahimi, M., Sarker, M. K., **Bianchi, F**., Xie, N., Doran, D., and Hitzler, P. (2018). **Reasoning over RDF Knowledge Bases using Deep Learning**. arXiv preprint arXiv:1811.04132.
- **Bianchi, F, Rossiello, G., Costabello, L,. and Palmonari, M. (2020). Knowledge Graph Embeddings to Support Explainability.** In: I. Tiddi, P. Hitzler and F. Lecue, ed., Knowledge Graphs for eXplainable AI. IOS (to appear).

ACKNOWLEDGEMENTS

I would like to thank the following people that contributed to this research.

Many thanks to **Prof. Pascal Hitzler**, that gave me the possibility of joining his lab at Wright State University. I am grateful to him because I had the chance to learn novel topics while I was in the United States.

Many thanks to **Prof. Luciano Serafini**, for the interesting talks and the possibility of working together on a wonderful topic.

Many thanks to the people of **SpazioDati**[1], that were also kind enough to share their data and some of their valuable technologies with me to run experiments.

I would also like to thank my students **Valerio Di Carlo**, **Manuel Vimercati** and **Paolo Nicoli** for the amazing work they did during their studies. I would also like to thank all my other students that brought valuable contributions to this research.

Finally, I'd like to thank my supervisor, **Prof. Matteo Palmonari**, to which I am really grateful. During these three years he allowed me to follow the research directions I wanted and always explained me how to correct my mistakes. He was the one that taught me how to be a better scientist.

1 https://spaziodati.eu/en/

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

LTNS Logic Tensor Networks

MLNS Markov Logic Networks

CADE Compass-Aligned Distributional Embeddings

TWEM Temporal Word Embedding Model

TWE Temporal Word Embedding

NAC-S News Article Corpus Small

NAC-L News Article Corpus Large

MLPC Machine Learning Papers Corpus

T1 Analogy Testset 1

T2 Analogy Testset 2

STAT Staticness

WEM Word Embedding Model

TWA Temporal Word Analogy

KG Knowledge Graph

LCS Least Common Subsumer

ACPS Average Children Pairwise Similarity

IC Information Content

KG Knowledge Graph

WPATH Weighted Path Length

TEE Typed Entity Embeddings

TE Type Embeddings

EE Entity Embeddings

KG Knowledge Graph

Part I

BACKGROUND

1 INTRODUCTION

Artificial intelligence (AI) can be defined as the field that studies *the synthesis and analysis of computational agents that act intelligently* [102]. An agent is something that acts and a computational agent is an agent whose action can be explained *in terms of computations* [102]. One of the main goals of AI is understanding the principle of intelligent behavior. There are many currents in AI, but two of them have been influential to the development of computational agents in the last decades, *Knowledge Representation* (KR) and *Natural Language Processing* (NLP). Knowledge Representation has the objective of defining models that can be used to structure information and uses logic as its main foundation. Key property of logic is inference: starting from structured information, new information can be inferred with the use of deductive, inductive or abductive reasoning methods. In order to structure information, artificial languages have been defined. These languages contain symbols that are used to refer to real-world individuals. One now widely adopted method for KR is the Knowledge Graph (KG), a knowledge abstraction model in which entities (e.g., individuals like Barack Obama, the city Honolulu, etc...) are connected by relationships (e.g., Barack Obama, birth Place, Honolulu). The success of these methods inspired people from the computational linguistic community to use artificial languages to treat natural language properties; for example, natural language inference studies the ways in which one sentence is a consequence of other sentences using logical approaches.

While NLP employed with success artificial languages, from the '50s a new theory of language, called distributional semantics, started to get the attention of linguists. This theory roughly states that the meaning of language expressions can be inferred from their usage in natural language (e.g., the words "cat" and "kitten" are used in similar contexts). In the last years, with the advent of algorithms able to process large amounts of data, algorithms based on distributional semantics have been widely employed in NLP. These algorithms create a vector based representation of linguistic items and have been effectively applied to different tasks such as natural language inference [26], word similarity [100], natural language generation [136], question answering [149]. Part of this success is also due to the recent growth in popularity of deep learning approaches [54] that can now ingest a large amount of data and have outperformed other algorithms in several tasks. The success of these methods got the attention of researchers in the KR field that started to use these algorithms to generate vector representations of structured information [25, 126]. More recently, approaches that learn to combine reasoning and neural networks have been proposed [114], this last family of approaches is usually called neuro-symbolic [48].

1.1 BACKGROUND

Vector representations of words and entity names [25, 93]) have been widely adopted as approaches in knowledge representation. In these representations, knowledge is encoded into dense vectors with the general aim of representing elements that are deemed to be similar with similar vectors (i.e., similar entities should be represented with *close* vectors).

Representations that are based on distributional semantics can be generated from textual documents by considering the co-occurrence of elements in text [92]. These representations are *contextual* in a sense that they use information that comes from the usage of language expressions in textual documents to derive the representations. These representations have got much attention in the literature and have been applied to different tasks [6, 26, 37, 73, 100, 109, 120, 136, 145, 149]. Word2vec [93] is an out-of-the-box algorithm that efficiently generates vector representations of words; these representations are also called word embeddings. The introduction of word2vec had a great impact in the AI community. These representations capture intrinsic properties of languages thus offering the possibilities of solving propositional analogical reasoning task with words (e.g., "Rome" is to "Italy" as ? is to "France") and of evaluating the similarity between words. However, distributional semantic based representations do not allow structured logical reasoning in the same way KR approaches do.

These text-based representations have been found to be correlated with associative learning by psychologists [77] and there is evidence that these representations encode bias and stereotypes that are typically present in text [27]. Thus, studying how this kind of representations mutate is a valuable task for artificial intelligence. Moreover, since these representations capture bias coming from context, *representations generated from different contexts will be different*. For example, in the '50s, people used words in a way that is different from how we use them now; we can think of the word "amazon" or the word "gay" in the fifties (i.e., temporal context). At the same time, an English speaker and an American speaker might use different terms to refer to the same object (e.g., lift/elevator) or use the same term to identify different things, for example the word "football" can be interpreted as two completely different sports (i.e., language context).

As we said in the last years, several cognitive scientists have shown that these representations are correlated with associative learning and they capture cognitive biases and stereotypes as they are encoded in text corpora. If language is encoding important aspects of cognition and our associative knowledge, and language usage change across the contexts, the comparison of language usage in different contexts may reveal important associative knowledge patterns. Thus, if we want to reveal these patterns, we need ways to compare distributional representations that are generated from different text corpora.

Different approaches have been introduced to study and compare these representations [5, 6, 27, 49, 52, 145], but there is currently no integrated framework that allows to effectively compare corpus-based representations that come from different sources. Moreover, these frameworks analyze language expressions that are ambiguous: using entity names in place of words in the representations can better characterize the problem. Eventually, once we represent entity names, we can use logical reasoning, via a method that deals with vector representations, to logically compare different representations.

In this thesis, we present a framework to compare distributional representations of words, entities, and types. This framework provides means of aligning representations that come from different corpora and to compare them by looking at semantically similar words and effectively computing the similarities. On top of this framework, we show that we can use a neuro-symbolic system (i.e., a system that combines symbolic logic and neural networks) to provide support for logical inferences over different vector spaces.

1.2 MOTIVATIONS

Different people have different representations of the world and this is reflected in the way they speak and write. As we said before, language evolves during time. Two examples of this kind are the entity name "dbr:Amazon" or the word "apple": the entity was once less ambiguous but the advent of the company brought us ambiguity in the usage of the symbol "amazon", the same has happened with the word apple. The same problem happens also across different partitions of the same language, English vs American (e.g., lift/elevator). Comparing different representations of language becomes important to understand how language evolves across time [73] and to map language differences. Moreover, distributional representations lack the possibility of defining reasoning methods that are proper of logical languages.

Caliskan et al. [27] have found that word representations that are currently used in artificial intelligence are biased towards different aspects and some of them are critical like race and gender. This means that when we use algorithms we might include bias without being aware of it. It is important to find way to understand, compare and evaluate the representations and how they evolve: think for example of an algorithm that use word representation to suggest possible candidates for a new job position: we do not want these algorithms to be biased towards race and gender, but we want them to focus on the specific aspects that we want to evaluate.

1.3 PROBLEM STATEMENT

Recently introduced algorithms [93] allow us to consider textual documents and generate vector representations of words by considering word-context co-occurrences; these algorithms are based on the distributional hypothesis that states that similar words appear in similar contexts. These algorithms try to create similar vectors for similar words. A simple way to study these differences in language is to generate different vector representations for different corpora, but this does not allow to compare all the different representations: while neural networks provide extensive methods to generate embeddings, their low-level stochasticity does not allow us to generate comparable representation. For example, running the word2vec algorithm two times on the same corpus generates different vectors for the same words. Effectively aligning language representations is fundamental for many different tasks: looking at language evolution between text produced in different periods is useful to explore the evolution in meaning [73, 109, 120, 145]. Nonetheless, aligning language coming from different sources can help to evaluate the intrinsic language differences (e.g., geographic differences [6]) between these sources and study language bias [27].

State-of-the-art approaches to align representations of word embeddings have focused on temporal data [73, 109, 120, 145]; while a lot of work has been done to study language bias and stereotypes, few works have explored language variations aligning embeddings [6]. To the best of our knowledge, no general framework to compare different corpora has been proposed.

The research problem that we will discuss and study in this thesis is related to the creation of a framework for the comparative analysis of language items. We refer to this framework as a corpus-based framework since it is based on textual documents.

1.4 CONTRIBUTIONS

This thesis has the aim of defining an efficient and effective framework to compare natural language expressions with an efficient and effective method. The major contributions of this research work are related to the definition of a framework for the comparison of distributional models, aiming to compare both words and structured knowledge that comes from a knowledge graph. In detail we define:

1. An alignment methodology for distributional representations that is both effective and efficient and that can be used to align representations that come from different sources;
2. A purely distributional model of Knowledge Graphs' elements like entities and types that can have multiple contexts of usage: This model is based on an entity linker and it can be thus applied to different sources of text;
3. An approach that combines neuro-symbolic techniques with entity embeddings to provide support for logical reasoning on distributional representations.

We will use (1) to generate comparable representations of natural language expressions. This will allow us to compare representations that come from different sources as text from different time periods or from different newspapers. Since language is ambiguous we will need to rely on (2) to first find entity names and type names from a knowledge graph inside the text before generating the actual representations. Since using (2) allows us to generate representations of entities we can use a (3) neuro-symbolic model to learn to reason over these distributional representations.

1.5 ORGANIZATION OF THIS THESIS

Figure 1 depicts the structure of the following thesis. Chapters 4 and 5 cover two different aspects of the problem that are closely related but independent. The rest of the thesis has a linear structure.

We now show more details of the chapters of this thesis, publications are listed following a chronological order, we highlight the most important ones using a red *:

Chapter 1: Introduction In this chapter, we have introduced the content of this thesis, outlining the main concepts and the most relevant content. The rest of the chapter will also outline the main research questions that this research work aims to answer.

Chapter 2: Preliminaries In this chapter we explain the basic notions that are part of this research work. We explain the main applications and differences between logic-inspired semantics and distributional semantics, showing the main components and some differences.

Chapter 3: Related Work We explore related approaches by analyzing three different areas: comparative methods, word embeddings and knowledge graph embeddings. Notions related to knowledge graph embeddings approaches will also be present in an upcoming book chapter:

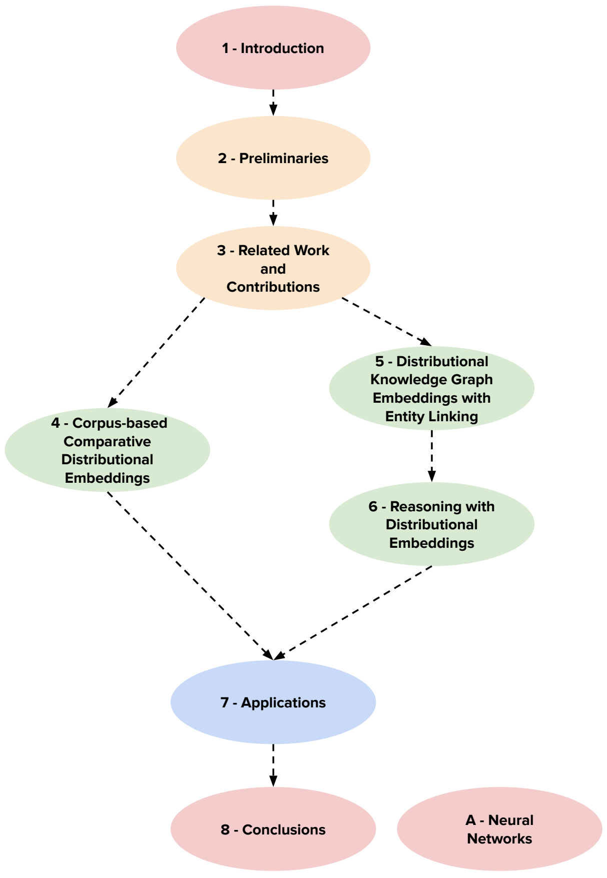

1 - Introduction
2 - Preliminaries
3 - Related Work and Contributions
4 - Corpus-based Comparative Distributional Embeddings
5 - Distributional Knowledge Graph Embeddings with Entity Linking
6 - Reasoning with Distributional Embeddings
7 - Applications
8 - Conclusions
A - Neural Networks

Figure 1: The structure of this thesis. Red ellipses represent the more general chapters; yellow ellipses contain the material that is related to what we describe in this research work and outline the main contributions; the green ellipses contain the main contributions of this research while the blue ellipse contains applications and possible future directions that come out from this thesis.

- Bianchi, F., Rossiello, G., Costabello, L,. and Palmonari, M. (2020). **Knowledge Graph Embeddings to Support Explainability.** In: I. Tiddi, P. Hitzler and F. Lecue, ed., Knowledge Graphs for eXplainable AI. IOS (to appear).

Chapter 4: Comparative Distributional Framework In this chapter, we outline the ideas behind our Corpus-based Comparative Distributional Framework and we introduce Compass-Aligned Distributional Embeddings (CADE) a novel method to efficiently and effectively align distributional representation that comes from different sources.

The research questions answered by the following chapter are:

- **Q4.1** which are the performance of an unsupervised method for the implicit alignment of distributional models on a paradigmatic case?

- **Q4.2** can this method be used even in context when some sources provided less data than others?
- **Q4.3** to which extent can this method be used in different contexts?

Results described within this chapter are based on the following work, in which we show how distributional representations of temporal word embeddings can be aligned.

- *Di Carlo, V., Bianchi, F. and Palmonari, M. (2019). **Training Temporal Word Embeddings with a Compass**. In AAAI.

This method has been extended in this thesis and novel experiments to show the level of generalization of the method have been run. The contents of this chapter are meant to describe a paper, that extends the one stated above, that will be submitted after the conclusion of this thesis.
Chapter 5: Distributional Knowledge Graph Embeddings with Entity Linking This chapter focuses on one of the contributions of this research work, which is the introduction of a purely distributional model for the representation of entities and types of a knowledge graph using entity linking. In this chapter we aim to answer the following research questions:

- **Q5.1** can we create purely distributional models of KG elements (i.e., entity and types) that can be applied to any text?
- **Q5.2** is the notion of similarity computed within this space good enough to support similarity-based approaches to reasoning like analogical reasoning?
- **Q5.3** which is the effect of the entity linking phase on the embeddings?
- **Q5.4** which are the properties of a distributional representation of types?

Recent work about the topic of this chapter has been published in the following articles:

- Bianchi, F., Soto, M., Palmonari, M., and Cutrona, V. (2018, June). **Type Vector Representations from Text: An empirical analysis**. In DL4KGS@ESWC.
- Bianchi, F. and Palmonari, M. (2017, November). **Joint Learning of Entity and Type Embeddings for Analogical Reasoning with Entities**. In NL4AI@AIIA.
- *Bianchi, F., Palmonari, M., and Nozza, D. (2018, October). **Towards Encoding Time in Text-Based Entity Embeddings**. In ISWC.

Chapter 6: Reasoning with Distributional Embeddings

In this chapter, we add one more layer to our comparative framework and we show how we can combine distributional representations of entities and types with a neuro-symbolic system, the Logic Tensor Networks. This will allow us to do a comparison over distributional spaces using logical axioms.

The research question answered by the following chapter is:

- **Q6.1** Can we provide a method to do logical inference over distributional representations?

This chapter is mainly based on the following work:

- *Bianchi, F., Palmonari, M., Hitzler, P., and Serafini, L. (2019). **Complementing Logical Reasoning with Sub-Symbolic Commonsense**. In International Joint Conference on Rules and Reasoning.

Other articles that we published in the neuro-symbolic field:

- Ebrahimi, M., Sarker, M. K., Bianchi, F., Xie, N., Doran, D., and Hitzler, P. (2018). **Reasoning over RDF Knowledge Bases using Deep Learning**. arXiv preprint arXiv:1811.04132.
- Bianchi, F. and Hitzler, P. (2019). **On the Capabilities of Logic Tensor Networks for Deductive Reasoning**. AAAI Spring Symposium.
- *Hitzler, P., **Bianchi, F.**, Ebrahimi, M., and Sarker, M. K. (2019) **Neural-symbolic integration and the Semantic Web**. Semantic Web, 1-9.

Chapter 7: Applications This chapter contains applications of the models we have defined in the previous chapters. We will show that the use of a purely distributional model for types and entities in a comparative framework can bring value in different tasks such as knowledge exploration and neuro-symbolic reasoning.

Chapter 8: Conclusions Finally, we end this thesis by providing conclusions to this work by also highlighting the most important content. Eventually, we provide possible research directions that may start from the results provided within this thesis.

Appendix: Neural Networks We also add an appendix with a short introduction to neural networks and machine learning. For a more complete introduction, we refer to the book by Goodfellow, Bengio, and Courville [54].

1.6 REPRODUCIBILITY

Recently, reproducibility has become a great issue in AI-related fields. Thus, each of the experiments that were run for this thesis can be replicated using codes and models freely available online. Since the experiments have been introduced in different papers which contain links to different GitHub repositories, we decided to create a single GitHub repository that contains all the instructions and the references to the contributions that are described within this thesis[1].

1 https://github.com/vinid/phd_thesis

2 PRELIMINARIES

In the following chapter, we introduce the notions that stand behind this research work. This chapter should give the reader most of the required knowledge to access the rest of this work. We will begin this chapter with an overview related to what meaning is and about two different methods that have been introduced to deal with it. Then, we explore the details and some applications of these methods, and we eventually end the chapter with the notation used in this thesis and a summary of the chapter.

2.1 TWO DIFFERENT AND INTERLACING STORIES

Logic has been introduced to describe the properties of mathematics [43] and can be considered as a family of *artificial languages* used to describe general properties. For example, First-order Logic (FOL) can be used to describe the properties of mathematical relations and functions as shown by Peano's axiomatization of arithmetic [128, 131]. Logic defines means to do reasoning over properties, and thus logical reasoning became the preferred tool to provide account for inferences about the world that surrounds us because of its robust mathematical foundations. The meaning we assign to a sentence is related to the concept of truth, a key concept in logic. Our world contains objects and we want to be able to use a logic language to build representations of these objects in order to use them in computational settings and subsequently infer meaningful facts [102]. Rapidly skipping in time, logical approaches have become popular in the knowledge representation community. We can cite logic programming languages like Datalog [55] and logical languages like Description Logics [3] as examples of influential methods in the knowledge representation field. The two central notions in symbolic logic are consequences and inferences. Representing properties like "each mammal is an animal" $\forall x : \mathrm{mammal}(x) \rightarrow \mathrm{animal}(x)$ and supporting inference over symbols allows us to derive new knowledge.

The linguistic community has taken much inspiration from these approaches and has extensively used logical approaches for tasks carried out on natural language expressions [71]. Artificial languages based on and inspired by formal logic have been used to account for the meaning of natural language expressions [71].

In the context of linguistics, we can mention truth-conditional semantics [71, 103], in which the meaning of a sentence is in function of its truth conditions: the meaning of a sentence comes from the conditions that make the statement true or false. Note, that this is independent from the actual truth value of the sentence. One can understand the meaning of a sentence even if it has no evidence related to the validity of the statement included in the sentence.

Human language is a communication system based on a discrete set of arbitrary and conventional *symbols* and a set of rules of how to compose them to convey meanings [83]. Logical methods and inferential processes had a significant impact on natural language processing. The main principle is that to interpret natural language under the form of artificial languages. For example, "the cat is on the table which is in the room" opens up to the possibility of inferring sentences as "the cat is in the room".

On the other hand, starting from the '50s a theory of meaning was introduced in the field of linguistics. This theory is generally called *distributional semantics*. The leading advocates were the linguists Harris and Firth [46, 62], but other hints of this theory have been also found in the late writings of the philosopher Wittgenstein [139]. This theory advocates for a usage-based perspective on language: the general idea is that the meaning of linguistic items derives from the usage in language of those items. This theory was widely adopted by linguists [77] and has captured the attention of psycho-linguistics and researchers with cognitive science backgrounds [27].

In more recent years, algorithms that are based on this theory have been widely adopted in natural language processing [36, 93, 100, 101]; these algorithms have been of inspiration for similar approaches onKR as knowledge graphs [25, 41, 81, 98, 126, 133, 134, 138, 140, 141].

In the next section, we will explore the main characteristics of these two approaches that have different stories but were used to solve similar tasks and influenced one another.

2.2 LOGIC-INSPIRED SEMANTICS

2.2.1 *Applications*

The definitions of methods to efficiently and effectively organize knowledge is fundamental to many different tasks. For example, databases can be modeled using logic. Logical reasoning and inferential processes are what make logical reasoning a powerful tool for artificial intelligence. Logical languages provide methods for symbol manipulation, in fact there is a line of research that sustains the hypothesis that intelligence is strongly related to symbol manipulation. *A physical symbol system has the necessary and sufficient means for general intelligent action.* [96][1].

1 Note that this theory has not reached consensus and have several critics that sustain that there is more to intelligence than symbol manipulation with some people suggesting that neural networks [54] can actually overcome some of the limits of

Knowledge has been traditionally considered over distinct layers; the most classical distinction is between words and concepts [32], which allows to take into account linguistic phenomena such as ambiguity. Moreover, concepts are generally organized in connected layers. In cognitive psychology, concepts are organized in three layers: superordinate, subordinate and basic [90]. Concepts in layers are connected by specific hierarchical relations such as *hypernymy*.

2.2.2 *Principles and Definitions*

A logical language is specified by a syntax and associated with a semantics. The syntax defines the way symbols are organized in more complex formulas, while the semantics is (often) given by an interpretation function that associates truth values to formulas.

We hereby give a very intuitive and simplified introduction to first order logic. For a complete introduction, we refer to one of the many textbooks present in literature [102]. We will refer again to first order logic when we will describe a method to combine distributional representations and symbolic logic using neuro-symbolic methods in Chapter 7. First-order logic allows us to describe knowledge about a domain such as entities and individuals: we have objects and we want to be able to build representations of these objects in such a way we can reason about them in computational settings [102]. We can make these representations by introducing terms to refer to **individuals**, which are real elements of the worlds (e.g., people, cities) and **predicates**, to talk about properties of individuals. To define an individual or a relation we use symbols and we manipulate symbols on the basis of logical approaches. Logical languages, for example first-order logic (FOL), offer a syntax that specifies how symbols can be manipulated and semantics through the interpretation of symbols.

A logical language $\mathcal{L}$ contains the following elements: a set of *terms* that contains constants symbols as names for the individuals (e.g., "Milan", "Italy", "University of Milan-Bicocca") and that contains variables (e.g., x, y and z)[2]; a set of predicates to express a relation between individuals (e.g., $location(x, y)$). We define an *atomic formula* as a formula that applies a predicate to a series of terms (e.g., $location(Milan, Italy)$. Predicates can take different arguments as inputs and the number of arguments is defined as the arity of the predicate. More complex formulas can be defined by composing different formulas with the use of connectives and quantifiers (e.g., $\forall$, $\exists$, $\neg$, &, $|$, $\rightarrow$). Note that logic is compositional. Also, we refer to formulas that do not contain variables as *ground formulas*.

logic. This is also why approaches that combine symbolic reasoning and computational methods (such as neural networks) have been introduced in the state of the art. See [95] for a nice overview on the symbolic/neural network debate.

2 For the sake of giving only a simplified introduction we ignore functions in this definition.

The semantics of the language is given with the use of an interpretation function I over a domain D. We can consider an interpretation I_1 and a domain D_1 that represents the set of the things that have existed in the world. Terms are interpreted as names for elements (e.g., $I_1(\mathrm{Milan})$ = the city Milan). Predicates are instead interpreted as relations and sets in the domain D (e.g., $I_1(\mathrm{location}(x, y))$ the relation that holds in D_1 between cities and countries).

Atomic formulas are interpreted as true or not in a given interpretation, that is, satisfied by an interpretation, $I_1(\mathrm{location}(\mathrm{Milan}, \mathrm{Italy}))$, if and only if the individuals denoted by the symbols "Milan" and "Italy" are, in D_1 , in the relation used in I_1 to interpret the predicate "location". The interpretation of complex formulas is recursively defined from the interpretation of variables, connectives and quantifiers.

INFERENCE A key aspect of logic is the inferential process that allows us to draw new facts from current state of knowledge. We say that $\mathrm{populated\_place}(\mathrm{Milan})$ is logical consequence of a set of premises $\{\forall x, \mathrm{city}(x) \rightarrow \mathrm{populated\_place}(x), \mathrm{city}(\mathrm{Milan})\}$ if and only if the ground axiom $\mathrm{populated\_place}(\mathrm{Milan})$ is true in all the interpretations in which the premises are true (it is satisfied by all the interpretations that satisfy the premises). If a formula ϕ is true in at least one interpretation then the formula is satisfied and that interpretation is a model for ϕ.

KNOWLEDGE BASE A knowledge base is a structure to organize knowledge that supports inferential mechanisms that are inspired by the notions of logical consequence.

Knowledge Graphs

A special case of knowledge base is the Knowledge Graph (KG) that is, roughly speaking, a knowledge base in which we have only predicates of arity 2. Considering only predicates with arity 2 allows us to represent a knowledge base using a graph structure. A KG describes entities (e.g., the city Milan) of the real world and links them through the use of edges that represent relationships. Note that this structure allows us to use an infix notation to describe knowledge facts: the prefix notation of the following relationship $\mathrm{country}(\mathrm{Milan}, \mathrm{Italy})$ can be rewritten using the notation $(\mathrm{Milan}, \mathrm{country}, \mathrm{Italy})$.

Different definitions of knowledge graph exist [39]. For example, we can describe a KG as a labeled multi-graph or as a set of subject-predicate object-triples (s, p, o). It is possible to interpret the triples as ground formulas under the form $\mathrm{predicate}(\mathrm{subject}, \mathrm{object})$ (e.g., $\mathrm{country}(\mathrm{Milan}, \mathrm{Italy})$). See Figure 3 for an example of knowledge graph.

We also hereby introduce the concept of ontology, that is closely related to the concept of KG. *An ontology is a representation of conceptual system via a logical theory* [51, 53] that supplies a shared vocabulary that can be used to model a domain: for example, typing of the individuals (e.g., $\mathsf{City}(\mathsf{Milan})$). For the scope of this thesis, in which we focus our attention on entities and entities representation, we use the following definition of knowledge graph in which we highlight the main components that are used through this thesis.

Definition 1 (Knowledge Graph) *A knowledge graph is 6-tuple that contains* $(E, L, T, P, A, \mathcal{L})$*:*

- E*: a set of symbols that identify entities (≈ constants);*
- L*: a set of literals, i.e., strings, numbers, etc. (≈ constants) ;*
- T*: a set of type symbols (≈ unary predicate symbols);*
- P*: a set of relation symbols (≈ binary predicate symbols);*
- $A_{\mathcal{L}}$*: a set of axioms in a logical language* $\mathcal{L}$*.*

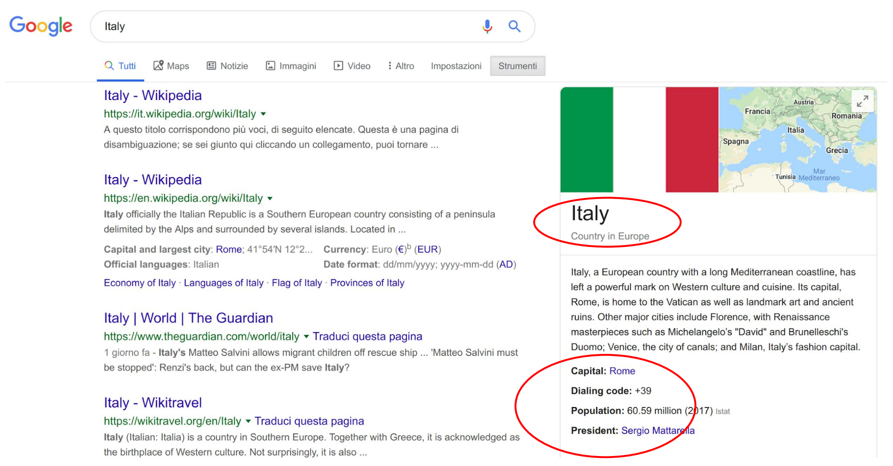
The axioms of a knowledge graph can be divided in:

- A*-facts:*
  - *Types of entities and literals:* $\mathsf{type}(e) \mid \mathsf{type}(l)$ *- with* $e \in E$ *and* $l \in L$*, that specify that an entity is of a given type;*
  - *Relations between entities:* $r(e_1, e_2) \mid r(e, l)$ *with* $e_i \in E$ *and* $l \in L$ *and* $r \in P$*.*
- A*-general-axioms: they depend on the language used to describe the ontology; we consider as aminimal requirement for a knowledge graph having axioms with the form* $\forall x(\mathsf{type}_1(x) \to \mathsf{type}_2(x))$ *that specify subtype dependencies that define a partial order over* T*.*

Observe that this definition accounts for KGs whose semantics is defined more expressively by logical axioms as well as simple KGs used in the industry and stored in graph databases.

KGs are now a widely adopted method for knowledge representation: for example, many public KGs exist as DBpedia, YAGO, and Wikidata. One famous example of KG used in the industry is the Google KG, used to enrich the results of the Google search engine. KGs are sources of knowledge and there are different ways to query this large amount of information. Google uses its knowledge graph to support is search engine; Figure 2 shows an example of usage of the Google KG to give additional content to users that search information about the term "Italy". In fact, knowledge graph are also important for knowledge exploration [18].

KG's entities are generally organized in sub-types hierarchies (e.g., (Italy, instance of, Country) and (Country, sub-type of, Populated


Italy

Tutti · Maps · Notizie · Immagini · Video · Altro · Impostazioni · Strumenti

Italy - Wikipedia
https://it.wikipedia.org/wiki/Italy
A questo titolo corrispondono più voci, di seguito elencate. Questa è una pagina di disambiguazione; se sei giunto qui cliccando un collegamento, puoi tornare ...

Italy - Wikipedia
https://en.wikipedia.org/wiki/Italy
Italy officially the Italian Republic is a Southern European country consisting of a peninsula delimited by the Alps and surrounded by several islands. Located in ...
Capital and largest city: Rome; 41°54′N 12°2... **Currency**: Euro (€)b (EUR)
Official languages: Italian **Date format**: dd/mm/yyyy; yyyy-mm-dd (AD)
Economy of Italy · Languages of Italy · Flag of Italy · Provinces of Italy

Italy | World | The Guardian
https://www.theguardian.com/world/italy ▾ Traduci questa pagina
1 giorno fa - **Italy's** Matteo Salvini allows migrant children off rescue ship ... 'Matteo Salvini must be stopped': Renzi's back, but can the ex-PM save **Italy**?

Italy - Wikitravel
https://wikitravel.org/en/Italy ▾ Traduci questa pagina
Italy (Italian: Italia) is a country in Southern Europe. Together with Greece, it is acknowledged as the birthplace of Western culture. Not surprisingly, it is also ...

Italy
Country in Europe

Italy, a European country with a long Mediterranean coastline, has left a powerful mark on Western culture and cuisine. Its capital, Rome, is home to the Vatican as well as landmark art and ancient ruins. Other major cities include Florence, with Renaissance masterpieces such as Michelangelo's "David" and Brunelleschi's Duomo; Venice, the city of canals; and Milan, Italy's fashion capital.

Capital: Rome
Dialing code: +39
Population: 60.59 million (2017) Istat
President: Sergio Mattarella

Figure 2: Example of usage of the Google Knowledge Graph to extend query results (highlighted in red circles).

Place). These types are usually defined in an ontology. In Figure 3, we show an example of KG in which we highlight the most important components: entities share links one with the other and are also connected to the type hierarchy.

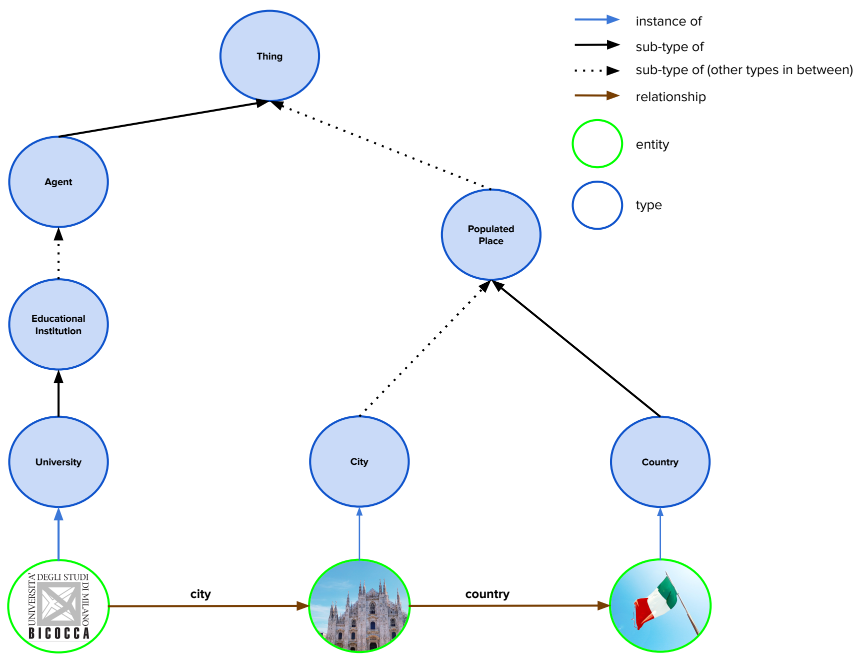


Figure 3: Example of a knowledge graph with a type hierarchy.

A short note on the semantic web languages

To represent, share knowledge on the web specific languages have been defined. The semantic web [14] is an "extension" of the standard web where structured metadata is added to pages. In this way, information relative to the page is easily accessible to both humans and machines: the semantic web is focused on the "meaning" of data and allows us to create interlinked knowledge bases in which to store data. Linking different resources on the web brings benefits to data accessibility and interoperability. With the advent of this new methodologies, logic languages to describe data have become a well-known standard. Structured conceptualizations of knowledge have taken the rise with the advent of the semantic web. Historically, semantic web languages anticipated the popularity of KGs representations because those languages were mainly introduced as mechanisms for knowledge sharing and interoperability.

Three are the core components of the semantic web:

- RDF: *Resource Description Framework* is a language that is used to describe web resources (e.g., the city Milan). A Uniform Resource Identifier (URI) is associated with each resource in such a way that it can be uniquely identified;
- SPARQL: A query language to query data stored in RDF. SPARQL can be used to query knowledge bases by using a simple syntax that is close to SQL;
- RDFS and OWL: These two are languages to model web ontologies. These languages define vocabularies and offer models to define data. For example, RDFS offers a set of entailment rules to derive new triples[3].

Data in the semantic web is generally represented with the use of RDF assertions, i.e., triples of RDF resources that are structured in a subject-predicate-object structure.

DBpedia

A KG that is frequently mentioned in this research work is DBpedia. DBpedia [2] is a project that aims to create a structured version of the content available on Wikipedia; the core idea is to create an endpoint for encyclopedic knowledge that can be easily accessed through the use of query mechanisms. Every page in Wikipedia has a unique identifier in DBpedia. DBpedia uses an *extraction framework* to extract information from a Wikipedia page and represent it using a structured and interlinked format. For example, from the page of the former president of the United States Barack Obama, the framework extracts the fact that Barack Obama was born in the state of Hawaii. DBpedia contains subject-predicate-object triples and it is one of the most popular knowledge graphs in the semantic web field. DBpedia contains lots of information; the DBpedia website states that in

3 https://www.w3.org/TR/rdf11-mt/#rdfs-entailment

DBpedia there are more than 4 million entities and more than 3 billion triples [4]. Entities in DBpedia are identified with the URI `http://dbpedia.org/resource/X`[5] while ontological elements are identified with the following URI `http://dbpedia.org/ontology/Y` (e.g., `http://dbpedia.org/ontology/Country` for the types and `http://dbpedia.org/ontology/location` for the properties[6]). We will use *dbr:* and *dbo:* prefixes to shorten URIs used to identify DBpedia entities and types of the DBpedia ontology (e.g., dbr:Milan). For example, if we access the page of the resource Milan[7] on DBpedia, we can see many possible connections links to other resources: for example, through the use of the predicate *country* we can access the resource *Italy*. This interlinked structure generates a graph that is generally referred to as the Knowledge Graph.

2.3 DISTRIBUTIONAL SEMANTICS

2.3.1 *Applications*

To reach the point in which Artificial Intelligence can be considered a real *intelligence* we must to find ways to give machines the possibility of understanding how language works and how communication can be made a relevant component and a relevant skill of autonomous algorithms. Intelligent agents could have an enormous impact on society and could greatly affect the economy through the accomplishment of many difficult human-like tasks [99]. To achieve this goal, we must be able to account for the representation of meaning. While logical approaches to model meaning in language have been extensively used, the last two decades have seen a rising wave of interest in the theory of *distributional semantics* an approach to semantics that advocates a "usage-based" perspective on the computation of word meaning. Distributional semantics is based on the assumption that the statistical distribution and the frequency of usage of words inside textual documents can reveal information about the meaning of words themselves. The intuition that drove the development of the algorithms that we will discuss in the future chapters is well described by a famous phrase that was pronounced by J.R.Firth's *"you shall judge a word by the company it keeps"*[45]. Distributional semantics has greatly influenced the natural language processing field and algorithmic implementation of this technique has been applied to different tasks as natural language inference [26], word similarity [100],

4 `https://wiki.dbpedia.org/about/facts-figures`

5 In general, the last part of the URI, namely X, is equal to the Wikipedia page name with few exceptions.

6 Note that ontological types have a capital letter in their name while properties are generally camelcase but start with a lowercase letter.

7 `http://dbpedia.org/page/Milan`

natural language generation [136], question answering [149] and others.

2.3.2 *Principles and Definitions*

Distributional Language

To better illustrate how this hypothesis works we give one famous example that was originally introduced by Stefan Evert [8]: suppose you do not know the meaning of the word "*bardiwac*" but you are given some examples of sentences in which the word "bardiwac" appears:

- He handed her a glass of *bardiwac*;
- Beef dishes are made to complement the *bardiwacs*;
- Nigel staggered to his feet, face flushed from too much *bardiwac*;
- Malbec, one of the lesser-known *bardiwac* grapes, responds well to Australia's sunshine;
- I dined off bread and cheese and this excellent *bardiwac*;
- The drinks were delicious: blood-red *bardiwac* as well as light, sweet Rhenish.

Even from this short list of sentences, one can capture a general understanding of the meaning of the word "bardiwac" as a red alcoholic beverage. This is the essence of distributional semantics: meaning can be found the context [77]. The definition of the concept context can vary widely across the different algorithms. The simplest case of context is co-occurrence: a word appears in the context of those words it co-occurs with. We expect the words *cat* and *kitten* to occur in similar contexts and thus being similar. Note that however also the words *cat* and *dog* could co-occur in some contexts, thus making the two words similar, but maybe less similar that *cat* and *kitten*. Words that occur in different contexts, such as *smartphone*, while not be similar to *cat*. This effect allows us to define a graded similarity that changes with the number of contexts. In fact we can summarize what we have said about the similarity by considering the following: *the degree of semantic similarity between two words* w_1 *and* w_2 *is a function of the similarity of the contexts in which* w_1 *and* w_2 *usually appear.* We in fact expect that the meaning of the words *dog* and *cat* to be similar, since both are domestic animals, have four legs, an owner, they eat, and so on. Another general and interesting idea behind the distributional hypothesis is that we can say that knowing a word does not mean to know its definition that comes from the dictionary. Instead, knowing a word is the fact of being able to use that word in the correct contexts.

8 https://esslli2016.unibz.it/wp-content/uploads/2015/10/dsm_tutorial_part1.slides.pdf

Strong and Weak Distributional Hypothesis

Lenci [77] explains the distinction between two different possible interpretations of distributional hypothesis: a *weak* distributional hypothesis and a *strong* distributional hypothesis of the distributional hypothesis. We follow Lenci [77] in the discussion of these approaches.

The weak distributional hypothesis is generally used to analyze text under a semantic point of view. It advocates the correlation between the meaning of a word and the contexts in which the word appears. This interpretation is the one that is mainly used currently and is the one on which different algorithms that will see in the next chapter are developed: we consider words and contexts and we generate representations for those words using a vector space. Roots of this idea can be found in the work of Harris [62], but also in Wittgenstein [139], advocating for the need to look at the use of words in the language to understand meaning.

The strong distributional hypothesis as a psychological point of view [94]. The strong distributional hypothesis starts from the idea that the distribution of words has a *causal* effect in the formation of the semantic representations of words. The idea in this case is to consider the relation with the cognitive associative mechanism of stimuli co-occurrence: the associations between words representations are stronger (and become stronger over time) in the human mind when they tend to appear in the same contexts. Obviously, it is possible to include a more general view of this hypothesis that includes a wider number of contexts that are not restricted to the linguistic one. Information about the sounds and the environment can also be included to account for a bigger cognitive understanding of these semantic similarity patterns. As we saw in the example of the word "bardiwac" it is really important to observe how learning and understanding words is strongly related to seeing the usage of the words in contexts.

Different arguments have been thrown against distributional hypothesis to outline (and comprehend) its limits. For example, if language is compositional (and also recursive, as formal semantic methods try to replicate), how can we include composition in distributional semantics? How can we deal with more complex linguistic phenomena such has hypernymy? This is a current field of investigation as it is not simple to account for composition in the context of distributional semantics.

In general, meaning in a sentence derives from the combination of the words used: we can combine words to express more about the world that surrounds us. Researcher that have argued against distributional semantics have often used the fact that a theory like distributional semantic should be able to cover aspects of formal semantics like *compositionality* and *inference*[9].

In the literature there do exist different work related to solving *compositionality* in distributional semantics. The simplest form of compositionality comes from summing representations: if we represent words meaning with vectors[10] we can combine them by summing the vectors. For example, the meaning vector of*"New York Times"* can be generated summing the single meaning vectors [78]. Different problems arise from this method but the easiest to comprehend is that sum is a commutative operation and thus shuffled phrases will all have the same vectors (i.e., *"the pen is on the table"* and *"the table is on the pen"* will have the same vector). Note that recently, more expressive algorithms based on related assumptions as ELMo [101] and BERT [36] have been introduced in the state-of-the-art to deal more complex language expressions. *Inference* has been addressed with the use of deep-learning models that also integrate information coming from distributional assumptions [26], but it can be also addressed with hybrid models that combine logic axioms and use vector as variables/terms [11, 50]. On this issue, in Chapter 6 we will show an example of how we can combine distributional representations with a logical system [20].

Despite all of these problems, the interest and the use of distributional semantics is growing in natural language processing due to the fact that corpora of large dimensions are now available on the web and that the hardware we posses is fast enough to allow us to use efficient algorithms. Moreover, many of these algorithms work in practice and have greatly advanced the field and thus, even if distributional semantics does not cover all the aspects of formal semantics, it is widely recognized as an important theory.

To summarize what we have said about distributional semantic we introduce the following general definition.

Definition 2 (Word Distributional Semantics) *The similarity between two words is a function of the contexts in which these two words appear.*

In Chapter 5, we will compare this definition with similar definitions of distributional semantics for entities and types of KGs.

From Words to Vectors

Given the definition of distributional semantics, it is now important to find means to apply this theory to computational models. Language

9 However, recent work [137] has suggested that distributional semantic should not be "blamed" for some of these limits, since it should be considered as a theory of meaning.

10 Something we will see in the next section, but it is relevant to quickly introduce this concept here.

is made of words that come from a vocabulary (we will use V to identify the vocabulary) and we need to find ways to account for the meaning of these words under a computational point of view. One of the first problems that Natural Language Processing as to deal with how to represent words meaning inside a computer or a computational systems by means and by the instruction of distributional semantics. Today, the most common way to represent word is to use *vectors* in a vector space: words are embedded in a multi-dimensional *vector space* and we can interpret them as points that can be compared. The process of "embedding" words in the vector space is what brought the community to call these representations "word embeddings". Word embeddings is in fact the general term that is used to refer to this kind of representations (the term "embedding" is used also for other types of embeddings, when we will discuss knowledge graph representations we will talk about knowledge graph *embeddings*).

To compute the similarity between two words in the vector space one can use a measure called cosine similarity. This measure computes the cosine of the angle between the vectors of the respective words: the smaller the angle between two vectors is, the more similar the vectors are. The cosine similarity is the dot product of the vectors that is divided by the magnitudes of the two vectors; it ranges from 1 for identical vectors to -1 for opposite vectors. The particular property of cosine similarity is that dividing by the magnitudes of the vectors normalizes the computation allowing us to ignore the vector length. This makes, in general, the similarity more robust to the effect of the frequency of each word in text (i.e., frequency often influences the length of the vectors [113]). Cosine similarity is not the only measure that exist to compute the similarity between word vectors but it is one of the most used. Nonetheless, the distance between points/vectors in the vector space can be computed using Euclidean distance or other higher-order distances.

Definition 3 (Cosine Similarity) *The cosine similarity between two vectors is given by the cosine of the angle between two vectors.*

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{u \cdot v}{||u|| ||v||} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Local and Distributed Representation

We can distinguish two different ways of representing words with vectors: we refer to the first as a local representation and to the second one as a distributed representations [44]. This distinction comes from one of the most vivid debates in the AI field during the '80s: how can we store conceptual information inside neural algorithms? Local representations are meant to represent a single concept with the activity of a single neural unit. On the other hand, distributed representations are meant to account for a pattern of activity of more neural units [63]. The last interpretation is particularly interesting, because it describe a model that is also close (in an high level sense) to how our brain might represent concepts (i.e., by patterns of neural activity).

Local representations map the i-th word of the vocabulary V to the vector $\mathbf{w}_i$ in a vector space $\mathbb{R}^n$, where n is the cardinality of V and the i-th element of $\mathbf{w}_i$ is set to 1, while all the other elements are set to zero. We generally refer to this kind of vectors as the *one-hot vectors*. For example, given the list of words of the vocabulary $V = \{\mathrm{the}, \mathrm{cat}, \mathrm{is}, \mathrm{on}, \mathrm{table}\}$, the word *"cat"* of the vocabulary V can be represented as a vectors of zeros with only one 1 in the position indexed by its own index in the vocabulary $\mathbf{w}_2 = \langle 0, 1, 0, 0, 0 \rangle$ (i.e., "cat" is the second element and thus the 1 will be in the second component of the vector).

Distributed representations encode the information over multiple components of the vector. There are two ways of doing this, by using components/dimensions that can be interpreted or not. Take for example, the symbol *"cat"*. We can use an explicit encoding that considers vowels, consonants and numbers to represent it with the vector $\mathbf{w} = \langle 1, 2, 0 \rangle$ (1 vowel, 2 consonants and 0 numbers). Instead a possible implicit vector can be $\mathbf{w} = \langle 0.5, 1.9, 2.9 \rangle$, where the components are not directly interpretable. In literature this distinction is usually referred to as *explicit* vs *implicit* vectors [79]. Implicit distributed representations are commonly used and thus we will refer to them when we talk about distributed representations.

Definition 4 (Distributed Representations) *Distributed representations are dense vectors; explicit distributed representations have vector components that can be interpreted, while the components of implicit distributed representation do not possess a direct meaning or interpretation.*

Note that the similarities between local representations are always the same because vectors are orthogonal.

Distributional Representations

What the above definitions do not provide is a way to effectively represent words in a way that can be used to do tasks. What is generally used to deal with words is the vector space model [127], in which words can be, for example, represented as vectors. Models that are based on distributional semantics aim to create representations in which, for example, similar vectors should represent similar words (i.e., words that occur in similar contexts). These algorithms take large amounts of text in input to create these vector representations. Figure 4 shows an example of what a vector space model built under the distributional hypothesis should create: "cats" and "dogs" are similar

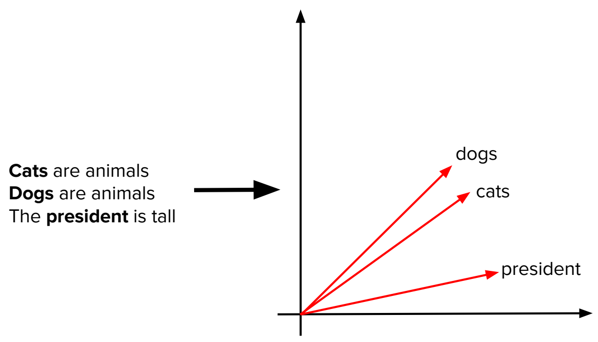


Figure 4: From sentences to the vector representation of words. We expect words that appear in similar contexts to have similar vector representations.

words and tend to occur in similar contexts (e.g., those shared by animals, those shared by house pets, etc...) and they tend to share fewer contexts with words like "president"[11].

There are different ways to generate these representations, we will explore some of them in the related work chapter (Chapter 3) of this thesis.

Definition 5 (Distributional Representations) *Distributional representations are a subset of distributed representations for which the value of the components of the vectors is based on the distributional hypothesis.*

Word Embeddings

In this section we give some details about an algorithm to create distributional representations of words that earned much success during the last years: word2vec. *Word2vec*, introduced by Mikolov et al. in 2013 [93] (more details about this algorithm are presented in the next section), is a word embeddings model, based on neural networks, capable of learning vector representations of words. The main advantage of word2vec over other methods is that it is efficient to train. Word2vec is often trained on large amounts of text and in the paper the authors also have presented negative sampling and hierarchical softmax that are two methods to greatly increase the speed of the algorithm. Word embeddings learned by word2vec exhibited a good capability at capturing syntactic and semantic regularities in a language [93], but it has been shown that these representations also capture bias in language [27].

This ability of capturing semantic regularities in language is what makes these algorithms really interesting: for example, they can be

11 However, it is possible to have a sentence in which both "president" and "cat" co-occur; this is why, as we see in the next chapters, the algorithms used are based on large amounts of text.

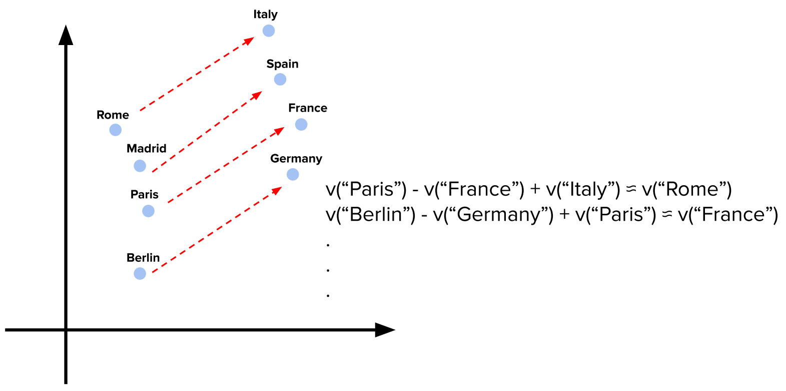

Italy
Spain
France
Germany
Rome
Madrid
Paris
Berlin
v("Paris") - v("France") + v("Italy") = v("Rome")
v("Berlin") - v("Germany") + v("Paris") = v("France")
.
.
.

Figure 5: Examples of linguistic properties in word embeddings in two dimensions. The "relation" capital-country seems to be stable in the space.

used to solve analogical tasks. Note that it was found that *analogy-making is one of the most important ways in which adults and children make sense of their world* [122], and analogies are very important for knowledge acquisition [130]. A particular kind of analogy, is the so-called *propositional* analogy: $a : b :: c : d$; the analogical reasoning task is to infer an unknown term, for example c, that is related to d in the same way that a is related to b [65]. Word embeddings exhibit this property of solving analogies. For example, word embeddings generated with word2vec can efficiently solve propositional analogies under the form "Paris":"France"::"Rome":"?" using vector operations). See Figure 5 for an example on how word vectors can be positioned in the vector space. The position in the vector space allows us to solve analogies by using simple sums and differences over vectors. Analogies like "Paris":"France"::"Rome":"?" are solved by computing the operation $v(\text{"France"}) - v(\text{"Paris"}) + v(\text{"Rome"}) \approx v(\text{"Italy"})$ where $\approx$ means that the answer of the analogy has to be searched in the neighborhood of the space where the operation points to.

Only recently the introduction of the aforementioned *word2vec* was key to develop the field[12]. Word2vec is a neural architecture that has been proposed in two different variants: Continuous Bag-of-words (CBOW) and Skip-gram (SG). Both architectures are simple feed-forward neural networks with one hidden layer, and they are trained over a large corpus of text. There are no non-linearities between the layers (except for a softmax function to compute the output scores of the network) and thus the projections are linear; this makes the algorithms less prone to overfitting and also faster to train. The training examples for the models are extracted from text and are generally based on the concept of target word and context words that appear inside the corpus within a distance from the target word: for example, in a

12 Note that the idea we presented before of vector representation that reflect the proximity of words has been widely adopted [12, 30].

sentence like “the cat is on the table”, “cat” might be the target word and “the”, “is”, “on”, “the”, “table” the context. CBOW gets the context words as input and the task on which it is trained is to predict the target words. Instead, SG is trained by considering the task in the opposite way: given the target word the model should predict the contexts. Once the models have been trained, the word embeddings are extracted from the first weight matrix of the neural network.

In CBOW, the training objective for a single training sample is to maximize the following conditional probability: given the context words $w_{k_1}, \cdots, w_{k_l}$ that appear around a target word w_j, predict the probability of observing that specific target word. CBOW computes the conditional probability $P(w_j | w_{k_1}, \cdots, w_{k_l})$ where $c \neq j$. On the other hand, the Skip-gram training objective is the opposite of the one we just presented for CBOW: in this case in fact we want to go from the target word to the context word $P(w_{k_1}, \cdots, w_{k_l} | w_j)$ where $c \neq j$, and thus the conditional probability is reversed.

Thus, following this training procedure, in word2vec each word is represented by a vector that encodes information about the co-occurrence of words. The window used to define the context of a words is an hyperparameter of the model.

More details about the Continuous Bag-of-words Model

CBOW learns weights in two different matrices, the target embeddings (i.e., the one that projects the input word to the embedded space) and the context embeddings (i.e., the one that projects from the embedded space to a space of dimensionality V). In CBOW, context embeddings $\mathbf{c}_j$ are encoded in what we call the *context weight matrix* $\mathbf{C}$ of the neural network, while target embeddings $\mathbf{u}_k$ are encoded inside what we call the *target weight matrix* $\mathbf{U}$. The importance of this difference will become clearer in later chapters, were we use this distinction between these two matrices to generate a novel algorithm that is based on CBOW. Starting from a corpus D that contains sequences of words $w_1, \ldots, w_n$, CBOW predicts the target word w_i from the *context window* of $M = 2L$ words around it, e.g. if we consider a window $L = 2$ the context is $\langle w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2} \rangle$ (we hereby use the indexes to represent the position of the words in the corpus). CBOW’s neural network architecture is described using a sample architecture in Figure 6. The input of the neural network is a set of words $w_1, \cdots, w_L$, embedded in as one-hot vectors $\mathbf{x}_1, ..., \mathbf{x}_L$. The matrix $\mathbf{C}$ is shared across all the input words. To compute the final vector in the hidden layer of our architecture, CBOW first takes the average of the context embeddings corresponding to the input words (thus combining the information of all the context in one single vector before re-projecting it to tue output).

Once an algorithm like CBOW has been trained, what we get is vector representations for words as the one in Figure 4, although we

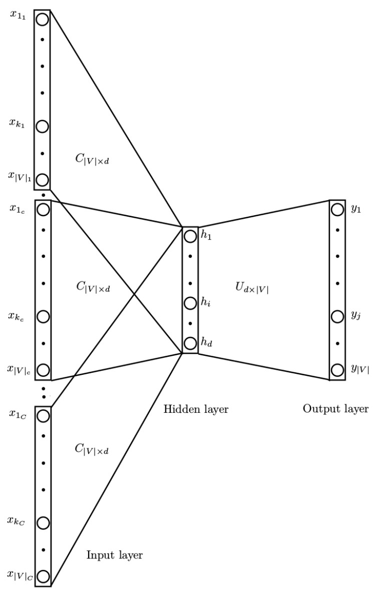


Figure 6: A schematic representation of the CBOW model.

recall to the reader that these representations also show linguistic regularities as in Figure 5.

Note that Mikolov et al. [93] have proposed an optimized way to generate word2vec embeddings that also considers the use of negative samples that are randomly sampled from text (roughly speaking it puts closer co-occurring vectors and puts far the target word and the negative samples). The main idea behind the model is the same since we are again dealing with target and context word, but the architecture becomes a bit different. Without the prediction in V dimension the model is indeed simpler. More details about this aspect of word2vec can be found in [106].

Distributional Semantics and Cognitive Sciences

Recent works analyze distributional semantics in psycholinguistic contexts. Distributional semantics has been found to provide representations that are *correlated with associative learning* [77]. As Lenci points out distributional semantic representations have been widely used to model different phenomena in psychology as similarity judgments, semantic and associative priming, semantic deficits, semantic memory deterioration[77].

Note that there now exists a rich literature related to cognitive aspects of word embeddings, in particular in the contexts of bias, stereotype and perception analysis [15, 27, 49]. For example, Caliskan et al. (2017) have found that word representation derived from distributional approaches capture strong components of stereotypical and biased language. Some of these biases are problematic, since they reflect on race or gender. Generally these algorithms learn representations implicitly without being forced to learn the bias, and this becomes an effect that scientists need to be aware of. Think for example of the usage of word embeddings in the context of curriculum analysis for hiring, we do not want algorithm to be biased towards gender and race.

2.4 SUMMARY AND COMPARISON

In this chapter, we have introduced two different approaches to account for meaning: one related to KB and mainly based on artificial languages, while the second one has a closer relationship with natural languages. The main difference between the two approaches lies in the way they represent meaning: logical approaches describe knowledge using a discrete set of symbols to form formulas and the semantics of symbols is defined using an interpretation function that maps formulas to a truth value; instead meaning in distributional approaches is in function of contextual information and it is represented using vectors of continuous values. Logic-based approaches require the definition of a set of symbols while distributional semantics derives representations starting from natural language expressions. On the other hand, logic allows symbol composition by definition, something that is a limiting factor in distributional representation for NLP. In this thesis we will also see models that are inspired by distributional semantics for KGs; these approaches are often called knowledge graph embeddings.

Methods based on distributional semantics allow us to create vector spaces in which each word is represented by a vector and, in general, similar vectors will be represented with similar components.

Note that in computer science, distributional representations are often used to initialize deep learning algorithms [26, 100, 136, 149]; groups of cognitive scientists are in general more interested in deeper aspects of these representations and in methods to identify bias and compare representations [15, 27].

2.4.1 *Notation*

Table 1 summarizes the most important mathematical notation that we use in this thesis. Other notation that is relevant to specific chapters will be introduced when needed. Note that we will also use subscripts and superscripts to generally identify elements of a sequence (e.g., w_i is the i-th word).

ADDITIONAL RESOURCES

- Ehrlinger and Wöß provide an interesting discussion on the many definitions for the term Knowledge Graph and analyze

Notation	Meaning
x	Letters identify scalars
$\mathbf{x}$	Bold letters identify vectors
$\mathbf{X}$	Bold capital letters identify matrices
w	The letter w will be used to identity words
$\mathbf{w}$	Bold w identifies the specific embedding $\mathbf{w}$
E	Set of the entities of a Knowledge Graph
e	The letter e will be used to identity entities
$\mathbf{e}$	Bold e will identify the specific embedding $\mathbf{e}$
R	Set of the relationships of a Knowledge Graph
r	The letter r will be used to identity relationships
$\mathbf{r}$	Bold r identifies the specific embedding $\mathbf{r}$
$< s, p, o >$	Knowledge Graph Triple

Table 1: Notation used in this thesis.

the differences between them. On formal semantics we suggest reading the book by Kearns that introduces the most important topics of the field [71]. Lenci has written a recent article that gives both an overview and an introduction to distributional semantics [78]; for a discussion that relates to the contributions of distributional semantics to the linguistic field, see [21]. On the recent discussion about the fact that distributional semantics should not be adequate to model formal semantics see the work by Westera and Boleda [137]. Ultimately, we suggest [54] as a reference book on deep learning and neural networks.

3 RELATED WORK AND CONTRIBUTIONS

We now describe the general idea of which are the goals of a comparable framework, while the formal definition of the framework will be given in Chapter 4.

CORPUS-BASED COMPARABLE FRAMEWORK GOALS We start by defining the goals of a corpus-based comparative distributional framework. As explored in Chapter 2, distributional models define a usage-based and context-based perspective on meaning [78] that generate representations that are correlated with associative learning [77] and that also have been found able to capture context-related bias [27]. If language is encoding important aspects of cognition and our associative knowledge, and language usage change across the contexts, the comparison of language usage in different contexts may reveal important associative knowledge patterns. Thus, if we want to reveal these patterns, we need ways to compare distributional representations that are generated from different text corpora. For example, we would like to model variation in temporal contexts [60, 73], and capture how words like “gay” and “amazon” move during time. At the same time, we would like to analyze corpora from different sources (e.g., different newspapers, different topics). Moreover, we would also like to introduce a reasoning methodology that allows us to combine distributional semantic with logical reasoning to further extend the comparison on a logical level (this theme will be explored in Chapters 6 and 7).

In the rest of this Chapter we analyze three state-of-the-art, and at the end of each section we will provide a comparison that analyze and provide justifications for some of our choices:

- We first explore related approaches for comparison analysis and evaluate some of the limits of current approaches; our analysis will be focused on temporal word embeddings approaches, since these approaches have to align representations to make comparisons and are the ones that we will use in the experimental section of Chapter 4;
- Since in this thesis we focus on a distributional model we will also explore other approaches of the state-of-the-art and discuss why we focused on an algorithm like word2vec;
- Finally, since part of the objective of this thesis is the analysis of unambiguous linguistic expressions (i.e.entity names), we

discuss the state-of-the-art approaches for representations of KG. Our aim here is to show that we need a particular type of knowledge graph embedding model that relies on entity linking.

3.1 COMPARING DISTRIBUTIONAL REPRESENTATIONS

We do a short introduction on methods used to compare distributional representation, and we will then focus on the comparison of temporal word embeddings and we will describe some methods for cross-lingual alignment (another area in which alignment is important). The method we cite generally possess the following characteristics: they are unsupervised (no direct supervision in training), they are corpus-based and they provide methods for distributional alignment).

3.1.1 *Overview on Semantic Comparison*

The use of distributed word embeddings for word-level comparative semantic analysis is based on the ability of these models to capture and express semantic biases present in the original texts, as shown by the works [23] and [27]. Both works were able to show that distributed representation contain human-like biases, either by using crowd-sourced experiments or by replicating the results of the Implicit Association Test developed in [56] that was run with humans.

From these results multiple works were published, generalizing the study of biases to the analysis of "semantic differences" in corpora over various dimensions such as geographically situated language [52], diachronic corpora [49, 60, 73, 125, 145], and corpora of different sources [52, 61]. The nature of the studied semantic differences was also expanded, encompassing gender [49, 148], racial and ethnic stereotypes [49, 52, 125] and sentiment analysis [61].

The various approaches for word-level semantic comparison between corpora can be grouped into two main categories: wordset association and representation shift.

WORDSET ASSOCIATION The original methodology introduced by [27] to measure bias investigates the association for two sets of opposite target words (e.g., *scientific* and *artistic* professions) against two sets of polarising attribute words (e.g., *male*-like and *female*-like words). Other works such as [49] use a generalization of the original method where the wordsets are replaced by their respective average representation and the second target set is substituted with its complement to background, meaning all words that do not exhibit some particular connotation (e.g., *immigrant*-like terms and everything else), enabling for single-target analyses. Another approach is the one used in [61], where a label propagation algorithm is used to induce a sentiment score from a seed lexicon (wordset) to the graph of neighbors. All these techniques do not require an alignment of the representations since they use a computed score for indirect comparison instead.

REPRESENTATION SHIFT For aligned representation it is possible to investigate the semantic change of a word by simply comparing its representations (in the next section we add more details on this topic); since the context of diachronic corpora is one of the most common applications for aligned representation models and the evolution of word usage is often the object of interest here the literature [60, 73] is especially rich. While most methods use simple vector similarity, others use first neighborhood search [145]. A hybrid approach is the one of [52] where a wordset-based approach is used on aligned representation, allowing for comparative semantic analysis on both diachronic and geographical dimensions.

3.1.2 *Comparing Distributional Representations: an Alignment Problem*

Due to the stochastic nature of neural networks, algorithms like word2vec always generates vector spaces with different coordinate systems. Thus, if one wants to compare word vector spaces generated from different sources (i.e., text from different periods, text coming from different newspapers) she cannot compare vectors directly. Figure 7 briefly illustrates the problem: representations generated from different sources cannot be directly compared.

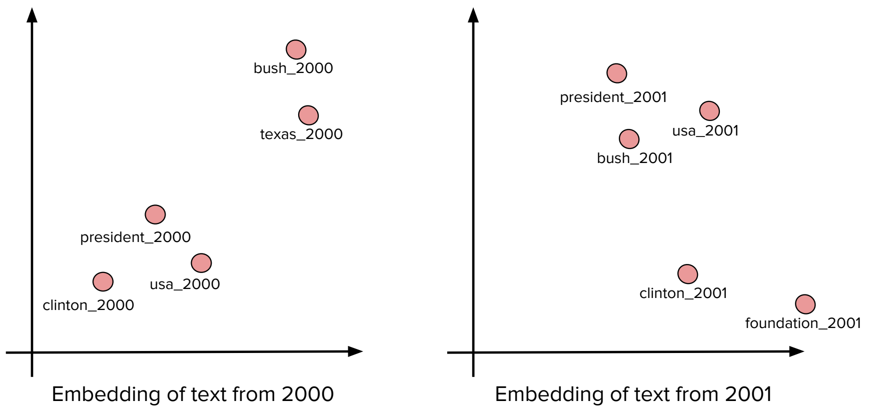


Figure 7: The problem with multiple distributional representations. There is no direct way of comparison between two slices. In 2000 and 2001 one Clinton and Bush were both presidents of the United States, but their vectors are not in similar close positions. Thus the representations of the two spaces are not comparable.

3.1.3 *Temporal Word Embeddings*

This alignment problem has been deeply studied in the field of temporal word embeddings, in which researcher wants to create vector representations of words in different periods of time to analyze semantic change [60, 73, 109] (see [21] for a brief overview over the semantic change topic). In this section, we mainly focus on the alignment of temporal word embeddings, but at the end of the section, we will cite other approaches for the alignment of word vector spaces, for example, cross-lingual approaches.

A Temporal Word Embedding Model (TWEM) is a model that learns *temporal word embeddings*, i.e., vectors that represent the meaning of words during a specific temporal interval. For example, a TWEM is expected to associate different vectors to the word *gay* at different times: its vector in the representation of the year 1900 is expected to be more similar to the vector of terms like *joyful* than its vector in 2005. By building a sequence of temporal embeddings of a word over consecutive time intervals, one can track the semantic shift in meaning that occurs in the word usage.

The training process of a TWEM relies on *diachronic text corpora*, which are obtained by partitioning text corpora into temporal “slices” [60, 145]. Because of the stochastic nature of the neural networks training process, if we apply a Word2vec-like model on each slice, the output vectors of each slice will be placed in a vector space that has a different coordinate system (see again Figure 7). This will preclude the comparison of vectors across different times [60]. A close analogy would be to ask two cartographers to draw a map of Italy during different periods, without giving either of them a compass: the maps would be similar, although one will be rotated by an unknown angle with respect to the other [115]. To be able to compare embeddings across time, their vector spaces corresponding to different time periods have to be aligned. The analogy of the compass will return again in Chapter 4 where we will actually use it to explain how our method for comparable distributional representations works.

Most of the proposed TWEMs align multiple vector spaces by enforcing word embeddings in different time periods to be similar [73, 108]. This method is based on the assumption that the majority of the words do not change their meaning over time. This approach is well-motivated but may lead, for some words, to excessively smoothen differences between meanings that have shifted along time. A remarkable limitation of current TWEMs is related to the assumptions they make on the size of the corpus needed for training: while some methods like [60, 120] require a huge amount of training data, which may be difficult to acquire in several application domains, other methods like [108, 145] may not scale well when trained with big datasets.

Different researchers have investigated the use of word embeddings to analyze semantic changes of words over time [60, 73]. We identify two main groups of approaches that are based on the strategy applied to align temporal word embeddings associated with different time periods: we refer to the one as *pairwise alignment* in which pairs of vector spaces are aligned, while we refer to the second one as *joint alignment* in which global constraints on the optimization process are used to generate vector spaces that are aligned after training.

3.1.3.1 *Pairwise Alignment*

Pairwise Alignment-based approaches align pairs of vector spaces to a unique coordinate system: [72] and [124] align consecutive temporal vectors through neural network initialization; other authors apply various linear transformations after training that minimizes the distance between the pairs of vectors associated with each word in two vector spaces [60, 73, 120, 147]. Essentially what we can learn is a matrix $\mathbf{M}_{s^1,s^2}$ that maps words from the vector space s_1 to the vector space s_2.

3.1.3.2 *Joint alignment*

Joint alignment-based approaches train all the temporal vectors concurrently, enforcing them inside a unique coordinate system: [6] extend Skip-gram Word2vec tying all the temporal embeddings of a word to a common global vector (they originally apply this method to detect geographical language variations); other models impose constraints on consecutive vectors in the Positive Point-wise Mutual Information (PPMI) matrix factorization process [145] or when training probabilistic models to enforce the "smoothness" of the vectors' trajectory along time [5, 108]. This strategy leads to better embeddings when smaller corpora are used for training but it is less efficient then pairwise alignment.

3.1.4 *Cross-lingual Approaches*

There are many different works that try to generate aligned representations by first defining anchors point that should not move in the space: a set of reference coordinate to which align everything, this is common in the cross-lingual or multi-lingual alignment community, in which the objective is to generate aligned representations of different languages. Multi-lingual lexicons are used to stabilize the position of some words in the vector space [42, 115, 142]: defining a set of anchors or a mapping dictionary requires domain knowledge and might make be a strong assumption in some contexts since it requires apriori domain knowledge. Thus, Facebook proposed an approach to align multi-lingual corpora without lexicon, this approach, named MUSE [31], leverages on adversarial training, to align different multi-lingual corpora.

COMPARISON Despite the differences between the pairwise and the joint alignment strategies, both strategies try to enforce the vector similarity among different temporal embeddings associated with the same word. While this alignment principle is well-motivated from a theoretical and practical point of view, enforcing the vector similarity of one word across time may lead to excessively smooth the differences between its representations in different time periods. Finding a good trade-off between *dynamism* and *staticness* seems an important feature of a TWEM (this property will be evaluated in detail in Chapter 4 were we compare our approach with state-of-the-art models). Finally, note that very few models proposed in the literature do not currently require explicit pairwise or joint alignment of the vectors, and these models all rely on co-occurrence matrix or high-dimensional vectors [9, 57]. Consider that these strategies assume temporal continuity, something that we cannot ensure in our comparative framework. Note that wordset approaches are not unsupervised and generally require to define specific lexicons, thus making it impossible to have an implicit comparative framework.

3.2 WORD EMBEDDINGS

We have already introduced word embeddings and distributional semantics in Chapter 2. Hereby we want to briefly mention other approaches that have been introduced in the state of the art to deal with word representations. We will skip the discussion on word2vec since it was already presented.

In the field of word embeddings, there is a distinction between methods that are based on counting [29, 35, 117]: for example, creating word-document matrix, word-word matrix for which each cell contains the co-occurrence of two words. To some of these matrices, dimensionality reduction techniques are often applied to generate low-dimensional representations [35]. We do not analyze these methods deeply since we focus on those methods that are now more popular that are based on neural-networks. We refer the reader to the work of Turney and Pantel [127] as a summary of word representations that precedes the advent of deep learning.

3.2.1 *Predictive Models*

Predictive models are usually based on neural-networks and are the most recent kind of models to create embeddings of words. They do not directly consider word counts and word-word co-occurrence matrix as older approaches, but instead, they are trained to *predict* the contexts in which the words usually appear inside a text corpus. Training these embeddings is a matter of initializing vectors and then using the information in the corpus (i.e., the word sequence) to predict a word from another. This is usually done using a sliding window that passes through all the text by generating training samples. These algorithms are trained using supervised methods, but are often referred as unsupervised, since we rely on this "trick" of using textual data that is available create the training samples. The spike in the interest for these methods has also been given by the fact that in the last years there is more availability of good computational power and that it is now easier to train neural network using modest hardware. We however remakr that wile there is some evidence that predictive models perform better on several tasks than count-based models [7] there are on the other hand, other authors that point out that under certain conditions, predictive and count-based models are similar [78].

Neural Network Language Model (NNLM) by Bengio et al. was one of the first models to be introduced to generate embeddings. A simple feed-forward neural network is used to learn word embeddings and a statistical language model jointly. The *language model* predicts the probability distribution of the occurrence of a word, given some previous words that appear in a text corpus. In this way, word embeddings are learned by scanning the text and predicting words.

While after NNLM different models were introduced, in 2013 it was the paper by Mikolov et al. that brought big interest and also a great advancement in the community of word representations. Mikolov et al. presented two new word embedding models, *Continuous Bag-of-Words Model* (CBOW) and *Skip-gram Model*. Generally called under the name word2vec, these two architecutres have greately influenced the field fo computational linguistics and of machine learning[28]. In fact, word2vec can be applied to any task in which language is involved such as question answering or document classification. Details on word2vec were given in Chapter 2.

3.2.2 *Beyond Word2Vec*

After Word2Vec many predictive models have been proposed, in this section we briefly describe there more recent models that are having a big impact in the community GloVe [100], ELMo [101] and BERT [36].

GloVe is a word embedding method that proposes an approach that combines the strengths of matrix factorization techniques with those of predictive models. GloVe base itself on the ratio of the co-occurrence between words, thus building probabilities. The training process is more efficient compared to the factorization methods thanks to an iterative optimization method that is based on weighted least squares that is meant to give less weight to rare co-occurring word. ELMo is a deep learning architecture that it is used to create a language model. The algorithm is trained over a large corpus and uses Bidirectional Language Model (BiLM) to solve a prediction task by considering the left context and the right context of any given word. A BiLM is composed of two Recurrent Networks [54] that combine the information of the left context and the information of the right context of a given word. It is clear that this fact makes ELMo an algorithm that is more advanced then word2vec: remember that word2vec does not consider the order of the words in text (and in fact one of its limit is how to deal with compositionality); ElMo is able to include information of the context by considering the sequence of the words. Also in ELMo, the network is trained to predict the context word based on the left or the right context. After the training procedure is completed, ELMo creates different vector representations for each word. We used ELMo in a recent research paper where we map words to their ontological concepts [129].

BERT is based on an ELMo-like architecture which introduces the random masking of words. This approach increases the generalization capacity of the model. Both BERT and ELMo consider char-based representations and are very difficult to train. ELMo creates a vector for *each* occurrence of each word and this make it difficult to train.

COMPARISON In Chapter 4 we will implement our corpus-based comparative framework by extending word2vec. This is due to the following reasons: (i) word2vec is efficient and effective and has been studied a lot in the literature; (ii) ELMo and BERT are also char-based, something that is not useful when we consider entity names; (iii) the most recent deep learning embedding algorithms are expensive to train both under a point of view of infrastructure (e.g., GPUs to buy) and of time (several hours or days), see the recent work by Strubell, Ganesh, and McCallum [118] on this aspect. Word2vec is simple and can be easily run on a large corpora even with modest hardware; (iv) while GloVe can be considered an alternative, in Chapter 4 we will use a key aspect of word2vec to implement our alignment methodology that is not directly applicable in GloVe.

3.2.3 *Evaluating Word Embeddings*

There are different possible ways to evaluate word embeddings [68]. In their work, [93] considered analogical reasoning tasks. Mikolov et al propose a method to automatically resolve analogy questions using the trained word embeddings, reaching state-of-the-art results. For example, the male/female relationship between words seems to be automatically learned by the model, which successfully resolves the analogy *king : man = queen : woman*. This kind of test is used to verify the quality of the alignment of the embedded representations. We recall that word embeddings are able to represent linguistic regularities by using vector operations like $v(\mathrm{bigger}) - v(\mathrm{big}) + v(\mathrm{small})$; operations result in a point in the space in which the nearest point should be the correct answer (i.e., $v(\mathrm{smaller})$). Word embeddings can be used to evaluate word similarity; there do exist different gold standards containing human-annotated pairs of words [68]; word embeddings are also frequently used to initialize deep learning algorithms on different tasks: instead of learning to solve a natural language processing task from a training set (e.g., question answering) pre-trained word embeddings can be used to start deep learning training approaches from an already generated representation of words. inference [26, 100, 136, 149]

3.3 KNOWLEDGE GRAPH EMBEDDINGS

We now explore related approaches in the field of knowledge graph embeddings. We categorize knowledge graph embeddings in those that use mainly the structure of the knowledge graph and those that use distributional information. Note that Knowledge Graphs can also be viewed as 3-dimensional tensors for which the dimensions are described by the subject, the predicate and the object. Tensor representation is high-dimensional and tends to suffer from the same limits that count-based model for word embeddings had. Nevertheless, models that use Tensor representation and Tensor decomposition for knowledge graph embeddings exist, as for example [98].

In the following, we will divide the knowledge graph embedding field in two main categories: structure-based embeddings and text-enhanced embeddings. However, note that there are different ways of categorizing knowledge graph embeddings [132].

3.3.1 *Components*

While different methods to create embeddings have been introduced in the literature, there is an underlying common base on which most of the methods can be connected to. This common base structures the definition of a knowledge graph embedding approach by considering three elements:

- Representation. The first step in a knowledge graph embeddings algorithm is the definition of the desired representation for the elements of the KG: for example, relationships can be defined as vectors [25] or as matrices [98].

- Scoring function. The next step requires the definition of a function that aggregates the information of the triples in such a way that it expresses the confidence about a given triple.

- Loss function. Finally, the loss function, a common component in machine learning models, is introduced to learn the representations.

Differently for what has been done in the previous section, we now give some details of an example of knowledge graph embedding model (TransE [25]) that will help the reader in understanding how the above components are combined.

3.3.2 *Knowledge Graph Embeddings with TransE*

We start this section with an example of an embedding model for knowledge graphs, namely TransE [25] a model for knowledge graph embeddings. TransE was introduced by Bordes et al [25] and, together with RESCAL [98], was one of the first approaches to start the entire knowledge graph embedding field. Fundamental idea of TransE is to embed entities in a space in which for each triple (s, p, o), the following formula holds for the respective vectors: $\mathbf{s} + \mathbf{p} \approx \mathbf{o}$. This means that given a triple $< \mathtt{Rome}, \mathtt{country}, \mathtt{Italy} >$ we expect the model to learn that for the respective vector $\mathbf{Rome} + \mathbf{country} \approx \mathbf{Italy}$ holds (see Figure 8).

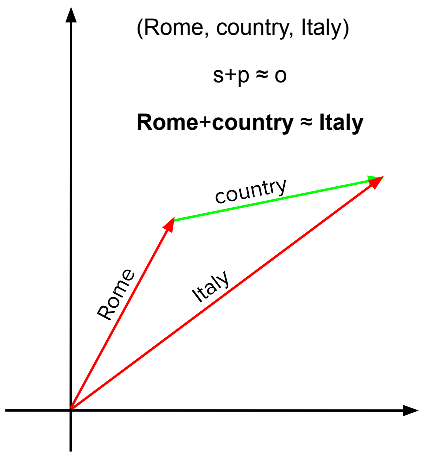


Figure 8: Schematic example of the idea behind TransE

Corrupted Training

Machine learning algorithms require positive and negative examples to learn and since knowledge graphs contain facts of the real world, they mainly contain positive elements. The technique to solve this problem, often used to train knowledge graph embedding models, is to generated corrupted/false triples starting from true triples: for example, given $< \mathtt{Barack\ Obama}, \mathtt{president}, \mathtt{USA} >$ one might generate the triple $< \mathtt{Barack\ Obama}, \mathtt{president}, \mathtt{Italy} >$. Thus a training triplet has either the head or tail replaced by another entity (but this does not happen for both at the same time) [25].

Learning Representations

If we consider the list of elements defined above, we can say that the way entities and relationships are represented is by using vectors. As for the score function, the model aggregates the information by computing the L_1 or the L_2 (d) norm between the pairs $(\mathbf{h}+\mathbf{p},\mathbf{o})$. Eventually, the model defines the following loss function to learn the representation for both entities and relationships; this loss function combines the score function and uses the methods of the corrupted triples to generate the representations.

$$\mathcal{L} = \sum_{s,p,o \in S} \sum_{s',p,o' \in S'_{s,p,o}} [\gamma + d(\mathbf{s}+\mathbf{p},\mathbf{o}) - d(\mathbf{s'}+\mathbf{p},\mathbf{o'})]_+$$

where $[x]_+$ is the positive part of x and γ is a margin hyper-parameter. And $S'_{s,p,o}$ is the set of corrupted triples. TransE only uses the knowledge that is contained inside the knowledge graph: the representations that are generated come from the **structure** of the knowledge graph. We use the term *structure-based* embeddings to identify embeddings that are based on the same assumption.

To summarize and compare TransE with the bullet point list that above, we can explain TransE as follows:

- Representation. $\mathbf{s}$,$\mathbf{p}$, and $\mathbf{o}$ are vectors in $\mathbb{R}^n$
- Score Function. L1 or L2 distance, $s(\mathbf{s}+\mathbf{p},\mathbf{o})$
- Loss Function. $\sum_{s,p,o \in S} \sum_{s',p,o' \in S'_{s,p,o}} [\gamma + d(\mathbf{s}+\mathbf{p},\mathbf{o}) - d(\mathbf{s'}+\mathbf{p},\mathbf{o'})]_+$

3.3.3 *Structure-based Embeddings*

Related to TransE, a plethora of approaches has now been introduced in the knowledge graph embeddings fields [25, 41, 81, 98, 126, 133, 134, 138, 140, 141]. A central hypothesis of these works is that relational information contained in the KG is structured. Originally, the seminal work of this technique is found in the paper of Bordes et al. [24] in which each relation is represented by a pair of linear transformations on the vector entities that the relation links. Following this approach, several works use relation representation in similar forms as translations [25], quadratic forms [69], hyperplane projections [134], or entity clustering [81]. A different point of view has been given by the conceptual spaces community [67]. In this model, entities sharing types are mapped to be closer in the vector space using space constraints.

As explained above, these embeddings are usually obtained by minimizing the global function (often referred to as *score function*), which encourages the proximity of related pairs of entities in the knowledge graph. More recently, methods like CompleEx [126] that use a complex space to represent entities and relationships have been effectively used in the link prediction task. Other approaches combine structural knowledge with logical rules to generate better representations [58].

Hyperbolic Embeddings

Generally, the models we just described require an high number of dimensions to generate good embeddings. This limits comes from the fact that euclidean geometry might not be the best geometry to fit these graph based representations. Given this limit, researchers have used hyperbolic geometry to account for better representation of graph structures. Non euclidean geometries are based on the rejection of Euclid's postulate about *parallel lines*: *given a plane, a line and a point that is not on the line, at most one line that is parallel to the given line can be drawn through the point*. On the other hand, in *hyperbolic geometry*, there are always at least two parallel lines given a line and a point not on it. These approaches have been now widely used to represent tree-like structures [87, 97, 111, 119] and are having valuable recognition also in Natural Language Processing [74, 123]. Hyperbolic embeddings are in fact really useful when we need to learn embeddings of tree/ontological structures. Differently from these approaches, in our framework we learn embeddings of types under distributional conditions (see Chapter 5). In a recent work, we show that these two ways of generating the embedding of an ontology are different and can be effectively combined to account for two different types of information [129].

3.3.4 *Distributional Knowledge Graph Embeddings*

Models that use co-occurrence from documents that contain links are now commonly used in literature [10, 138]. For example, Wiki2vec [138] uses word2vec over Wikipedia text and generate the representation for both entities (by looking at links co-occurrence) and words. We can refer to approaches that base themselves only on the co-occurrence of entities in text as *Directly Distributional Knowledge Graph Embeddings*. We call this model *directly distributional* because they only rely on the distributional hypothesis to generate the representations.

Another prominent model in this category is RDF2Vec [104]: it uses an approach that combines techniques from the word embeddings community with knowledge graphs in RDF. In fact, it generates embeddings of RDF entities and relationships by first generating a virtual document that contains lexicalized walks over the graph and then use word embeddings algorithm on the virtual document to create the representation. The representation is thus based on entity-relationship-entity co-occurrence in the walk. We can refer to this approach as *semi-distributional* since it is based on the distributional hypothesis, but it does not start from natural language sentences.

There instead exists a variety of models that make use of textual information [41, 67, 133, 135, 138, 140, 141] to enhance the performance of knowledge graph embeddings techniques, we refer to these models as *Mixed Knowledge Graph Embeddings* [132].

For example, mixed KG embedding (TEKE) [135] focuses on Wikipedia inner links and replaces them with Freebase entities and then use word2vec on the generated corpus. No specific use of online annotators is run[1]. Pre-trained representations can be used to initialize knowledge graph embeddings and to generate representations that can, in some cases, outperform other methods [144].

Jointly [133] is an embedding method in which textual knowledge is used to enrich the representation of entities and relationships. In this work, both entities and words are mapped to a common vector space and vectors associated with words and entities which represent a common concept are forced to be closer in the vector space. Jointly is used on an analogical reasoning task: in which the analogies are solved using words, and their results do not considerably improve those of a standard the skip-gram baseline. Semantic Space Projection (SSP) [140] is a method that learns the representation of triples by also considering the description of the entities. Description-Embodied Knowledge Representation Learning (DKRL) [141] includes the description of the entities in the representation. DKRL uses a convolutional layer to encode the description of the s entity into a vector representation and use this representation in the loss function. Words vectors coming from the entity description can be initialized with the use of word2vec embeddings. Thus, this method represents each entity by considering the composition of the words in its description and not by considering the general co-occurrence of an entity with others. One key advantage of DKRL [141] is that it offers the possibility of doing zero-shot learning of entities by using the description of the entities themselves.

Directly distributional embeddings models tend to capture the semantics of language within the text. This is different from the structural embeddings approach. Similarity from textual embeddings has been compared with the similarity coming from entity embeddings [84], showing limitations of structured knowledge graph embeddings in retrieval tasks. Thus, Directly distributional embeddings are able to capture information that structural knowledge graph embeddings do not capture.

1 Nevertheless, the authors mention that the use an annotator is a possibility.

Table 2: Some approaches and the task they have been tested on

Method	Analogical Reasoning	Link Prediction	Triple Classification
TransE [25]		X	
TransH [132]		X	X
TransR [82]		X	X
ComplEx [126]		X	
NTN [116]			X
Jointly [133]	X		X
TEKE [135]		X	X
KALE [58]		X	X

In this work, we want to propose a method that generates entity representations that can be compared. We need a model that can generate embeddings from different text sources.

3.3.5 *Evaluating Knowledge Graph Embeddings*

Knowledge Graph Embeddings are generally evaluated on link prediction tasks. This means that given a triple $(s, p, ?)$ the model has to find the correct o. Usually, the model produce a ranking list containing all the estimated $\hat{o}$ entities. From this list, corrupted triples used in training and other correct triples (i.e., $< \mathtt{Italy}, \mathtt{hasCity}, ? >$ has multiple answers) are removed [25]. Another task often tested in the context of knowledge graph embeddings is triple classification [116] (i.e., detecting if a triple is true or false). Generally, analogical reasoning is a task that is not always tested in the context of knowledge graph embeddings and only Jointly [133] has been used to generate better word representation that could solve analogical reasoning tasks. In Table 2 we show some approaches with the respective task they have been tested on.

Limits of the recent results that come from KG embeddings have been outlined in a recent experimental paper [70], where the authors suggest that the variability in performance by the models might be more related to careful parameter tuning than to the novelty and the complexity of new architectures. For example, SSP [140], one of the mixed approaches, had lower performance than structured well-tuned approaches like ComplEx [126].

COMPARISON In Table 3 we summarize where our approach locates itself in the literature by citing other approaches with respect to our task of corpus-based comparative analysis. Structure-based approaches cannot be used because they do not consider text and they just embed the graph. Semi-distributional methods like RDF2Vec [104] do not consider natural language expressions. Mixed methods are generally meant to be used in other tasks as link prediction. Directly distributional methods can be applied only on already on text that contains hyperlinks and thus have limited applicability. In the context of this comparison, it is important to cite an approach by Iacobacci, Pilehvar, and Navigli[66] that annotates text using the linker Babelfy[2] and then generate embedding using word2vec. This approach is indeed much similar to what we want to do. The two main differences are that while their approach is focused on a model that embeds word senses we are more interested in embedding entities and types of a knowledge graph. Moreover, we will focus our experiments on the use of an open-source entity linker, which is DBpedia Spotlight [88] that is free to use[3]. Anyway, while our method does not deeply extend the others, it has the advantage of relying on entity linking techniques for knowledge graphs. This is useful because we want to be able to use it on different text sources with the aim of generating representations that are supported by associative learning [78] and that can contain contextual bias [27]. Moreover, we will show in Chapter 5 that this model can be used for both entity and type representations.

Method	Dependency	Applicability
Structure-based (e.g., [25, 98])	KG	No
Directly Distributional (e.g., [10, 138])	Text and hyperlinks	Only to hyperlinked text
Semi-distributional (e.g., [104])	KG	Not directly
Mixed (e.g., [140, 141])	KG and text	Not directly
Our approach (TEE [17])	Text and entity linking	Yes

Table 3: Comparison between approaches to generate entity representation for corpus-based comparable embeddings in relation to the applicability to our task.

3.4 CONTRIBUTIONS OF THIS THESIS

3.4.1 *Corpus-based Comparable Distributional Framework*

In this thesis we introduce a framework to compare representations that are generated under the distributional hypothesis. To do this, we need a method that allows us to align distributional representation in a way that is closely related to what other researchers have done in the context of temporal word embeddings [60]. Our approach is effective and efficient and in Chapter 4 we will show that its performance are better than other state-of-the-art approaches; moreover we will show that this model is not constrained by temporal assumptions and thus it can be used to align also text that comes from different sources, thus being a suitable model for a comparative framework.

2 http://babelfy.org/
3 Note that in Chapter 7 we will show some results that are based on the use of another annotator, that is SpazioDati's Dandelion.

3.4.2 *Distributional Embeddings with Entity Linking*

To represent and compare entities in text we need a method that allows us to generate entity representations. In this thesis we use a distributional knowledge graph embedding model for entities and types that relies on entity linking techniques. While this method is not far from models already defined in the state-of-the-art, the fact that is entirely based on entity linking allow us to use it on any text.

3.4.3 *Reasoning with Distributional Embeddings*

We provide a method to reason over distributional representations. This method is based on LTNs a prominent model in the neuro-symbolic field that allows us to do reasoning over vector space, allowing us to make logical comparisons between different representations. In Chapter 6 we will describe the method we use and also do a short summary of related work of the field that we did not cite here to focus on distributional approaches.

SUMMARY OF CONTRIBUTIONS Contributions of this research work are spread among three different levels:

- a methodology for efficient and effective alignment of distributional representations that allows us to compare representations that are generated from different corpora (Chapter 4);
- the definition of a purely distributional model for knowledge graph's entities and types (Chapter 5) that can be used with the previous contribution to account for the comparison of entity names;
- a method to perform logical reasoning over distributional representations (Chapter 6) that is based on the representations of the entities of the previous contribution and that will allow to do comparative reasoning with the use of axioms.

Part II

CORPUS-BASED COMPARISON OF DISTRIBUTIONAL MODELS OF LANGUAGE AND KNOWLEDGE GRAPHS

4 CORPUS-BASED COMPARATIVE DISTRIBUTIONAL EMBEDDINGS

In this Chapter, we introduce Compass-Aligned Distributional Embeddings (CADE) the main component of our comparative framework, that is, a model to effectively align and compare distributional representations that are generated from a collection of text corpora [37]. We experiment this model with state-of-the-art approaches in the temporal domain (a domain in which is important to align distributional representations), and we also show that Compass-Aligned Distributional Embeddings (CADE) is not constrained by temporal assumptions (as many of the baselines that we are going to consider) and can thus be applied on corpora with other characteristics.

The alignment approach for distributional representation introduced in this chapter has been published as:

- Di Carlo, V., Bianchi, F. and Palmonari, M. (2019). **Training Temporal Word Embeddings with a Compass**. In AAAI.

While this work was focused on temporal data, in this Chapter we formalize the distributional framework and we give novel details about the alignment model showing that it is general enough to handle text that comes from different sources.

4.1 CORPUS-BASED COMPARATIVE FRAMEWORK

If language is encoding important aspects of cognition and language usage changes across the contexts, the comparison of language usage in those contexts may reveal important knowledge patterns. A corpus-based comparison framework should provide methods to explore the difference in meaning across different contexts. For example, consider a corpus that contains texts written in British English and a corpus that contains texts written in American English. We might ask ourselves if the word "flat" has the same meaning in British and American English. Frequently, in British English, the word "flat" is used to describe a place where people live, while in American English it is more often used to indicate flat surfaces. In fact, the equivalent of "flat" in American English can be considered the word "apartment". While under a lexical point of view the words are the same, their semantic meaning changes. An intuitive way to view this is under an analogical point of view: "flat" is to British English as "apartment" is to American English.

A comparative framework should help us in computing and exploring the semantic differences in meaning that words assume in

Corpus One (D1)	D1 Input	Corpus Two (D2)	D2 Output
US	apartment	UK	**flat**
US	labeled	UK	**labelled**
US	gasoline	UK	**petrol**
1987	reagan	1997	**clinton**
1987	walkman	2007	**ipod**

Table 4: Some example of comparisons we would like a comparative model to be able to find find. These examples can also be viewed as analogies between different corpora.

different corpora. Table 4 shows some examples of possible comparisons that we would like to make between different corpora in different domains. We show two possible pairs of domains in the Table, one that considers the differences in semantic meaning of words between American and British English and another one that considers the differences in semantic meaning of words during different years. We might want to compare tokens that come from corpora of different time periods [60, 145] or of different topics (e.g., games, business, politics, ...) [6].

We already saw that word meaning can be represented with the use of dense vectors (i.e., embeddings) and that generating embeddings of domain-specific corpora results in representations that encode implicit biases of the text [27]. It is necessary then, to find models to compare different representations created from different sources, but this is not possible with standard word embeddings due to an alignment problem.

4.1.1 *Recap: The Alignment Problem*

When we generate word embeddings from different corpora it is not possible to directly compare a word vector of the first corpus with a word vector of the second corpus. This limitation is due to the intrinsic stochasticity of neural networks that eventually generate vector spaces with different coordinate systems. It is thus important to find a way to align these representations, otherwise, the comparisons are not possible. Figure 9 summarizes the problem we encounter when we use, for example, the standard word2vec algorithm on two different corpora (in the Figure we consider text that was written in two different periods). It is not possible to compare the vector of the two slices: the vector of the word "president" occupies to completely different positions in the vector spaces.

Essentially, what we would like our model to do is to allow us to map words' meanings used in two different corpora using a function

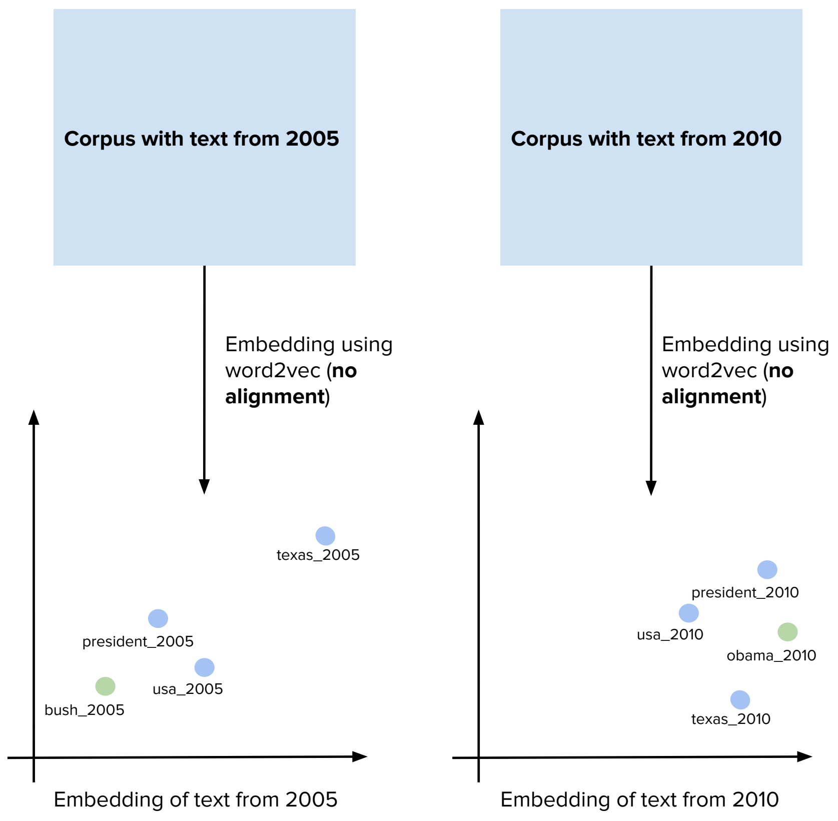

Corpus with text from 2005
Corpus with text from 2010
Embedding using word2vec (**no alignment**)
Embedding using word2vec (**no alignment**)
texas_2005
president_2005
usa_2005
bush_2005
president_2010
usa_2010
obama_2010
texas_2010
Embedding of text from 2005
Embedding of text from 2010

Figure 9: Word embeddings generated from two different corpora are not directly comparable. The word "president", for example, occupies to completely different positions in the vector spaces.

ϕ. The idea is depicted in Figure 10, where we show a temporal-based comparison and a language-based comparison.

4.1.2 *High-level Description of the Framework*

As inputs to our comparative framework, we have a collection of texts that can be sliced according to different context-based criteria. There are several possible different slicing that we could consider: text can be divided by considering the period of time in which it was written or by considering categories (e.g., business, sports, etc...).

As an output, we want our model to provide aligned representations, one for each corpus in input, that can be used to compare different words.

4.1.3 *Definition of a Comparative Framework*

In the following definitions, we use the more general term *token* to refer to words and entities names using a single term. We in fact remind that as explained in Chapter1, we want to apply our frame-

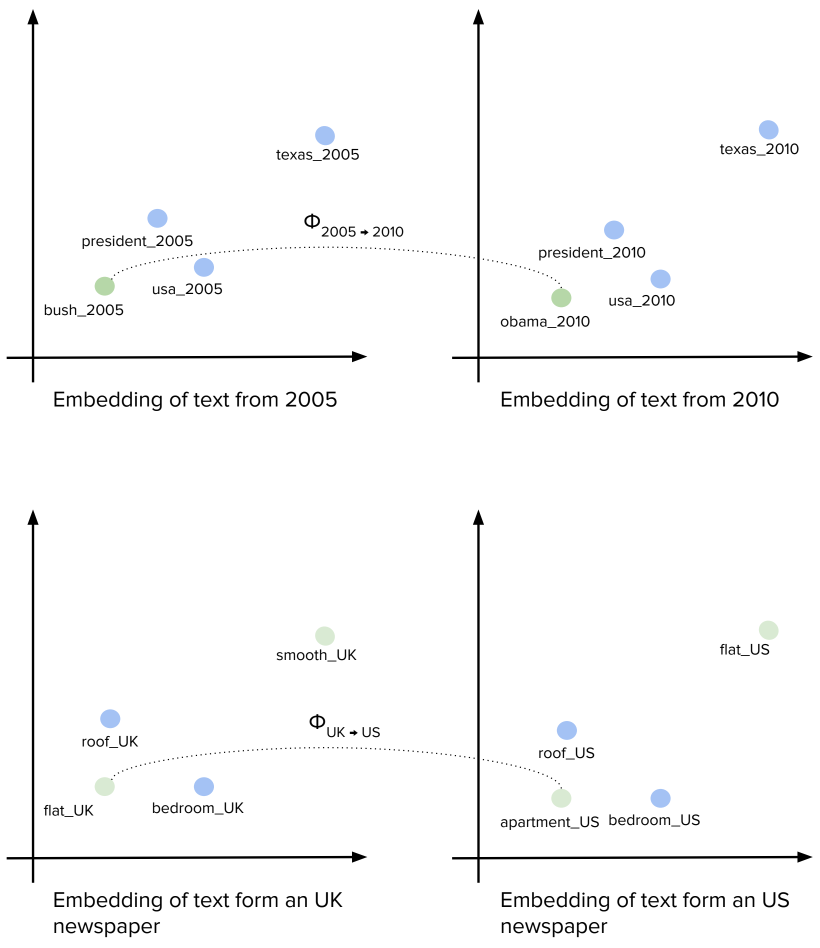


Figure 10: A comparative framework should allow us to compare words from different corpora.

work to distributional models of words and elements in a knowledge graph (a distributional model for representing entities and types of a KG will be introduced in the Chapter 7).

The word2vec algorithm that we use in this thesis is based on tokenization (e.g., separate tokens considering the space character) and thus it will not make differences between words or entities when they are tokens.

The goal of a comparative framework is to support the comparative analysis of the semantic meaning of words in different corpora. Thus, a comparative framework should allow us to find correspondences between different corpora.

In our comparative framework we have access to a **collection** D that consists of various corpora $\{D^1, \ldots, D^i, \ldots, D^n\}$. When considered as part of a collection, we refer to each corpus D^i as to the **slice** i of the collection. V is the vocabulary of the collection D, that is, the set of tokens that occurs in the collection D. With w_k we instead denote as the k-ith token in the vocabulary.

V^i represents the vocabulary restricted to the slice D^i, i.e, the subset of V that contains only tokens that occur also in D^i. While $\mathbf{C}^i$ is the vector space associated with the slide D^i, i.e., the set of vectors associated to V^i (the vocabulary restricted to the slice i).

$\mathbf{c}^i_{(k)}$ is instead the the vector associated to the k-ith token in the $\mathbf{C}^i$ vector space, i.e., the vector space associated to the slice i. We refer to this vector also as to the i-th slice-specific vector (or, equivalently, i-th corpus-specific vector) of the token w_k, and we refer to $\mathbf{C}^i$ as to i-th slice-specific vector space (or, equivalently, i-th corpus-specific vector space).

Observe that while we can define a bijection between V^i to $\mathbf{C}^i$, i.e., each token that occurs in a slice i is associated with a slice-specific vector, the mapping between V and a vector space $\mathbf{C}^i$ is partial, because a word $w \in V$ may not occur in V^i and thus not have a slice-specific vector in $\mathbf{C}^i$.

When we do not need to refer to a specific word and the context will make the notation clear, we will also use the simplified notation $\mathbf{c}^i$ to refer to the vector of a word w in the $\mathbf{C}^i$ vector space and we will use the lower index to identify the element of a sequence of words.

Given the different distributional representations $\mathbf{C}^i$ we need to find corresponding items in pairs of slices. We thus need a correspondence function that allows us to map a token in one slice to the element it corresponds to in another slice (see, for reference, Table 4).

Definition 6 (Correspondence Function) *Given two slices D^i and D^j with vocabularies V^i and V^j, we define the correspondence function $\phi_{D^i \to D^j}$ as a function that for every token $w_k \in V^i$ associates a token $w_h \in V^j$ if and only if $\mathbf{c}^i_{(h)}$ is the most similar vector to the vector $\mathbf{c}^i_{(k)}$.*

Observe that given two different vector spaces $\mathbf{C}^i$ and $\mathbf{C}^j$ we can always compute a vector-based similarity function between their vectors. However, if the vector spaces are not aligned, i.e., if the vectors are generated using different coordinate systems, the vector-based similarity cannot be considered a proper semantic similarity measure. The alignment problem consists therefore in ensuring that two vector spaces $\mathbf{C}^i$ and $\mathbf{C}^j$ are aligned so that geometric similarity measures computed between vectors of the two spaces are semantically meaningful (see Section 4.1.1 for the discussion on the alignment problem).

Note that there are two possible outcomes from the correspondence function in case of aligned embeddings: given two slices D^i and D^j, the $\phi_{D^i \to D^j}$ function can associate to a token w_k a (i) different token w_h (i.e., $\mathbf{c}^j_{(h)}$ is the most similar vector to $\mathbf{c}^i_{(k)}$) or (ii) the same token w_k (i.e., $\mathbf{c}^j_{(k)}$ is the most similar vector to $\mathbf{c}^i_{(k)}$). We expect tokens that have the same semantic meaning in different slices to have highly similar vectors. Instead, tokens for which the meaning changes in two different slices, should have different vector representations. This effect is generally referred to as semantic shift.

Examples of shifted tokens can be the word "gay", that has changed meaning over time [73] or the word "lift" in British English and American English or the entity name "dbr:Google" that has moved from a search-engine only company to a broader-range company during the years. This effect is often referred to as *meaning shift* in the context of temporal word embeddings. At the same time, we expect a word like "president" to have a more stable meaning in two slices that contains general newspaper data.

Definition 7 (Shifted Token) *A token w occurring in a corpus D^i and in a corpus D^j is shifted if and only if $\phi_{D^i \rightarrow D^j}(w_k) = w_h$ and $h \neq k$, i.e., if and only if its correspondence function returns a different word.*

Finally, we give the definition of a comparative distributional framework.

Definition 8 (Comparative Distributional Framework) *A comparative distributional framework is a quadruple $\mathcal{F} = (D, C, V, \Phi)$, where $D = \{D^1, \dots, D^n\}$ is the collection of slices of text, $C = \{C^1, \dots, C^n\}$ is the set of corpus-specific distributional models generated from the slices, such that each C_i is generated from the slice D^i. $V = \{V^1, \dots, V^n\}$ is the set of vocabularies restricted to each slice and Φ is a set of correspondence functions $\phi_{D^i \rightarrow D^j}$, defined for all i and j, i.e., between every slice pair D^i and D^j in the collection.*

This is the description of the atomic units of a comparative framework. In the rest of the Chapter, we will make mainly reference to words, but has introduced above, since we rely on tokenization, the model we describe is general enough to handle also other types of tokens as entity names, as we will see in Chapter 7. In fact, in Chapter 5 we will introduce a general method to handle entity names of a knowledge graph.

4.2 Compass-Aligned Distributional Embeddings (CADE)

As outlined in the previous section, an effective comparison between different slices has to pass from an effective alignment procedure of the vector spaces. In this section, we introduce Compass-Aligned Distributional Embeddings (CADE), a model to align and make comparable different distributional representations obtained from different slices of text. Our model is based on a generalization of the CBOW model introduced in 2013 by Mikolov et al.

Main Idea: looking it from a more general perspective, our model is based on a simple but effective idea. We first train a CBOW model on a general text and then we use the **trained second matrix** of the CBOW network to initialize the second matrix of a novel CBOW model that we are going to use on the text of each slice. We freeze this second matrix of each CBOW model; this in fact constraints the CBOW models in such a way that after training all the embeddings that we extract from them are aligned.

We hereby restate the inputs and the outputs of our comparative framework: starting from a **collection** D that consists of multiple slices of text $\{D^1, \dots, D^i, \dots, D^n\}$ we want to generate n aligned representations $\mathbf{C}^i$ that can be compared. We first describe the two assumptions that drive our method and then we will show in detail how these two assumptions can be implemented.

4.2.1 *The Compass Heuristic*

Our approach is inspired by an assumption made in previous work by [73]: the majority of words does not change meaning over time: while some words have shifted their meaning over time (e.g., "amazon", "apple", "gay"), most of the words tend to have a stable meaning. Nevertheless, we believe that this assumption is also true for other corpora divided by topic: two sources that use the same language might use the same word in a different way (i.e., think of the differences between British and American English), but most of the words used for communication have a strong shared meaning (otherwise it would be impossible for a British English speaker to understand an American English speaker).

From this assumption, we derive a second one: we assume that a shifted word, i.e., a word whose meaning has shifted, appears in the contexts of words whose meaning changes slightly. However, differently from the latter assumption, we believe that our assumption is particularly true for shifted words. For example, the word *clinton* appears during some temporal periods in the contexts of words that are related to his position as president of the USA (e.g., *president*, *presidential*, *administration*); on the contrary, the meanings of these words have not changed much (i.e., they still have the same meaning). In a sense, *the words in the context of the shifted word are generally more stable*.

Generalizing this assumption, we can say that we have the same effect with the word "petrol" when its meaning shifts from British English to American English (i.e., the American English equivalent is "gas/gasoline"). This word will appear in contexts related to "cars" and "economy", that are words that have a more stable meaning.

4.2.2 *Extending CBOW*

We adapt this heuristic to the CBOW model. We recall that this model is based on two matrices, a target matrix, and a context matrix (see section 2.3.2 for more details), this distinction is important since it is one of the main points of our algorithm. In general in CBOW, after training, we extract the word embeddings from the context matrix.

Considering again the two assumptions we outlined in the previous section, we can consider the target embeddings as static, i.e., to freeze them during training, while allowing the context embeddings to change based on co-occurrence frequencies that are specific to a given slice. Thus, our training method returns the context embeddings as word embeddings.

Before freezing the static embedding, we need to initialize them. We use a first training of the CBOW model on the concatenation of the slices of the collection D, thus building a general embedding that will be used to identify a coordinate system for all the other embeddings. This embedding is the **compass embedding**.

The process is thus twofold: (i) we first generate a *general* representation of the embeddings, computed on all the data we have, and then (ii) we use this representation to initialize the specific CBOW model by also freezing the target matrix with the pre-trained embeddings from step (i).

The frozen embeddings act as a *compass* and makes sure that the temporal embeddings are already generated during the training inside a shared coordinate system. In reference to the map drawing analogy [115] (Section 3.1.3 introduced this analogy), CADE draws maps according to a compass, i.e., the reference coordinate system defined by the compass embeddings.

4.2.3 *The CADE Model*

In this section we formally introduce CADE, that is our extension of CBOW that can be applied to sliced corpora to generate a set of aligned word embeddings. Note that CADE can be implemented on top of the two Word2vec models, Skip-gram and CBOW. Here we present the details of our model using CBOW, since we empirically found that it produces models that show better performance than Skip-gram when doing experimental evaluations on small datasets.

Training

The training process of CADE is divided into two phases, which are schematically depicted in Figure 11.

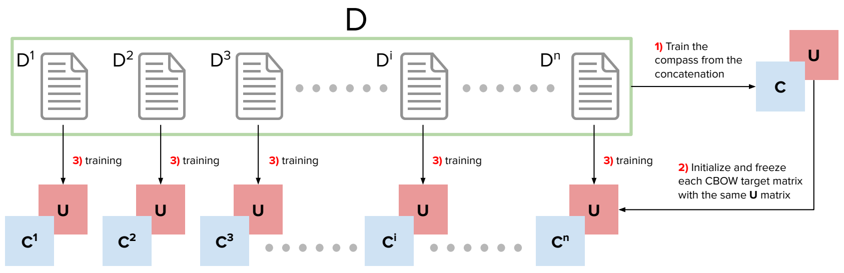

D
D1
D2
D3
Di
Dn
1) Train the compass from the concatenation
U
C
3) training
3) training
3) training
3) training
3) training
2) Initialize and freeze each CBOW target matrix with the same U matrix
U
C1
U
C2
U
C3
U
Ci
U
Cn

Figure 11: The CADE model.

(i) First, we construct two *compass* matrices **C** and **U** by applying the original CBOW model on the concatenation of the document in the collection D; **C** and **U** represent the set of *compass context embeddings* and *compass target embeddings*, respectively. (ii) Second, for each specific slice D^i, we construct the context embedding matrix $\mathbf{C}^i$ as follows. We initialize the output weight matrix of the neural network with the previously trained target embeddings from the matrix **U**. We run the CBOW algorithm using the specific slice D^i. During this training process, the target embeddings of the output matrix **U** are not modified (i.e., we *freeze* the layer), while we update the context embeddings in the input matrix $\mathbf{C}^i$. After applying this process on all the slices D^i, each matrix $\mathbf{C}^i$ will represent our word embeddings for the slice i. Here below, we further explain the key phases in our model, that is, the update of the matrix for each slice, and the interpretation of the update function in our model.

Given a slice D^t, the second phase of the training process can be formalized for a single training sample $\langle w_k, \gamma(w_k) \rangle \in D^i$ as the following optimization problem:

$$\max_{\mathbf{C}^i} \log P(w_k|\gamma(w_k)) = \sigma(\mathbf{u}_k \cdot \mathbf{c}^i_{\gamma(w_k)}) \tag{1}$$

where $\gamma(w_k) = (w_{j_1}, \cdots, w_{j_M})$ represents the M words in the context of w_k which appear in D^i ($\frac{M}{2}$ is the size of the context window), $\mathbf{u}_k \in \mathbf{U}$ is the target embedding of the word w_k (the one that is fixed and that we are not going to update), and

$$\mathbf{c}^i_{\gamma(w_k)} = \frac{1}{M}(\mathbf{c}^i_{j_1} + \cdots + \mathbf{c}^i_{j_M})^T \tag{2}$$

is the mean of the context embeddings $\mathbf{c}^i_{j_m}$ of the contextual words w_{j_m}. The softmax function σ is calculated using Negative Sampling [93]. Please note that $\mathbf{C}^i$ is the only weight matrix to be optimized in this phase (**U** is the compass embedding that we do not update during training), which is the main difference from the original CBOW model. The training process maximizes the probability that given the context of a word w_k in a specific slice i, we can predict that word using the target matrix **U**. Intuitively, it moves the context embedding $\mathbf{c}^i_{j_m}$ closer to the target embeddings $\mathbf{u}_k$ of the words that usually have the word w_{j_m} in their contexts in corpus i. We extract and use the embeddings of the now trained context matrix: they will be already aligned, thanks to the shared target embeddings used as a compass during the independent training. Since the embeddings are aligned we should have slices that look like the ones depicted in Figure 10.

Properties

The method we propose can also be viewed as a method that is based on the same intuition given in [57] using neural networks and word2vec. It can also be viewed as a simplification of the models defined by [6, 108]. Despite the simplification and the fact that we propose a method that generalizes the CBOW model, in the experimental section we show that CADE outperforms or equals state-of-the-art algorithms that are more sophisticated on different experimental settings.

Note that our model is also efficient: it has the same complexity of running CBOW one time over the entire corpus D, plus the task of computing n times the CBOW algorithm over all the slices. Note that we also can take advantage of the fact that the training of the slices are independent to parallelize this last part of the process.

We believe that one key property of this model is that the simplicity of the training method can foster the application of these comparative embeddings in studies conducted in related research fields to study biases in language models [27].

We observe that, differently from those approaches that enforce similarity between consecutive word embeddings (for example, in the temporal domain [108]), CADE does not apply any time-specific assumption. In the next sections, we show that this feature does not affect the quality of the temporal word embeddings generated using CADE and that since this model is not constrained by a temporal assumption, it can be used with corpora that are sliced using different criteria (the last section of this Chapter reports the experiments that are related to the generalization of this method).

One last thing that is left to define in our comparative framework with CADE is how to make comparisons between different embeddings $\mathbf{C}^i$. Since our spaces are aligned, we can just move one word vector of a slice to another slice to find the corresponding element in by looking at the neighborhood in the space (by considering cosine similarity).

Definition 9 (Correspondence as Transposition) *Given two slices D^i and D^j, the ϕ function of a word w_k of D^i retrieves the 1-nearest neighbor of the vector of the word w amongst all the vectors of $\mathbf{C}^j$.*

This definition can be easily generalized to the Top-K neighbors. As already stated above, this function can be interpreted as an analogy between spaces. Indeed, we will experiment on a temporal analogical reasoning task in the next section.

Note that in the next Chapters, we will use the operation $D^1 \rightarrow D^2$ of a word to express the correspondence as transposition. For example, when considering text from different newspapers, e.g., the Guardian and the New York Times, we will use the operation GUA $\rightarrow$ NYT of the vector of the word "flat" and show the neighborhood of the vector of the word "flat_GUA" in the NYT space. A good comparative model should find the word "apartment" in the first positions of the neighborhood of the NYT space (as suggested by Figure 10).

In the next sections, we provide experiments to answer our research questions. Our deeper experimental evaluation is based on temporal word embeddings; we mainly focus on this category for two reasons: (i) temporal word embeddings are a field that is getting much interest lately [60, 73, 108, 145] and thus, the experimental evaluation and datasets are now standardized; moreover (ii), we believe that temporal data is a good prototype to model language differences: close-in-time slices tend to be more similar than far-in-time slices and thus, dealing with temporal data offers the possibility of evaluating more general characteristics of language change.

The rest of the chapter is divided into three parts, in the first we evaluate the performance of the model against temporal word embedding baselines, in the second we show some details related to the robustness of the model and in the third part we show an experiment on the generalization of CADE that shows that the model can treat cases that are more general than those based on temporal data.

4.3 PERFORMANCE OF CADE

We compare CADE with the state-of-the-art models that have shown better performance according to the literature. We use the two main methodologies proposed to evaluate temporal embeddings so far: temporal analogical reasoning [145] and held-out tests [108]. In the temporal word embedding field, a temporal word analogy is an analogy of the type "obama" : 2010 :: "bush" : 2005. Note that in CADE we can solve this analogy using the correspondence function defined above.

We recall that the results that we describe in this section have been previously published in [37]. In this section we will treat CADE has a member of the family of the Temporal Word Embedding Model (TWEM). However, we remark that while we can use CADE on temporal data, we will show that the model is more general.

Experiments on Temporal Analogies

To evaluate both Q1 and Q2 of this chapter, we focus on two different datasets and testset of the state of the art. We focused on two datasets, one that contains a large amount of text and the other one that contains a lower amount of text. This is useful to test the effects of different quantity of input text on the models' performances.

Datasets and Methodology

The *small dataset* [145] is freely accessible online[1]. We will refer to this dataset as NAC-S. The *big dataset* is the New York Times Annotated

1 https://sites.google.com/site/zijunyaorutgers/publications

Data	Words	Span	Corpus
NAC-S	50M	1990-2016	27
NAC-L	668M	1987-2007	21
MLPC	6.5M	2007-2015	9

Test	Analogies	Span	Categories
T1	11,028	1990-2016	25
T2	4,200	1987-2007	10

Table 5: Details of NAC-S, NAC-L, MLPC, T1 and T2.

Corpus[2] [112] employed by Szymanski, Zhang et al. to test their models. We will refer to this dataset as NAC-L. Both datasets contain a collection of corpora that is divided by year, the text in each year is a slice.

To these two datasets, two test set come associated: T1 introduced by Yao et al. and T2 introduced by Szymanski. They are both composed of temporal word analogies based on publicly recorded knowledge, partitioned in categories (e.g., *President of the USA*, *Super Bowl Champions*). The main characteristics of datasets and test sets are summarized in Table 5.

To replicate the experimental settings of Yao et al. and Szymanski we follow the following procedure: To test the models trained on NAC-S we used the T1, while to test the models trained on NAC-L we used the T2. We quantitative evaluate the performance of a model in the task of solving *temporal word analogies* (TWAs) [120]: a temporal word analogies is given in the form $w_1 : t_1 = x : t_2$ and the task is to find the word w_2 that is the most semantically similar word in t_2 to the *input word* w_1 in t_1, where t_1 and t_2 are temporal intervals. Because semantically similar words result in distributional similar ones, it follows that two words involved in a TWA will occupy a similar position in the vector space at different points in time. Then, solving the TWA $w_1 : t_1 = x : t_2$ consists in finding the nearest vector at time t_2 to the input vector of the word w_1 at time t_1. For example, we expect that the vector representation of the word *clinton* in 1997 will be similar to the vector representation of the word *reagan* in 1987. This operation is equivalent to the correspondence function we have introduced above and we will show that we can use the same operation in the experiment in which we generalize CADE.

We also report a more detailed analysis on the analogies tested in this context by considering how the models perform with respect to the span over time that an analogy covers (i.e., from 1987 to 1997 as in the example above): given an analogy $w_1 : t_1 = x : t_2$, we define *time depth* δ_t as the distance between the temporal intervals involved in the analogy: $\delta_t = |t_1 - t_2|$. In theory, the bigger δ is the more difficult

2 https://catalog.ldc.upenn.edu/ldc2008t19

the analogy is since it has to compare slices that are more distant one with the other.

Moreover, analogies can be divided into two subsets that will be useful for an in-depth analysis of the models: the set Static of *static analogies*, which involve an analogy pair of the same words (e.g., *obama : 2009 = obama : 2010*), and the set Dynamic of *dynamic analogies*, that are not static. We will refer to the complete set of analogies as All. Given a model and a set of TWAs, the evaluation on the given answers is done with the use of two standard metrics, the Mean Reciprocal Rank (MRR) and the Mean Precision at K (MP@K), we will show the results for all the subsets, namely Static Dynamic and All. Metrics computed on All give the general value of the performance of each model.

Baselines

We tested different models to compare the results of CADE with the ones provided by the state-of-the-art. We generally use the following configuration for CADE: CBOW and Negative Sampling using an extended version of the *gensim* library; in the section related to held-out experiments we will use a slightly different setting.

We compare CADE with the following models that are part of the pairwise and joint alignment families:

- *LinearTrans-Word2vec* (*TW2V*) [120].
- *OrthoTrans-Word2vec* (*OW2V*) [60].
- *Dynamic-Word2vec* (*DW2V*) [145]. The original model was not available for replication at the time of the experiments. However, they provide the dataset and the test set of their evaluation settings (the same employed in our experiments) and published their results using the same metrics. Thus, the model can be compared to our using the same parameters.
- *Geo-Word2vec* (*GW2V*) [6]. We use the implementation provided by the authors.
- *Static-Word2vec* (*SW2V*): a baseline adopted by Yao et al. and Szymanski. The embeddings are learned over all the collection, ignoring the temporal slicing. Note that this model solves all the analogies as if they were *static*, that is, the model always replies to an analogies using the same word given in input. This model will have an accuracy of 1 on *static* analogies.

Note that in this task we also tested the model introduced by Rudolph and Blei and we obtained results close to the baseline SW2V, that is a almost static model; these results have been confirmed by Barranco, Santos, and Hossain in the literature; we thus do not report the results for this model on the analogy task.

Experiments on NAC-S

The first setting involves all the models, trained on NAC-S and tested over the analogies in T1. The hyper-parameters reflect those of Yao et al.: small embeddings of dimension 50, a window of length 5, 5 negative samples and a small vocabulary of 21k words with at least 200 occurrences over the entire corpus. Table 6 summarizes the results.

We can see that CADE outperforms the other state-of-the-art models with respect to all the metrics used for the evaluation. In particular, it performs better than DW2V model, giving 7% more correct answers in the analogical task. We can also confirm the superiority of DW2V with respect to the pairwise alignment methods, as the author suggested in his work Yao et al. Unfortunately, due to the lack of the answers set and the embedding used by DW2V, we cannot know how well it performs over static and dynamic analogies separately. TW2V and OW2V scored below the static baseline (as also reported by Yao et al.), particularly on analogies with small time depth (Figure 12). In this setting, the pairwise alignment approach leads to huge disadvantages due to data sparsity: the partitioning of the corpus produces tiny slices (around 3.5k news articles for each slice) that are not sufficient to train a neural network properly; the poor quality of the embeddings affects the subsequent pairwise alignments. As expected, SW2V's accuracy on analogies drops sharply as time depth increases since the farther in time we go, the less static analogies exist (Figure 12). On the contrary, CADE, TW2V and OW2V maintain almost steady performances over different time depths. GW2V does not answer correctly almost any dynamic analogy. We conclude that GW2V alignment is not capable of capturing the semantic dynamism of words across time for the temporal analogy task. For this reason, we do not employ it in our second setting. In general, our results on this dataset prove that CADE provides good representations, it is also interesting that our experiment confirmed some of the results previously provided by Yao et al. Our comparative model can effectively solve the temporal analogies with better results than other models.

We show some properties related to the temporal analogies, giving hints on which are the most difficult tasks for a comparative model. The comparison of the models' performances across the 25 categories of analogies contained in T1 reveals new details about the models: TW2V and OW2V's correct answers cover mainly 4 categories, like *President of the USA*; CADE scores better over all the categories. Some categories are more difficult than others: even CADE scores nearly 0% in many categories, like *Oscar Best Actor and Actress* and *Prime Minister of India*. This discrepancy may be due to various reasons. First of all, some categories of words are more frequent than others in the corpus, so their embeddings are better trained. For example, thw word *obama* occurs 20,088 times in NAC-S, whereas *dicaprio* only 260. As noted by Yao et al., in the case of some categories of words, like

Model	Set	MRR	MP1	MP3	MP5	MP10
SW2V	Static	**1**	**1**	**1**	**1**	**1**
	Dynamic	0.148	0.000	0.263	0.351	0.437
	All	0.375	0.266	0.459	0.524	0.587
TW2V	Static	0.245	0.193	0.280	0.313	0.366
	Dynamic	0.106	0.069	0.123	0.156	0.205
	All	0.143	0.102	0.165	0.198	0.248
OW2V	Static	0.265	0.202	0.299	0.348	0.415
	Dynamic	0.087	0.058	0.099	0.124	0.160
	All	0.135	0.096	0.153	0.183	0.228
DW2V	Static	–	–	–	–	–
	Dynamic	–	–	–	–	–
	All	0.422	0.331	0.485	0.549	0.619
GW2V	Static	0.857	0.819	0.888	0.909	0.931
	Dynamic	0.071	0.005	0.092	0.159	0.225
	All	0.280	0.222	0.305	0.359	0.435
CADE	Static	0.720	0.668	0.763	0.787	0.813
	Dynamic	**0.394**	**0.308**	**0.451**	**0.508**	**0.571**
	All	**0.481**	**0.404**	**0.534**	**0.582**	**0.636**

Table 6: MRR and MP for the subsets of static and dynamic analogies of T1. We use MPK in place of MP@K. DW2V results are taken from the original paper [145].

presidents and mayors, we are heavily assisted by the fact that they commonly appear in the context of a title (e.g. *President Obama*, *Mayor de Blasio*). For example in CADE, *obama* during its presidency is always the nearest context embedding to the word *president*. Lastly, as noted by Szymanski, some roles involved in the analogies only influence a small part of an entity's overall news coverage. We show that this is reflected in the vector space: as we can see in Figure 14, presidents' embeddings extracted from CADE almost cross each other during their presidency, because they share a lot of contexts; on the other hand, football teams' embeddings remain distant. One other interesting aspect is that word frequency seems to have a valuable impact on the ability of the comparative framework of finding shifted words (we will give more hints about this in 7, but Figure 15 already shows that there is a positive correlation between the ability of solving an analogy and the frequency of occurrence of words that compose that analogy).

Experiments on NAC-L

This setting involves three baseline models, SW2V, TW2V, OW2V, and CADE. The models are trained on NAC-L and tested over T2. The set-

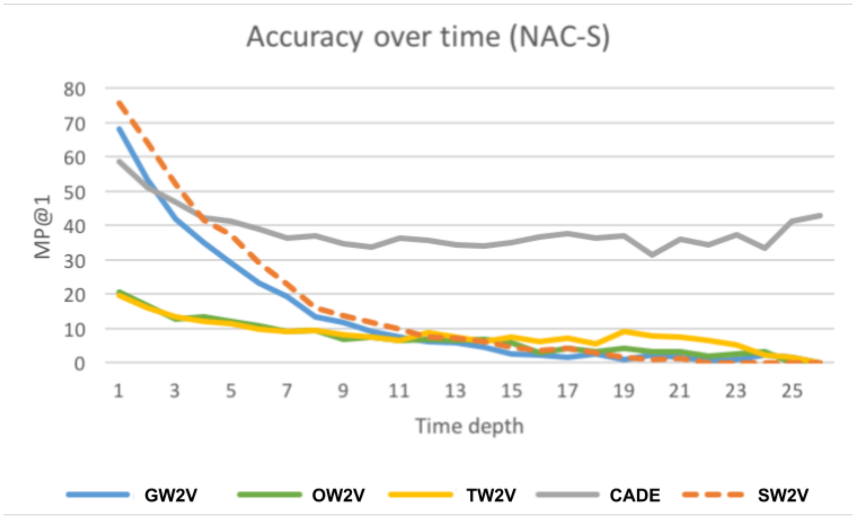


Figure 12: Accuracy (MP@1) as function of time depth δ_t in T1. Given an analogy $w_1 : w_2 = t_1 : t_2$, the time depth is plotted as $\delta_t = |t_1 - t_2|$.

tings' parameters are similar to those of Szymanski that introduced the task. Embeddings of size 100, a window size of 5, 5 negative samples and a very large vocabulary of almost 200k words with at least 5 occurrences over the entire corpus. We show the results of the temporal analogical reasoning task on this dataset in Table 7.

Model	Set	MRR	MP1	MP3	MP5	MP10
SW2V	Static	**1**	**1**	**1**	**1**	**1**
	Dynamic	0.102	0.000	0.149	0.259	0.326
	All	0.283	0.201	0.321	0.408	0.462
TW2V	Static	0.842	0.805	0.869	0.890	0.915
	Dynamic	0.343	0.287	0.377	0.414	0.467
	All	0.444	0.391	0.476	0.510	0.558
OW2V	Static	0.857	0.824	0.876	0.903	0.926
	Dynamic	0.346	**0.290**	0.379	0.420	0.462
	All	0.449	0.398	0.480	0.518	0.556
CADE	Static	0.948	0.936	0.959	0.961	0.967
	Dynamic	**0.367**	0.287	**0.423**	**0.471**	**0.526**
	All	**0.484**	**0.418**	**0.531**	**0.570**	**0.615**

Table 7: MRR and MP for the subsets of static and dynamic analogies of T2. We use MPK in place of MP@K.

CADE still outperforms all the other models with respect to all the metrics, but its advantage seems to be less than in the previous setting. We can say that CADE assigns lower ranks to correct answer words comparing to TW2V and OW2V; however, it gives first-position results as frequently as the competing models. Table 7 shows that the

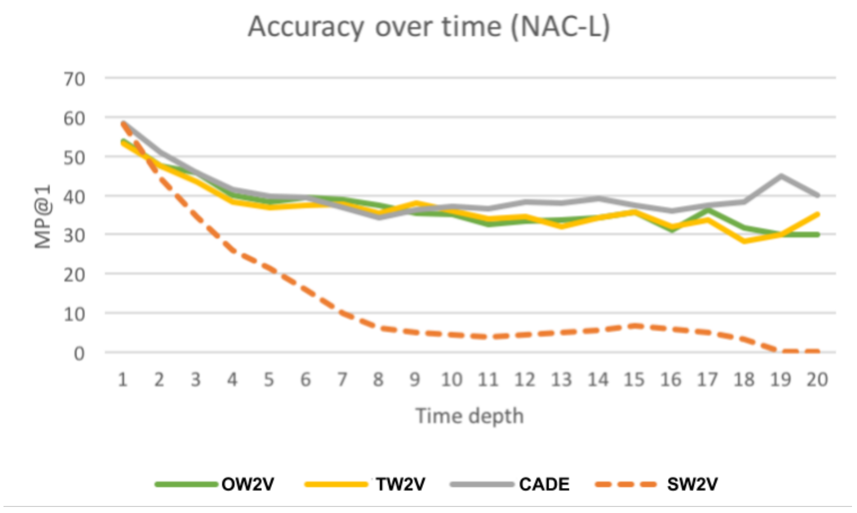


Figure 13: Accuracy (MP@1) as function of time depth δ_t in T2. Given an analogy $w_1 : w_2 = t_1 : t_2$, the time depth is plotted as $\delta_t = |t_1 - t_2|$.

advantage of CADE is limited to the static analogies. TW2V and OW2V score much better results than in the previous setting. This is due to the increased size of the input dataset which allows the training process to work well on individual slices of the collection.

In Figure 13 we can see how the three temporal models behave similarly with respect to the time depth of the analogies.

The comparison of the models' performances across the 10 categories of analogies contained in T2 reveals more differences between them. The results in terms of MP@1 are summarized in Table 8. TW2V and OW2V significantly outperform CADE in two categories: *President of the USA* and *Super Bowl Champions*. In both cases, this is due to the major accuracy on dynamic analogies; most of the time, CADE is wrong because it gives static answers to dynamic analogies. CADE significantly outperforms the other models in two categories: *WTA Top-ranked Player* and *Prime Minister of UK*. However in this case, CADE outperforms them both on dynamic and static analogies. Note that there is also a strong relationship between the frequency of occurrence of the elements of the analogies in text and the actual accuracy of CADE. In Figure 15 we show a scatter plot where we compare the mean (log) frequency of appearance of the words of a certain analogical category in T2 and CADE accuracy.

Experiments on Held-Out Data

In this section, we show the performance of CADE on a held-out test task, in which we try to predict the *slice* form which a held-out text comes from. We perform this test in two different ways. We tried to replicate the likelihood based experiments in Rudolph and Blei and

Category	SW2V	TW2V	OW2V	CADE
President of the USA	0.4000	**0.9905**	**0.9905**	0.8833
Secret. of State (USA)	0.1190	0.3000	**0.3619**	0.3405
Mayor of NYC	0.2476	**0.9643**	0.9524	0.9405
Gover. of New York	0.4476	0.9333	0.9381	**0.9786**
Super Bowl Champ.	0.0571	0.2024	**0.2524**	0.1452
NFL MVP	**0.0190**	0.0143	0.0143	**0.0190**
Oscar Best Actress	0.0095	**0.0119**	0.0071	**0.0119**
WTA Top Player	0.1619	0.2071	0.1548	**0.2857**
Goldman Sachs CEO	**0.1762**	0.0143	0.0238	0.1190
UK Prime Minister	0.3762	0.2762	0.2857	**0.4595**

Table 8: Accuracy (MP@1) for the subsets of the analogy categories in T2. The best scores are highlighted.

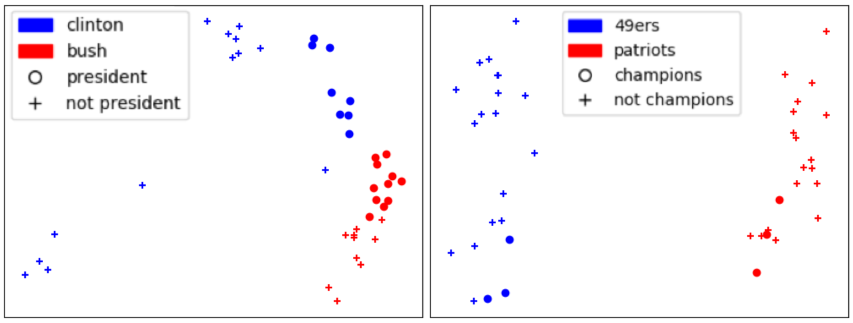


Figure 14: 2-dimensional PCA projection of the temporal embeddings of pairs of words from *clinton*, *bush* and *49ers*, *patriots*. The dot points highlight the temporal embeddings during their presidency or their winning years.

to further give confirmation about the performance of our model we also test the posterior probabilities using the framework described in Taddy. Given a model, Rudolph and Blei assign a Bernoulli probability to the observed words in each held-out position: this metric is straightforward because it corresponds to the probability that appears in Equation 1. However, at the implementation level, this metric is highly affected by the magnitude of the vectors because it is based on the dot product of the vectors $\mathbf{u}_k$ and $\mathbf{c}_{\gamma(w_k)}$. In particular, Rudolph and Blei applied L2 regularization on the embeddings, which prioritize vectors with small magnitude.

This makes the comparison between models trained with different methods more difficult: regularization over the dot product can bias the comparison of the results. Furthermore, we claim that held-out likelihood is not enough to evaluate the quality of a Temporal Word Embedding Model (TWEM): a good temporal model should be able to extract features from each temporal slice that are discriminative and to improve the likelihood based on those features. To quantify this

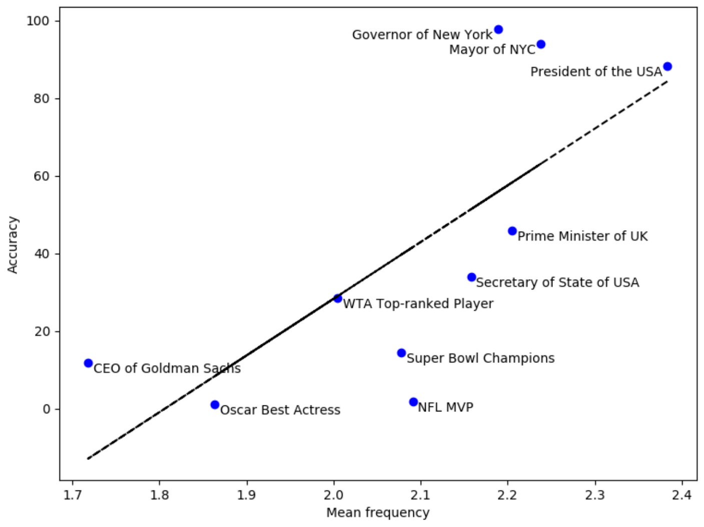


Figure 15: Relation between mean frequency of occurrence of the words of specific analogical categories and CADE accuracy in solving those analogies.

specific quality, we propose to adapt the task of document classification for the evaluation of TWEM. We take advantage of the simple theoretical background and the easy implementation of the work of Taddy. We show that luckily this new metric is not affected by the different magnitude of the compared vectors.

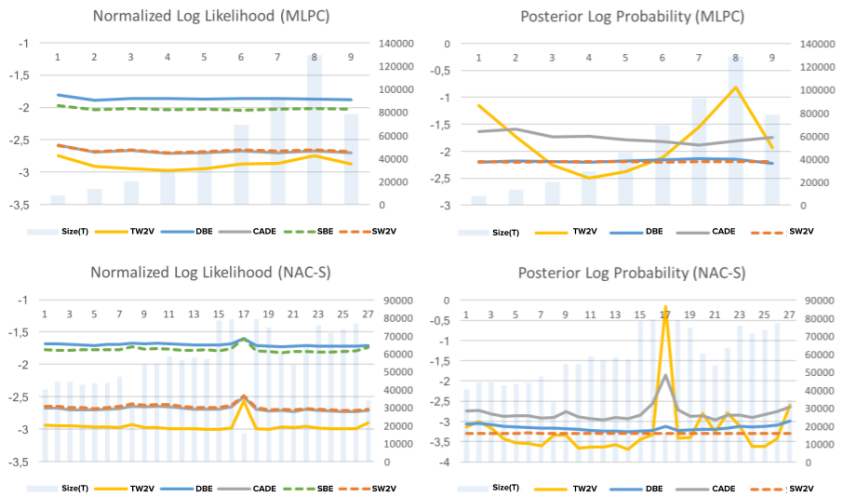


Figure 16: $\mathcal{L}^t_{\mathcal{V}}$ and $\mathcal{P}^t_{\mathcal{V}}$ for each test slice $\mathcal{T}^t$ and model $\mathcal{V}$. Blue bars represent the number of words in each slice.

Dataset We use two datasets to evaluate the models on this task, details of these datasets are summarized in Table 5. The Machine Learning Papers Corpus (MLPC) contains the full text from all the machine learning papers published on the ArXiv between 9 years, from April 2007 to June 2015. The size of each slice is very small (less than $130,000$ words after pre-processing) and it increases over the years. Rudolph and Blei made MLPC available online [108]: the text we obtain is already pre-processed, sub-sampled ($|V| = 5,000$) and already split into training, validation and testing (80%, 10%, 10%) to ease replication of the experiments. The data is shared in a computer-readable format without sentence boundaries: we convert it to plain text and we arbitrarily split it into 20-word sentences that are suited to cover our training procedure. To make another comparison we also used the NAC-S dataset, which was described in previous sections and used to solve temporal word analogies. Consider that, compared to MLPC, NAC-S has $\times 3$ more slices and it has approximately $60,000$ words per slice, with a small exception due to the slice of the year 2006. To prepare this dataset for evaluation we use the same pre-processing script provided by Rudolph and Blei and divided the data training and testing ($|V| = 21,000$).

We introduce a new notation that will be helpful to understand this experiment, we will refer to $\mathcal{V} = \{\mathcal{V}_{t \in 1 \cdots T}\} = \{\langle \mathbf{C}^{t \in 1 \cdots T}, \mathbf{U}^{t \in 1 \cdots T} \rangle\}$ as to the TWEM taken in consideration and we will use $\mathcal{T}^t$ to identify a temporal slice.

Methodology A We measure the held-out likelihood following a methodology similar to the one proposed by Rudolph and Blei. For this experimental evaluation we introduce the symbol $\mathcal{V}$ to identify a specific temporal word embedding model. Given a TWEM $\mathcal{V} = \{\mathcal{V}_{t \in 1 \cdots T}\} = \{\langle \mathbf{C}^{t \in 1 \cdots T}, \mathbf{U}^{t \in 1 \cdots T} \rangle\}$, we calculate the log-likelihood for the temporal testing slice $\mathcal{T}^t = \langle w_1, \cdots, w_N \rangle$ as:

$$\log P_{\mathcal{V}_t}(\mathcal{T}^t) = \sum_{n=1}^{N} \log P_{\mathcal{V}_t}(w_n | \gamma(w_n)) \tag{3}$$

where the probability $\log P_{\mathcal{V}_t}(w_n | \gamma(w_n))$ is calculated based on Equation 1 using Negative Sampling and the vectors of $\mathbf{C}^t$ and $\mathbf{U}^t$. As Rudolph and Blei, we equally balance the contribution of the positive and negative samples. For each model $\mathcal{V}$, we report the value of the normalized log likelihood $\mathcal{L}^t$:

$$\mathcal{L}^t_{\mathcal{V}} = \frac{1}{N} \log P_{\mathcal{V}}(\mathcal{T}^t) \tag{4}$$

and its arithmetic mean $\mathcal{L}_{\mathcal{V}}$ over all the slices.

Methodology B We adapt the methodology of Taddy to the evaluation of TWEM. We calculate the posterior probability of assessing a temporal testing slice $\mathcal{T}^t$ to the correct temporal class label t. In our setting, this corresponds to the probability that a model $\mathcal{V}$ predicts the year of the t-th slice given a held-out text that come from the same slice. We apply Bayes rules to calculate this probability:

$$P_{\mathcal{V}_t}(t | \mathcal{T}^t) = \frac{P_{\mathcal{V}_t}(\mathcal{T}^t) P(t)}{\sum_{k=1}^{T} P_{\mathcal{V}_k}(\mathcal{T}^t) P(k)} \tag{5}$$

A good temporal model $\mathcal{V} = \{\mathcal{V}_{t \in 1 \ldots T}\}$ will assign a high likelihood to the slice $\mathcal{T}^t$ using the vectors of $\mathcal{V}_t$ and a relatively low likelihood using the vectors of $\mathcal{V}_{k \neq t}$. We assume that the prior probability on the class label t is the same for each class, $P(t) = 1/T$. For implementation reason, we redefine the posterior likelihood as:

$$\mathcal{P}^t_{\mathcal{V}} = P_{\mathcal{V}_t}(t|\mathcal{T}^t) = \frac{1}{S} \sum_{s=1}^{S} \frac{P_{\mathcal{V}_t}(z^t_s)P(t)}{\sum_{k=1}^{T} P_{\mathcal{V}_k}(z^t_s)P(k)} \tag{6}$$

where z_s is the s-th sentence in $\mathcal{T}^t$ and $P_{\mathcal{V}_t}(z_s)$ is calculated based on Equation 3. Please note that this metric is not affected by the magnitude of the vectors because is based on a ratio of probabilities. For each model $\mathcal{V}$, we report the value of the posterior log probability $\mathcal{P}^t_{\mathcal{V}}$ and its arithmetic mean $\mathcal{P}_{\mathcal{V}}$ over all the slices.

Algorithms We test five temporal embedding models for this setting:

- Our comparative framework CADE
- TW2V (a baseline used in previous experiments)
- SW2V (a baseline used in previous experiments)
- *Dynamic Bernoulli Embeddings* (DBE) [108]
- *Static Bernoulli Embeddings* (SBE) [109].

Note that TW2V is equivalent to OW2V in this setting because we do not need to align vectors from different slices. DBE is the temporal extension of SBE, a probabilistic framework based on CBOW: it enforces similarity between consecutive word embeddings using a prior in the loss function, and specularly to CADE, it uses a unique representation of context embeddings for each word. We trained all the models on the temporal training slices D^t using the CBOW architecture, a shared vocabulary and the same parameters, which are similar to Rudolph and Blei: learning rate $\eta = 0.0025$, window of size 1, embeddings of size 50 and 10 iterations (5 static and 5 dynamic for CADE, 1 static and 9 dynamic for DBE as suggested by Rudolph and Blei). Following Rudolph and Blei, before the second phase of the training process of CADE, we initialize the temporal models with both the weight matrices **C** and **U** of the compass model: we experimentally noted that this operation improves held-out performances but it negatively affects the analogy tests. We limit our study to small datasets and small embeddings due to the computational cost: DBE takes almost 6 hours to train on NAC-S on a 16-core CPU setting. DBE and SBE are implemented by the authors using *tensorflow*, while all the other models are implemented in *gensim*: to evaluate them, we convert them to *gensim* models, extracting the matrices $\mathbf{U}^t$ and **C**.

Results

Table 9 shows the mean results for each model by considering the two metrics. In both settings, CADE obtains a likelihood almost equal to SW2V but a much better posterior probability. This is remarkable considering that CADE optimizes the scoring function only on one weight matrix $\mathbf{C}^t$, keeping the matrix $\mathbf{U}^t$ frozen. With respect to TW2V, CADE has a better likelihood and its posterior probability is more stable across slices (Figure 16). The likelihood scores of DBE and SBE are highly influenced by the different magnitude of their vectors: we can quantify the contribution of the applied L2 regularization comparing the two static baseline SBE and SW2V. Differently from CADE, DBE slightly improves the likelihood with respect to its baseline. However, regarding the posterior probability, CADE outperforms DBE. Our experiments seem to suggest the existence of an inverse correlation between the capability of generalization and the capability of extracting discriminative features from small diachronic datasets. Finally, experimental results show that CADE captures discriminative features from temporal slices without losing generalization power.

Dataset	M	SW2V	SBE	CADE	DBE	TW2V
MLPC	$\mathcal{L}_\mathcal{V}$	-2.67	-2.02	-2.68	**-1.86**	-2.88
	$\mathcal{P}_\mathcal{V}$	-2.20	-2.20	**-1.75**	-2.18	-2.83
NAC-S	$\mathcal{L}_\mathcal{V}$	-2.66	-1.77	-2.69	**-1.70**	-2.96
	$\mathcal{P}_\mathcal{V}$	-3.30	-3.30	**-2.80**	-3.16	-3.24

Table 9: The arithmetic mean of the log likelihood $\mathcal{L}_\mathcal{V}$ and of the posterior log probability $\mathcal{P}_\mathcal{V}$ for each model $\mathcal{V}$. Based on the standard error on the validation set, all the reported results are significant.

4.4 ROBUSTNESS OF CADE

In this section, we explore some of the assumptions that stand behind the model and its validity. In general, we found out that CADE cannot be used in contexts in which there is a low shared vocabulary between the slices. For example, using two different languages as slices bring to vector spaces in which specific correspondences are not really informative. We hereby perform different experiments with different objectives:

- Aligning the same corpus twice: we experimentally evaluated what is the robustness of CADE by first testing what happens if we try to align two identical slices:
- Staticness: we introduce the concept of staticness with respect to analogy solving, which shows quantitative results on how

CADE can model semantic change in the slices (i.e., it evaluates how much a model replies with the same answers);

- Text Scrambling: we take a corpus and build a collection that contains a slice and a clone of that slice. With a varying probability p we replace words in the clone corpus with a random word. We compute how the similarity between the same words in each slice is affected by this scrambling.

In the following, we will give all the details of our experimental settings, in such a way that the reader can easily replicate this experiment by using the code provided online.

Aligning the same corpus twice

To evaluate the *compass intuition*, we tried to use CADE to align two slices that are composed of the same text. We use a document D^1 and create an exact copy of it and use these two inside CADE. The collection D thus contains the same slice twice. We end up with two vector spaces for which the same words in the respective model have an average cosine similarity of 0.998 (computed on a sample of 1000 words). Words that occur less frequently tend to have a lower similarity due to a still existing intrinsic noise in the network. Nevertheless, in general, the alignment brings the slices to be almost identical with respect to the cosine similarity.

To perform this experiment, we run the model 1000 times with a corpus of 5000 articles that was cloned to create the second corpus and averaged the results. We used 20-dimensional embeddings with a window of 5.

Note that the similarity between the vectors is generally stable if we use cosine similarity, but that the same is not true if we change the distance measure to one that also considers the length (i.e., the norm) of the vector. To regulate the effect on the euclidean distance between vectors, it is important to consider the number of training epochs. In general, more training epochs generate "longer" vectors.

Staticness

TWEMs are designed to represent the dynamics of word embeddings over time, as opposed to static models like SW2V. Here we want to evaluate how much the word embedding changes over time. If a model generates temporal embeddings that change slightly, it will be almost *static* and similar to SW2V. We would like the comparative framework defined by CADE to be general enough to find a good threshold between accuracy over static and dynamic analogies. This corresponds to be able to find tokens that have and that have not shifted between two slices.

The current literature does not offer a standardized metric to quantify how "static" the temporal word embeddings generated by a model are. We introduce a new metric that allows us to measure the tendency of a model to give the same answers as SW2V. Given a test set of analogies and the answers given by a model, the STAT is measured as $\text{STAT} = \frac{1}{N}\sum_{i=1}^{N}\mathtt{isstatic}(i)$, where the function $\mathtt{isstatic}(i)$ is equal to 1 if the answer given to the i-th analogy is static, 0 otherwise. It follows that a perfect model (we will refer to it as *Target*) would have a $\text{STAT} = 1.0$ over static analogies and a $\text{STAT} = 0.0$ over dynamic ones; its staticness will be the ratio between the number of static analogies and the size of the test set: we call it the *target staticness*.

The staticness of a model may affect its performances: if a model gives too many static answers, it will poorly perform over dynamic analogies and vice versa for dynamic answers. To verify this intuition, we apply the metric in the two previous settings with the answers given by the models CADE, TW2V and SW2V. We can apply STAT over subsets of the full test set, like static and dynamic analogies (note that over static analogies, the staticness is equal to the accuracy).

Model	Subset	MP@1	STAT
SW2V	All	0.2664	1.0000
TW2V	All	0.1019	0.0784
	Static	0.1933	0.1933
	Dynamic	0.0687	0.0367
CADE	All	0.4041	0.3024
	Static	0.6676	0.6676
	Dynamic	0.3082	0.1697

Table 10: STAT and Accuracy (MP@1) for the subsets of the analogies in T1.

Model	Subset	MP@1	STAT
SW2V	All	0.2014	1.0000
TW2V	All	0.3914	0.3057
	Static	0.8052	0.8052
	Dynamic	0.2867	0.1794
CADE	All	0.4183	0.4814
	Static	0.9362	0.9362
	Dynamic	0.2873	0.3663

Table 11: STAT and Accuracy (MP@1) for the subsets of the analogies in T2.

Table 10 compares the results obtained by the two models with NAC-S and T1. CADE goes slightly over the target value of STAT =

0.2664; in fact, it gives static answers to 17% of the dynamic analogies (i.e., it replies with the same input word, for example "obama" : 2014 :: ? : 2005, and replies with "obama" instead of the correct answer "bush") and it gives dynamic answers to over the 30% of the static analogies of T1. We can see it also in Figure 17, where it gives too many dynamic answers to analogies with a small time depth, and too many static answers to analogies with greater time depth ($\delta_k > 5$). TW2V performs badly in this test set and gives dynamic answers more than 90% of the times. The poor performance of TW2V on dynamic analogies suggests that the temporal embeddings are highly dynamic due to their low quality and the insufficient training process. Our comparative model, that is not based on temporal assumptions, seems to be effective in dealing with static and dynamic analogies.

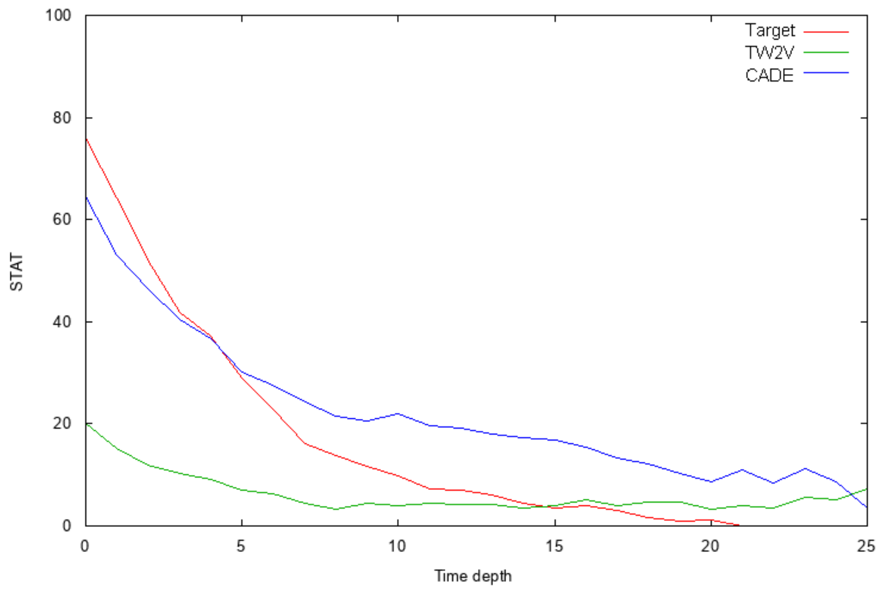


Figure 17: STAT as function of time depth δ_t in T1. Given an analogy $w_1 : w_2 = t_1 : t_2$, the time depth is plotted as $\delta_t = |t_1 - t_2| - 1$.

Table 11 reports the results relative to the big dataset settings, obtained with NAC-L and T2. TW2V returns 30% of the time the same answers as SW2V, CADE almost half of the times, while the test set contains of static analogies equal to the 20%. As a result, CADE scores better than TW2V on static analogies, but it scores the same as TW2V on dynamic analogies. This means that CADE gives less dynamic answers to dynamic analogies than TW2V but these dynamic answers are relatively more accurate. Moreover, CADE consistently gives more static answers then TW2V, independently from the time depth value (Figure 17). CADE generally represents the words too statically, i.e. the temporal embeddings of the same word are placed too close together. This leads to an advantage for CADE on static analogies but a disadvantage on dynamic ones. Moreover, we may expect that the static answer may sometimes "hide" the dynamic correct one. Indeed, in-

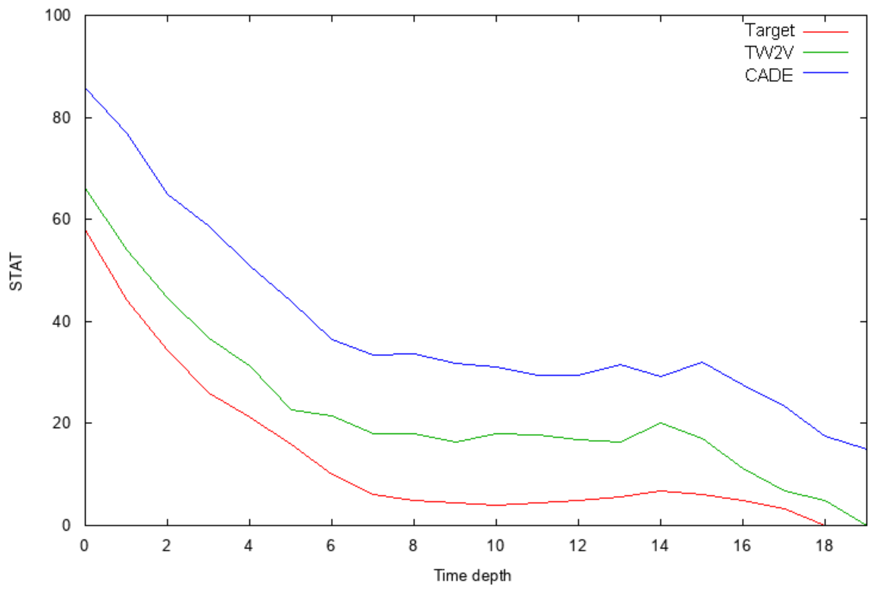


Figure 18: STAT as function of time depth δ_t in T2. Given an analogy $w_1 : w_2 = t_1 : t_2$, the time depth is plotted as $\delta_t = |t_1 - t_2| - 1$.

correct static answers to dynamic analogies represent half the errors of CADE (x2 more times than TW2V); in 17% of these cases, the correct dynamic answer is just behind the incorrect static one ($\mathrm{rank}(i) = 2$): it corresponds to a 5% drop in MP@1 on dynamic analogies. TW2V may have the opposite problem: their representation of words is too dynamic; as a consequence, they give less dynamic answers to static analogies than they should do, but this may be an advantage when dealing with dynamic analogies.

The reason behind this discrepancy of staticness between TW2V and CADE may lie in the alignment strategy: the compass based approach may move the temporal embeddings of a word closer together because they share many words in their context. For example, all the temporal embeddings of the word *bush* are close to the context embedding of the word *george*, because the two words appear nearby in all the time slices.

Text Scrambling

In this experiment we want to test which is the effect of random noise in the alignment. Starting from the input corpus (clean slice) we generate a second corpus (noisy slice) in which each word has a probability p of being replaced with a different word randomly sampled from the vocabulary. When the probability equals to 1 the two slices are completely different one with the other. We use CADE to align the two slices. At the end of the process, we sampled 1000 words from the vocabulary and we evaluate the average similarity between the two representations of each word, one in the clean and

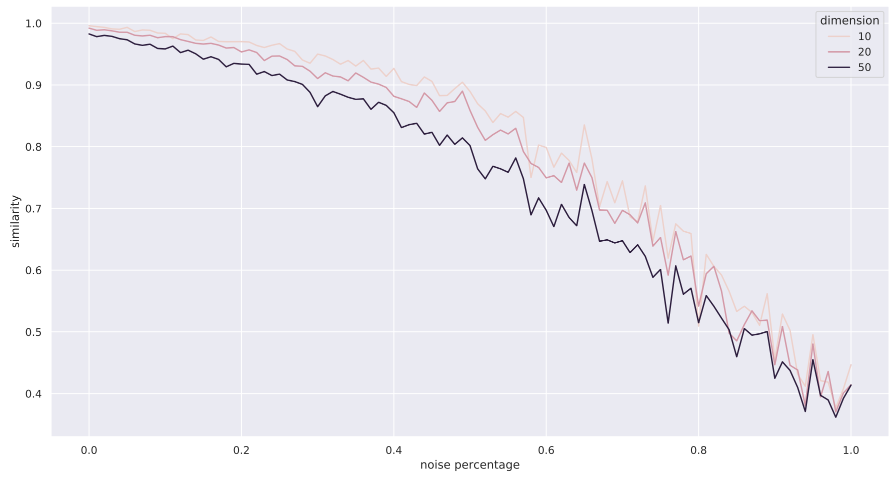


Figure 19: Similarity between same words in CADE in the clean corpus and the noisy corpus while we vary the percentage of noise

one in the noisy slice. We tested this methodology on embeddings of three different sizes: 10, 20 and 50. In Figure 19 we show the effect of this noising procedure on the words.

From the results, it seems that the model is able to sustain a noise value around 20% without loosing too much information. After 20% of noise, the similarities start to drop quickly.

4.5 GENERALIZATION OF CADE

In this section, we try to answer the research question Q3 of this chapter that is to evaluate the generalization capabilities of our corpus-based comparative framework. To show how CADE is able to generalize the alignment between word vector spaces generated from different corpora we use a novel dataset that contains text from newspapers.

Newspapers Data

In this section, we show a comparison between embeddings generated from an American newspaper, The New York Times (**NYT**), and from a British newspaper, The Guardian (**GUA**). We want to show that CADE is able to capture intrinsic language differences in the same language.

Data and Algorithms

Using the EventRegistry platform[3], we daily extracted articles from the New York Times and from The Guardian online platforms from the 9th of July 2019 to the 20th of September 2019. At the end of

3 https://eventregistry.org/

the process, we collected 14.480 articles from the New York Times, and 17.976 articles form The Guardian. We remove stop words from both the slices and bring the text to lowercase. Thus, in the context of our comparative framework, our collection D contains two slices, one with the text from the New York Times and one with the text from The Guardian.

To provide quantitative evidence for this comparison we use a list of pairs of words that show minor spelling differences in British and American English[4]: for example, British people tend to use the form "labelling" while Americans use "labeling". From these pairs of words that we extracted online, we removed words that appear with a frequency lower than 20 and 50 times and those words that were not present in our corpora. We end up with two sets of 279 (**BAW1**) and 131 (**BAW2**) pairs of words. In a certain sense, these pairs of words can be interpreted as analogies between the British English space and the American English space.

As a baseline algorithm, we use MUSE [31], a multi-lingual alignment tool proposed by Facebook at ICLR 2018. This tool can align word embeddings from multiple languages without parallel corpora to support the alignment. In our case, we will use it to try to align to different *versions* of the same language. To perform a fair comparison, we used the configuration of the algorithm suggested by the authors, and we trained the embeddings using the same procedure defined in the paper.

RESULTS We compare CADE and MUSE on the same task: given a word in the English language and moving its vector representation, we want to find its equivalent in the American language (by looking at the neighborhood). We look at the top-5 and top-10 neighbors, and we thus evaluate the HITS@5 and HITS@10 on both **BAW1** and **BAW1**. Results are visible in Tables 12 and Tables 13 and show that CADE performs better than the competitor in this task. In general, while the alignment provided by MUSE is also good, it lacks the ability to use general contextual information that is what makes CADE more effective (i.e., this is the effect of the compass). Another point is that when filtering words with a lower frequency of occurrence, both models become better with the mappings; these results confirm what was introduced in the experiment with temporal analogies: the number of occurrences is a key element to generate a good representation. Nevertheless, we underline that CADE is currently not able to align multi-language corpora that is the main area in which MUSE was proposed.

4 http://www.tysto.com/uk-us-spelling-list.html

Model	HITS@5	HITS@10
CADE	**0.60**	**0.64**
MUSE	0.40	0.56

Table 12: British-American spelling test with words with frequency higher than 20 in the corpus (**BAW1**)

Model	HITS@5	HITS@10
CADE	**0.81**	**0.86**
MUSE	0.51	0.60

Table 13: British-American spelling test with words with frequency higher than 50 in the corpus (**BAW2**)

4.6 SUMMARY OF THIS CHAPTER

We hereby give a short summary of this chapter, by answering the research questions we asked ourselves in the introduction.

- **Q4.1** which are the performance of an unsupervised method for the implicit alignment of distributional models on a paradigmatic case?
- **Q4.2** can this method be used even in the context when some sources provided less data than others?
- **Q4.3** to which extent can this method be used in different contexts?

In this Chapter, we have introduced CADE a model to align distributional representation by using a heuristic that allows us to efficiently training and aligning embeddings. We tested our model in a though evaluation against the temporal word embeddings method, a paradigmatic case, showing that our model is stable and provides an effective and useful alignment (**Q4.1**). What makes our algorithm different is the ability to be more stable and of not overweighting the shift of the words (that occurs for rare words and not for all the words). We run experiments on datasets of different sizes, giving evidence that this method, thanks to the use of the *compass* can generate good representations even in contexts in which there are fewer data available (**Q4.2**). Eventually, we have shown that the model is not constricted by any assumption as other models in the temporal domains, and we proved its generalization capabilities on a task in which we mapped British English and American English (**Q4.3**).

5 DISTRIBUTIONAL KNOWLEDGE GRAPH EMBEDDINGS WITH ENTITY LINKING

In this Chapter, we focus on the description of a method to generate distributional representations of entities and types of a KG from various sources. As outlined in Chapter 2 we want a model that can be applied to generic textual documents. Thus, we introduce Typed Entity Embeddings (TEE) a distributional knowledge graph embedding method for entity and types that relies on an entity linking system. We show that an interesting property of this model is that is able to solve analogical reasoning tasks with a good accuracy. Finally, we show some properties of novel distributional type representations that we described in [19].

The distributional knowledge graph embedding approach that we describe in this chapter was introduced in this published paper:

- Bianchi, F., Palmonari, M., and Nozza, D. (2018, October). **Towards Encoding Time in Text-Based Entity Embeddings**. In ISWC.

This work describes our distributional method integrated into an approach to computing the similarity of entities by considering time as a factor. In this chapter we explain the model in detail and show how to use it to solve analogies. One big limit of our approach that it will be easy to notice in the next sections is that it lacks the ability of representing KG's relationships.

5.1 DISTRIBUTIONAL SEMANTICS FOR ENTITIES AND TYPES

Distributional semantics has been applied with success to corpora that contains words [93]; in this chapter, we want to study a method to generate distributional representations of entities and types of a knowledge graph.

As outlined in the related work chapter, there are some works in literature that explore embeddings of entities from text; these works often start from already annotated/linked corpora (e.g, Wikipedia).

We show that combining entity linking and embedding algorithms it is easy to generate entity representations even from non annotated corpora.

5.2 DEFINITION

Consider the following sentence **S1**: "Paris was built close to the Seine". If we replace words with entities in the set E of the DBpedia KG (see Definition 1), this sentence becomes "dbr:Paris was built close to the dbr:Seine" ($\mathbf{S1_e}$); Note that we get the two following important properties when we switch to entities [16]:

- **DE1.** Entities are non ambiguous;
- **DE2.** Entities are logical constants of a language.

DE1 brings a shift from the standard word-based approach. A word has different meaning and its representation in the embedding contains all the information about all the occurrences in text. While the word "Paris" represents the meaning generated from the co-occurring of both the city in France and the one in Texas (and many other "Paris") the entity "dbr:Paris" represents only the city in France. DE2 implies that we can use these entities in reasoning tasks [20]. A simple definition of distributional semantics of entities is the following. These two facts regarding entities will be useful in the next chapter (Chapter 6) where we will implement a reasoning system over the entities.

Definition 10 (Entity Distributional Semantics) *The similarity between two entities is a function of the contexts in which these two entities appear.*

Note that the definition is simple and is close to the standard distributional semantics for words [40, 46, 62], but the key differences are defined by DE1 and DE2.

Entities are not the only thing that can be embedded: we can also generate embeddings of types, if we consider most specific types of the entities and we replace them in $\mathbf{S1_e}$ we get "dbo:City was built close to the dbo:NaturalPlace". Note, that it makes sense (under a point of view of meaning) to replace the types in $\mathbf{S1_e}$ with types in higher part of the hierarchy as in "the dbo:Animal was sleeping by the Place" ($\mathbf{S1_e^o}$). The similarity of ontological types would in this case take in consideration the fact that it exists a hierarchy behind the ontology. The definition of a type-based distributional semantics is as follows:

Definition 11 (Type Distributional Semantics) *The similarity between two types is a function of the contexts in which these two types appear.*

Distributional semantics of entities and types should inherit the properties that have been found by cognitive scientists in representations generated with word-based distributional semantics methods [27]. This is a key aspect that distinguish our methods from structured-based knowledge graph embeddings and that will allow us to do comparison between entities in Chapter 7.

In the following section we will see how to build representations of entities and types starting from text.

5.3 Typed Entity Embeddings (TEE)

In this work we propose a novel type of embeddings from text that we call Typed Entity Embeddings (TEE). We will first give an intuitive description of the process we follow to generate these representations, and we will give the details of the main component es in the next sections

These embedded representations of entities are *typed* because we include type information in the vector representation. We generate a vector representation for the entities, a vector representation for KG's types and then we concatenate the vector of each entity with the vector of its own most specific type.

The general process to generate Typed Entity Embeddings is depicted in Figure 20. The two main components of this process are a KG and an annotator that links words in text to a KG.

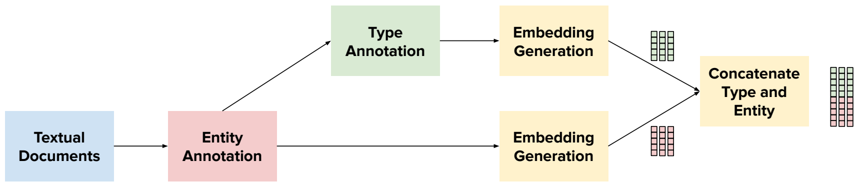


Figure 20: High level workflow to generate TEEs: we generate a vector representation for the entities, a vector representation for KG's types and then we concatenate the vector of each entity with the vector of its own most specific type.

The idea that drives Type Entity Embeddings is the following: we want to generate a representation of entities that also includes the information that come from on the ontology. This information should affect the similarity of the entities by giving more weight to the typing part: in the novel space we cities should be closer one with each other 21.

5.3.1 *Entity Embeddings from Text*

Despite the intrinsic limitations in the accuracy of Named Entity Linking techniques, we show that it is possible to rely on these techniques to first annotate text and then generate entity embeddings, instead of using the scarcer manually edited links that present in the text. Note that links are not always present in text and thus using an annotators allow us to consider text that comes from different sources. Named Entity Linking (NEL) and Named Entity Recognition (NER) are techniques that are now often offered as off-the-shelf services and have

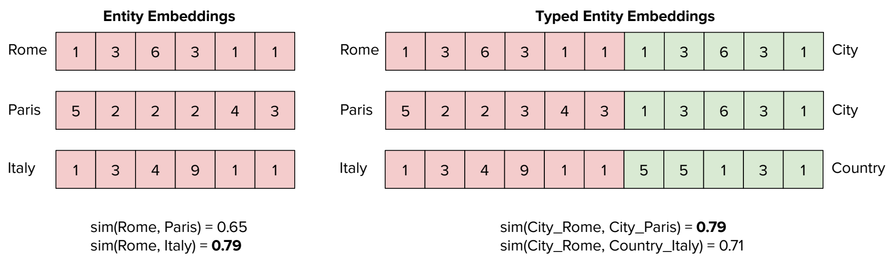

Entity Embeddings
Rome 1 3 6 3 1 1
Paris 5 2 2 2 4 3
Italy 1 3 4 9 1 1
sim(Rome, Paris) = 0.65
sim(Rome, Italy) = **0.79**
Typed Entity Embeddings
Rome 1 3 6 3 1 1 1 3 6 3 1 City
Paris 5 2 2 3 4 3 1 3 6 3 1 City
Italy 1 3 4 9 1 1 5 5 1 3 1 Country
sim(City_Rome, City_Paris) = **0.79**
sim(City_Rome, Country_Italy) = 0.71

Figure 21: Effect of the type component on the similarity. Cities become more similar in the entity-type space because they share the type *City*

seen a widespread adoption both in the industry[1][2], e.g., in the financial domain [1], and in research, e.g., to analyze biomedical data [76, 80, 146] and historical documents [89, 107].

Thus, starting from a document D that contains a sequences of sentences S_i that are composed of a sequence of words $w_1, \dots, w_n$ we generate an annotated version of each sentence. Note that annotation can be used to replace words with entities, but since not all the words are entities we are left with to possibilities: (i) we obtain a mixed annotated sentence with both words and entities (e.g., "`dbr:Barack Obama` is the president of the `dbr:United States`") or (ii) we remove the words and obtain a entity only sentences "`dbr:Barack Obama` `dbr:United States`". The big difference between these two possibility is that the latter one favors the relatedness between entities (i.e., entities are closer and there is an higher change that they are one in the context of the other). Note that the if the latter one is used, words are lost in the process. This should be taken in account with respect on the task one wants to tackle. In the following, we will use word2vec to generate the entity and type representations.

5.3.2 *Type Embeddings from Text*

Once we have linked words to the entities of a knowledge base $\mathbf{S1_e}$ we can use the knowledge base to retrieve the types of the entities. For example, the DBpedia KG offers data dumps in which each entity is associated to a type[3].

Note that our model is general enough to be applied to any corpus where NEL or NER algorithms are able to extract entities and associate them with a type (e.g., `BBC` is found as an instance of `Organization`). Our model can be therefore used in synergy with these services to compute similarity, relatedness and to explore soft reasoning tasks like analogical reasoning.

1 https://www.expertsystem.com/products/cogito-cognitive-technology/
2 https://dandelion.eu/
3 https://wiki.dbpedia.org/downloads-2016-10

Assuming the existence of a single and unique most specific type for each entity in the KG, we can replace each entity with its own most specific type in the text. Over this type-only virtual document, we can run a word embedding algorithm as we did for the entity embeddings, thus generating embedded representations for the types of a KG. We refer to this representation as the type embeddings (TE).

The assumption of the existence of a single minimal type for each entity is a strong one and it is important to deal with it carefully. In our experiments, we will consider DBpedia and in case of existence of multiple types for an entity, we can randomly select one without preference. There can exist entities that have multiple most specific types, but we experimentally computed that randomly choosing one of the most specific types does not impact the final quality of the representation; this is because the word embedding algorithm encodes all the information and removes much of the noise that is encountered during training. Note that for bigger or essentially different ontologies, this assumption might require revision.

5.3.3 *Generating Typed Entity Embeddings*

The generation of Typed Entity Embeddings (TEE) comes from the fusion of the two process we have just illustrated. We essentially join the processes described in the two previous sections and we concatenate the vector of each entity with the vector of its own type. See Figure 22 for a more detailed representation of the workflow.

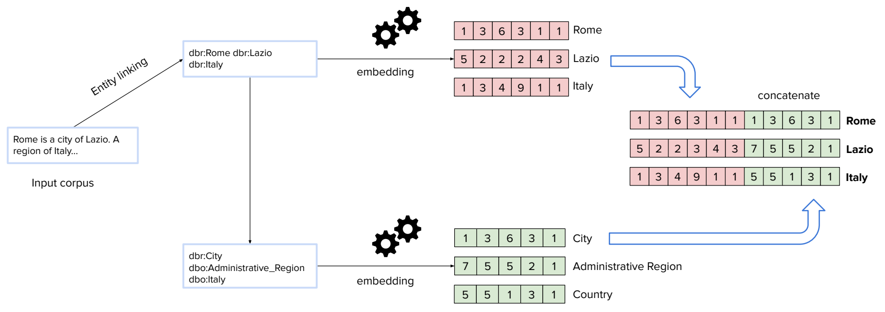

Rome is a city of Lazio. A region of Italy...
Input corpus
Entity linking
dbr:Rome dbr:Lazio dbr:Italy
embedding
1 3 6 3 1 1 Rome
5 2 2 2 4 3 Lazio
1 3 4 9 1 1 Italy
concatenate
1 3 6 3 1 1 1 3 6 3 1 Rome
5 2 2 3 4 3 7 5 5 2 1 Lazio
1 3 4 9 1 1 5 5 1 3 1 Italy
dbr:City dbo:Administrative_Region dbo:Italy
embedding
1 3 6 3 1 City
7 5 5 2 1 Administrative Region
5 5 1 3 1 Country

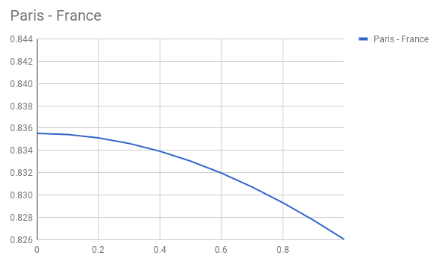
Figure 22: The process that is used to generate Typed Entity Embeddings

5.4 EXPERIMENTS

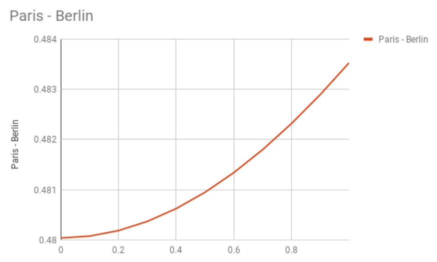
5.4.1 *Experiments on Typed Entity Embeddings*

Distributional Entities and Types Representations

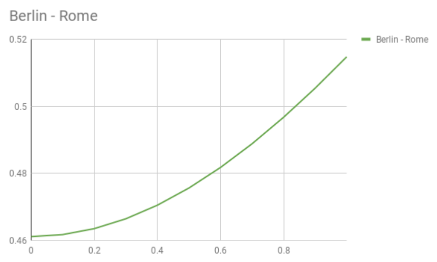
We hereby include some insights related to the effect of the concatenation of types and entity. We multiply the type component with values in a range [0, 1] to show the gradual effect of adding the component over the similarity. Once types are added, the similarity between dbr:Paris and dbr:France decreases (Figure 23a), this is because City and Country have different type representations. The increase of similarity is clear when we consider two entities that have the same type, like dbr:Berlin and dbr:Rome (Figure 23c). In the resulting joint space, similarity brings to different results by giving more results to the type aspect of the vector space. On the other hand, the similarity between dbr:Berlin and dbr:Paris also increases (Figure 23b) even if its respective types are different: City and Settlement. We can explain this behavior by the fact that vector representation of types is the result of an embedding model that according to the distributional hypothesis, consider similar types that appear in similar contexts.



(a)



(b)



(c)

Figure 23: Similarity between the entities Paris-France (a), Paris-Berlin (b) and Berlin-Rome (c) when gradually adding the type component.

5.4.2 *Analogical Reasoning with TEEs*

One task we would like to experiment is analogical reasoning with the entities. We show some example of analogies solved by TEE in Table 14. The interesting pattern here seems to be the ability of the model of retrieving entities of type that is consistent with the input type (i.e., singer-country-singer-country). The type component is the one that makes the analogy give answers that share types with the input. Keep in mind that these analogies return a ranking of answers,

Table 14: Examples of analogical reasoning tasks solved with TEE and with possible explanations

Analogical Operation	Result	Explanation
v(dbr:Oliver_Twist) - v(dbr:Charles_Dickens) + v(dbr:George_Orwell)	v(dbr:Nineteen_Eighty-Four)	Works from different authors
v(Divine_Comedy) - v(Dante_Alighieri) + v(Giovanni_Boccaccio)	v(The_Decameron)	Works from different authors
v(dbr:New_York) - v(dbr:Hudson_River) + v(dbr:Tiber)	v(dbr:Rome)	Rivers from different cities
v(dbr:Demi_Lovato) - v(dbr:United_States) + v(dbr:Romania)	v(dbr:Alexandra_Stan)	Singers from different countries
v(dbr:Barack_Obama) - v(dbr:United_States) + v(dbr:United_Kingdom)	v(dbr:David_Cameron)	Politicians from different countries
v(dbr:Pizza) - v(dbr:Italy) + v(dbr:France)	v(dbr:Galette)	Dishes from different countries
v(dbr:New_York) - v(dbr:Hudson_River) + v(dbr:Tiber)	v(dbr:Rome)	Rivers from different cities

that are the closet elements to the point to which the analogical operation brings us in the space (i.e., we do a neighborhood search).

Experimental set-up

Dataset.

- **Countries-Currencies** (**CI-E and CU-E**) To test our models, we used a modified version of the dataset originally proposed by Mikolov et al [93], that contains analogies between cities and countries like *Rome:Italy:?:France* and between countries and thier respective courrencies. We modifed this dataset by manually replacing words with their entity identifier (for example, Georgia, was replaced with dbr:Georgia_(Country)), the DBpedia identifier for Georgia.

- **People** (**P**) contains 102 analogies with entities identifier derived from *prime_minister-country* relation and *CEO-company* relations. For example, *Barack Obama*:*United States*::*David Cameron*:*United Kingdom* or *Mark Zuckerberg*:*Facebook*::*Elon Musk*:*SpaceX*. This dataset is generated by considering people positions held by those people in 2016.

Evaluation Measures. On the datasets *CU-E and CI-E* we used accuracy as in [93]. On the *People* dataset, we used HITS@K as we will show that the task is more difficult. HITS@K of a given analogy is 1 if the answer to that analogy is found in the top-k list of elements returned by the analogy.
Comparison. We show the comparison our models with different models proposed in the state of the art. The following results have been taken from [67]:

- EECS* model [67];

- TransE*, TransH*, TransR* [25, 81, 134].

In particular we compare with two baseline model with three baselines which precise objectives:

- SkipGram-Clean (skip-gram [93] on preprocessed text). Objective: evaluate the performance of a word based model on word analogies vs TEE on entity analogies. We want to understand if analogies are easier to solve in an entity distributional space. We will use the word analogies provided by [93] and we will refer to them as CI-W and CU-W for cities and currencies).

- Wiki2Vec [4], a model based on entities generated from user made links in Wikipedia. Objective: study the difference in performance between a model that is based on human annotation and our model that uses an automatic annotator. This model contains both words and entities, but we remove words in this setting to make the comparison fair. This model has been used as a baseline approach of several works (see for example, [17, 143, 150]).

Note that the comparison between SkipGram-Clean and TEE is not fair: one deals with words and one deals with entities. This experiment should suggest the difference between the two approaches but at the same time we want to remark that the experimental settings of the two differs heavily. This is also why we will underline the difference in the Table that contains the results.

The SkipGram-Clean model will be trained on the DBpedia's abstracts. For the P dataset, we test only TEE and EE.

Parameters. Since word embeddings algorithms require window and feature size parameters we tested different configurations of the algorithms. All the embeddings were generated using the SkipGram model of word2vec [93]. For our EE model w_e and s_e will represent window and size while TEE will also consider the size of the type embeddings w_t and s_t. The mapping function was run with $\alpha = 0.75$ to favor the similarity between the first two elements of the analogy. After different experiment we report the best model for each algorithm: TEE: window entities = 3 entity dimension = 100, window types = 3 type dimension = 200; EE window = 3 and dimension = 100; SkipGram-cleaned window = 5 and dimension = 400.

Corpus and Preprocessing. Our embeddings are generated 2016 DBpedia's abstract[5]. In this dataset there are 4M description of entities, we consider each abstract as a sentence to we pass to the word2vec algorithm. To make the comparison fairer, the corpus that was fed to the SkipGram-Clean algorithm has been pre-processed using standard techniques for text analysis (this motivates the *Clean* in the name). We removed stopwords using the English list found in the NLTK[6] package and we remove punctuation characters from the text, citations and numbers.

For the algorithms that are based on the entities no pre-processing has been done to the corpus, since all the work is left to the entity

4 https://github.com/idio/wiki2vec
5 http://wiki.dbpedia.org/downloads-2016-04
6 http://www.nltk.org/

Table 15: Results on the analogical tasks for knowledge graph embeddings from the paper in which the EECS model was introduced [67]. KG embeddings that start from structured representations are less effective on the analogical reasoning task.

Embedding	**Total**
TransE*	0.382
TransH*	0.382
TransR*	0.378
EECS*	0.595

linking system that has to annotate the entities. We use DBpedia-Spotlight[7] as NEL system because it is a well-known open source NEL algorithm that can be used without limitations, thus making our experiments easy to replicate. We empirically found that other NEL systems perform better than DBpedia-Spotlight (Wikifier, TextRazor, Dandelion) but they are subject to more restricted usage policies.

CORPUS SIZE The entity embeddings are trained over a corpus that contains 45M tokens while the word corpus contains 228M tokens. Thus, word embeddings have much more information to learn from. The number of distinct tokens for the entity corpus is 1.8M while the number of distinct tokens for the word corpus after pre-processing is 5.5M. As often done in word embeddings for both corpora we remove those words that appear less then 5 times in the corpus. The result of this last operation is that we have 770K unique entities and 900K unique words.

Structured Knowledge Graph Embeddings on Analogical Reasoning

We show results of the paper in which the EECS model was introduced [67] in Table 15. In which the 8363 analogies that contain entities were tested[8]. These results show that for these embeddings is difficult to solve analogical reasoning tasks; we believe that this happens because these models ignore the information that is present in the text that is valuable for this task. Nevertheless, it is important to remark that all these models encode relationships that are missing from our model.

Results on CI-E and CU-E

Figure 16 shows the results for the selected models. The first important result is that entity analogies are solved with higher precision

7 http://demo.dbpedia-spotlight.org/
8 They used 25% of the analogies for tuning the model and 75% for the actual testing.

Table 16: Results on the analogical tasks with respect to the entity-based dataset and the word-based dataset.

	CI-E	CU-E	Total
TEE	0.94	0.70	**0.92**
EE	0.87	0.10	0.77
Wiki2Vec	0.79	0.45	0.75
	CI-W	**CU-W**	**Total**
SkipGram-Clean	0.87	0.19	0.80

in TEE, and this is true for both CI-E and CU-E. The performance on CU-E significantly outperforms the other algorithms. The test on this last dataset is really important because it shows that it is a more difficult task to solve (currency and country also appear with a different frequency than city and country). Thus, one first significant result of this work is that using TEE helps in solving entity analogies, obtaining an accuracy on the complete dataset equal to 0.92.

The EE model seems to be less efficient than the Wiki2Vec baseline on the CU-E dataset. This might be because currencies are a difficult entity to disambiguate and their representation in the EE model is not perfect. However, adding the type to the TEE model greatly increase the performance, showing that the added information can be of great help in this task.

SkipGram-Clean reaches an high accuracy on the task with words. This is also due to two factors: (i) the text is heavily preprocessed, without preprocessing the accuracy is around 60%, that is a result close to what has been reported in the literature[9], (ii) the annotation phase might not be perfect and EE might suffer from noisy values (we will evaluate the effect of the annotation in the next section). Nevertheless, TEE reaches the best accuracy on the task with the entities.

Results on People

This task is more difficult than the first two and thus we show the HITS@K for K in [1, 5, 10, 15, 30] in Figure 24, in the legend we show the models' names with a compacted representation that also shows hyperparameters. With HITS@K the answer to the analogical reasoning task is correct if the answer is in the top-K list returned by the models after the operations. TEE outperforms EE. This is due to the fact that the type in the analogies are consistent: v(`dbo:Person`) - v(`dbo:Organization`) + v(`dbo:Organization`) brings in a space near other entity of type similar to `dbo:Person`. HITS@1 shows that only the 35% of the analogy are correctly solved. This is due to ambigu-

9 https://aclweb.org/aclwiki/Google_analogy_test_set_(State_of_the_art)

ity in the analogies: when asking for the respective person of Mark Zuckerberg in Apple Inc., our models often replies with John Sculley (CEO in the nineties) and Steve Jobs. Tim Cook, the current CEO, appears later in the resulting list.

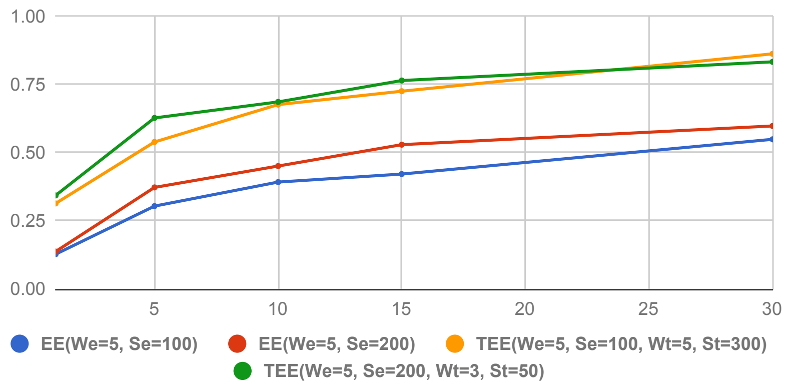


Figure 24: HITS@K on People dataset

5.4.3 *Robustness of Entity Linking for TEE*

Note that, on the other hand, it is important to have a valid entity linker. DBpedia Spotlight offers a *confidence* parameter that can be adjusted to find entities with a certain degree of correctness. This threshold is between 0 and 1 and is usually initialized with 0.5.

In this section we evaluate the effect of the DBpedia Spotlight confidence on the annotation. The quality of the annotation is a fundamental aspect of the process because it affects the generation of the embedding. We focus on DBpedia Spotlight and we annotate DBpedia 2016 Abstracts.

Spotlight comes with a confidence threshold parameter (range between 0 and 1) that can be used to cut out those annotations for which the linker is less confident on annotating. As a side effect, an higher confidence means few but good annotation (precision) with the risk of not having some annotations in the text and losing tracks of some entities, a lower confidence instead might bring to noisy results (recall).

Experimental Setup We annotate the abstracts using ranging confidence from 0 to 1 (with steps of size 0.1).

As a first result, Table 17 shows a comparison between the number of both multiple and distinct tokens found in text with respect to the confidence parameter given by spotlight.

As expected, as the confidence threshold gets lower, the number of retrieved entities increases, on the other hand this might have an impact on the quality of the embedding.

	C_0	$C_{0.2}$	$C_{0.4}$	$C_{0.6}$	$C_{0.8}$	C_1
Token	128M	125M	54M	38M	29M	6M
Distinct	1,881K	1,875K	1,823K	1,654K	1,397K	746K

Table 17: Tokens and distinct tokens found by DBpedia Spotlight over the Abstracts

	C_0	$C_{0.1}$	$C_{0.2}$	$C_{0.3}$	$C_{0.4}$	$C_{0.5}$	$C_{0.6}$	$C_{0.7}$	$C_{0.8}$	$C_{0.9}$	C_1
Accuracy CI-E	0.83	0.84	0.84	0.84	0.89	0.87	0.84	0.76	0.72	0.6	0
Missing CI-E	0	0	0	0	0	0	0	0	0	117	7,465
Accuracy CU-E	0.15	0.18	0.17	0.16	0.11	0.10	0.07	0.05	0.03	0.02	0
Missing CU-E	0	0	0	0	0	0	0	0	29	58	824

Table 18: Results of embeddings that have been annotated with different confidence scores of DBpedia Spolight.

To evaluate the quality of each annotated corpus C_i we embed it in a 100 dimensional vector space with the skip-gram algorithm (window = 5) and we use the CI-E and the CU-E datasets to evaluate the accuracy of the models. Actually, we want to check two things: 1) how well are analogy solved and 2) how many analogy we cannot solve because of missing entities (that might be removed by the linker if it is not too sure about the annotation). See results in Table 18. Results on this simple dataset seem to show that the effect of the annotation does not impact too much on the final outcome: low confidence still brings to a good amount of solved analogies. This means that word2vec can handle a bit of noise. On the other hand, having a confidence of 1 makes it impossible to solve analogies since for each analogy at least one element is missing from the generated representation.

5.4.4 *Properties of Type Embeddings*

In this section, we investigate some properties of the distributional type representations. We first presented these properties in a workshop paper [19]. To the best of our knowledge we were the first to define an approach that embeds types following a distributional framework. These experiments are meant to prove the properties of types in the vector space. We show some qualitative results related to these representations and we also report another possible application of the type embedding in the context of a class matching task in ontology in which we try to give some intuition about how to use TE to map two ontologies in the same space. While these experiments are not entirely related to the corpus-based comparative model, they give some indications on how this newly introduced model for type embeddings behaves.

Data We use 200-dimensional embeddings for the first experiments (window = 5). While we used Wikidata 2016-04 (another knowledge graph that uses a different set of URI to maps resources on the web) dumps[10] to project two different categorization systems.

Qualitative Analysis

We compare the similarity computed in TE using cosine similarity with the wpath measure [151] a measure recently introduced to compute the ontological similarity of concepts in a hierarchy that does not consider distributional information over types. Table 19 shows the comparison between a few types. It is clear that the similarity captured by TE is closer to a relatedness (`SoccerPlayer` and `SoccerClub`) but that also shows a strong characteristic of structural similarity: `Vein` and `Artery` are very similar in TE, this is because they tend to appear in similar context in text. In our paper [19] we experimentally show that the similarity in TE is lowly correlated with other ontological similarity measures that are based on the structure of the ontology [19].

Table 19: Comparison of similarities between the wpath measure and the similarity computed in TE

Type 1	**Type 2**	wpath sim	TE sim
`dbo:SoccerPlayer`	`dbo:SoccerClub`	0.17	0.72
`dbo:SoccerPlayer`	`dbo:Wrestler`	0.47	0.24
`dbo:RailwayLine`	`dbo:Station`	0.44	0.81
`dbo:Vein`	`dbo:Artery`	0.70	0.84
`dbo:RailwayLine`	`dbo:PublicTransitSystem`	0.11	0.79
`dbo:Company`	`dbo:Airline`	0.72	0.30

User Study on Type Similarity

We studied the effect of the similarity between types by considering a simple categorization task in which we involved 5 users that had already some experience with the semantic web.
Methodology We selected 31 nodes from the DBpedia Ontology and for each one we retrieved its most similar sibling and its least similar sibling (which correspond, respectively, to the nearest and to the farthest siblings in the space). For example, the most similar sibling of the ontological type `dbo:President` is the ontological type `dbo:PrimeMinister`, while the least similar is the ontological type `dbo:Mayor`. Users were given the first node (`dbo:President`) and were asked to decide which of the two siblings they considered more simi-

10 https://tools.wmflabs.org/wikidata-exports/rdf/exports/20160425/

lar. Users were forced to give an answer even in contexts in which it was not immediately clear which element was the most similar (e.g., is `dbo:Skyscraper` more similar to `dbo:Hospital` or `dbo:Museum`?). A strong bias in this experiment is that the two available options were chosen by considering their position in the vector space.
Results Resulting categorizations provided by user were quite similar, the 5 users agreed on many questions. Since the agreement between the user was high we used Gwet AC1 [59] to compute the level of agreement between users, obtaining a level of agreement equal to 0.9, not distant from 1 (that represents unanimity). If we consider the majority vote on the collected answers, we see that the answer is always the most similar sibling. This is an interesting result because it shows that this corpus-based similarity on concepts can capture human-like behavior even if this is a really simple experiment. This result is similar to other results obtained in the word embeddings literature [77].

Comparing Different Classification Systems

In this section we explore a simple applications of our distributional vectors of types to ontology matching, i.e., the task of matching types of two different ontologies. We generate the embeddings of two different ontologies in the same space and then we compare different types.

The process we follow is similar to the one described in Section 5.3.2 with a slight modification. Given the virtual document that contains the sequences of entity mentions we randomly replace each entity with one of two most specific types coming from two different ontologies: we select with probability 0.5 the type coming from the respective Wikidata entity[11] or the type of the entity itself in DBpedia. We thus generate a document that contains sequences of mixed types coming from two ontologies. Our intuition suggests that our embedded representations should show equivalent types near to each other because those types are used in the same contexts. In this setting, we generate 100-dimensional representations using the Skip-gram algorithm.

In Table 20 we show the pairs of wikidata-dbpedia (note the wikidata ids are formed with a Q followed by a series of numbers) types that are most similar one with the other in the embedded representation. The pairs that are not marked are the ones for which an equivalent class relation exists in DBpedia[12]. Consider that since we are just using word2vec we are not considering any syntactic or topological information to find this mappings.

11 We do this by first mapping DBpedia's URIs to Wikipedia's URIs, and then to Wikidata's URIs, where we can get the type of the entity.
12 DBpedia already contains some mapped concept from its ontology to wikidata.

Table 20: Top similar Wikidata entity and DBpedia class pairs

Wikidata (label)	DBpedia	Sim
Q4498974 (ice hokey team)	HockeyTeam	0.99
Q5107 (continent)	Continent	0.99
Q17374546* (Australian rules football club)	AustralianFootballTeam	0.99
Q3001412* (horse race)	HorseRace	0.98
Q4022 (river)	River	0.98
Q46970 (airline)	Airline	0.98
Q18127 (record label)	RecordLabel	0.98
Q13027888* (baseball team)	BaseballTeam	0.98
Q11424 (film)	Film	0.98
Q1075* (color)	Colour	0.98

An interesting property is that equivalent classes not defined in the DBpedia KG are also found has highly similar. Some examples of these types are reported in Table 21.

Table 21: Examples of similar types that do not have an equivalent class relation. Some elements have been shortened.

	Nearest Point	Label	Sim
`dbo:AmericFootTeam`	Q17156793	American football team	0.95
`dbo:Earthquake`	Q7944	earthquake	0.91
`dbo:Diocese`	Q3146899	diocese of the Catholic Church	0.93

5.5 SUMMARY OF THIS CHAPTER

We hereby give a short summary of this chapter, by answering the research questions we asked ourselves in the introduction.

- **Q5.1** can we create purely distributional models of KG elements (i.e., entity and types) that can be applied on any text?
- **Q5.2** is the notion of similarity computed within this space good enough to support similarity-based approaches to reasoning like analogical reasoning?
- **Q5.3** which is the effect of the entity linking phase on the embeddings?
- **Q5.4** which are the properties of a distributional representation of types?

In this Chapter, we have introduced TEE a distributional knowledge graph embedding model that is purely based on entity linking (**Q5.1**). This fact favors the applicability of the model to any kind of text, and in fact we will see the combination between TEE and CADE in Chapter 7. We saw that this model provides good result over the analogical reasoning task and this is mainly due to the combination of entities and types (**Q5.2**). We tested the effect on the entity linking experimenting on DBpedia Spotlight realizing that in general the impact of the final representation does not seem too affected by small variations in the confidence threshold of the annotator, but extreme values of that threshold can ruin the quality of the embeddings (**Q5.3**). Eventually, we have shown some properties of the type embeddings, the most important finding is that the similarity computed with TE is different from other measures from the state-of-the-art (**Q5.4**).

6 REASONING WITH DISTRIBUTIONAL EMBEDDINGS

In this Chapter, we show how the representation defined in Chapter 5 can be combined with reasoning system. Key aspect of this chapter is to show how we can include logical consequences inside a distributional representation. We will show results that have been previously described in our paper [20]. Note that this experiment is limited to a small test knowledge base, but we believe that the results can be generalized; in Chapter 7 we will show some examples related to how this system can be used in a comparative setting.

Note, we will use a tool defined logic tensor networks [114], in this paper the term grounding is used to describe the vector representations of the constants, but in logic has a specific definition. To keep consistent with the paper we will use the term grounding to refer to the vector representations of the constants, and we will refer to axioms like $\mathrm{mammal}(\mathrm{tiger})$ as *instantiated axioms*.

6.1 NEURO-SYMBOLIC LEARNING AND REASONING

Neuro-symbolic integration models [47, 48] aim at combining properties of symbolic reasoning and neural networks, to account both for data-driven learning and high-level reasoning, two tightly related aspects of human cognition. Additional advantages of this combination can be found in a higher explainability of learned knowledge and in the capability of softening some aspects of crisp logic-based reasoning approaches. This integration is also connected to the combination of sub-symbolic perception with high-level reasoning, a critical task in artificial intelligence [85].

LTNs [38, 114] are an example of a neuro-symbolic model that embeds first-order fuzzy logic in a vector space. In LTNs logic constants are represented as vectors and n-ary predicates are n-ary functions whose values are real numbers in the range $[0, 1]$. A neural network for each predicate learns both the representation of logic constants and the weights that characterize the n-ary function. Learning is based on a set of axioms.

As we have introduced in Chapter 2 computational linguistics has developed distributional models of language that have been found cognitively plausible at a large extent by psychologists [77]. We believe that these models, once adapted to be easily integrated with existing logical frameworks that combine learning and reasoning, can provide an account for both distributional knowledge and structured inferences that go beyond crisp reasoning approaches.

In this Chapter, we combine entity embeddings with axiomatic knowledge has the one found in knowledge bases that accounts for structured inference. Imagine an agent that has access to the following set of axioms {$\mathtt{species}(\mathtt{cat})$, $\mathtt{mammal}(\mathtt{tiger})$, $\mathtt{bird}(\mathtt{penguin})$, $\forall x : (\mathtt{mammal}(x) \rightarrow \mathtt{animal}(x))$}, we refer to the first three as instantiated atoms and to the latter one as universally quantified formula; with axiomatic knowledge in input the agent cannot infer instantiated atoms as $\mathtt{mammal}(\mathtt{cat})$. However, if the agent knows that cats and tigers are similar to each other and both are dissimilar to penguins, she might be able to infer that cats are mammals too (i.e., $\mathtt{mammal}(\mathtt{cat})$). Once the latter axiom has been inferred, the agent can make use of the universally quantified axioms $\forall x(\mathtt{mammal}(x) \rightarrow \mathtt{animal}(x))$ to infer that cats are also animals, bridging the gap with more complex inferences, see Figure 25 for a schematic representation of this idea. We believe that combining these two worlds would bring great benefits in reasoning approaches since one requires the help of the other.

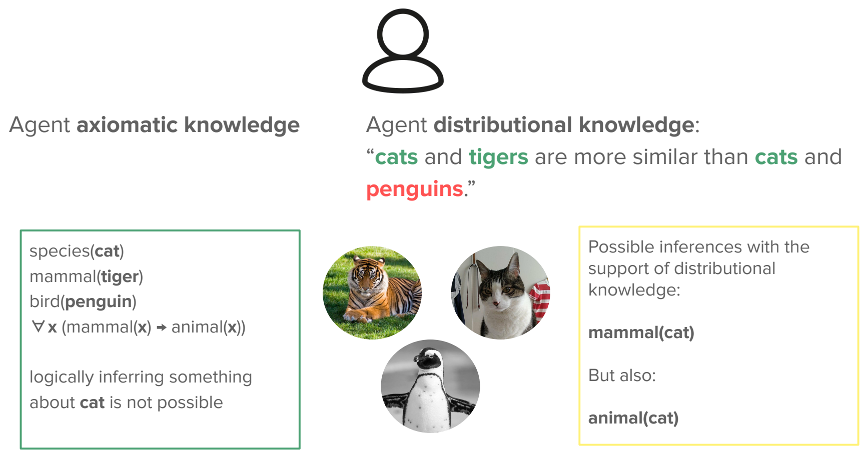

Agent **axiomatic knowledge**
Agent **distributional knowledge**:
"**cats** and **tigers** are more similar than **cats** and **penguins**."
species(**cat**)
mammal(**tiger**)
bird(**penguin**)
∀**x** (mammal(**x**) → animal(**x**))
logically inferring something about **cat** is not possible
Possible inferences with the support of distributional knowledge:
mammal(cat)
But also:
animal(cat)

Figure 25: Idea that drives the combination between distributional knowledge and LTNs.

We present an approach that feeds Entity Embedding (EEs) generated using distributional semantics introduced in Chapter 5[1]. We can describe this work in one paragraph: we show that combining knowledge under the form of text-based entity embeddings with LTNs is not only simple, but it is also promising because it can cover aspect not covered by standard logical approaches. Our experiments explore a limited part of a knowledge base but results show that the model is flexible and can be useful under different settings and use-cases.

1 In the experiment we will use EEs in place of TEEs

Related Work

While related work has been explored in Chapter 3 we want to summarize related work of the statistical relational learning and the neuro-symbolic fields in this section. We refer to recent surveys for discussions of different neuro-symbolic approaches proposed in the literature [47, 48], but note there are several work in this field. For example DeepProbLog [85] is a "deep" extension of ProbLog language [34] and the Neural Theorem Prover (NTP) was introduced as an extension of the Prolog language that supports soft unification rules with the use of similarity between embedded representations [105]. Different symbolic/statistical relational learning approaches have been introduced in literature to treat inference; For example, Probabilistic Soft Logic (PSL) [4] are an example of a statistical relational learning model that comes from the family of Markov Logic Networks (MLNs) [91]. Note that the combination between distributional semantics and logic has also been studied in what is now called *formal distributional semantics* [22].

We decided to focus on LTNs over other methods because it is completely based on vector space grounding, that is where our entities are located.

6.2 LOGICAL REASONING WITH ENTITY EMBEDDINGS

We propose a method to combine logical reasoning in the vector space with entity embeddings. We decided to focus on LTNs because the integration of embedded knowledge is straightforward: LTNs gives us the advantage representing first-order logic inside a vectors space and at the same time we can use entity embeddings as the starting vector space on which LTNs learn to do reasoning.

Logical Reasoning with Logic Tensor Networks

LTNs [38, 114] use first-order fuzzy logic and represent terms, functions, and predicates in a vector space. Connectives are binary operations over real numbers in $[0, 1]$. *T-norms* are used in place of the conjunction from classical logic (e.g., the t-norm can be interpreted as the min between two truth values). In LTNs we call grounding the action of representing elements of the logic language in the vector space. The operation $\mathcal{G}()$ will identify the grounding.

In LTNs, constants are grounded to vectors in $\mathbb{R}^n$ while predicates are grounded to neural networks that generate output values in $[0, 1]$. The neural network learns to predict the truth value of an atom $P(c_1, \ldots, c_n)$ as a function of the grounding of the terms $c_1, \ldots, c_n$ [114]. For a predicate of arity m and for which $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ are the groundings of m terms, the grounding of the predicate is defined as

$\mathcal{G}(P)(\mathbf{v}) = \sigma(u_P^T(tanh(\mathbf{v}^T \mathbf{W}_P^{[1:k]} \mathbf{v} + \mathbf{V}_P \mathbf{v} + B_P)))$ where $\mathbf{v} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_m \rangle$ represents the concatenation between the vectors $\mathbf{v}_i$, σ is the sigmoid function, **W**, **V**, **B** and u are parameters to be learned by the network while k is layer size of the tensor. It is easy to see that predicates are neural networks that can take in input constants and provide an output value in $[0, 1]$.

In LTNs training is interpreted as a maximum satisfiabilty problem: the task is to find groundings for terms and predicates that maximize the satisfiability of the formulas in the knowledge base. For example, for a grounded formula like $mammal(cat)$, the network updates the representation of the predicate $mammal$ (i.e., the parameters of the network) and the representation of cat (i.e., its vector) in such a way that the degree of truth of an instantiated atom is closer to 1. Optimization also works with quantified formulas as the following one $\forall x(mammal(x) \to animal(x)$; In fact, the universally quantified formulas are computed by using an aggregator defined over a subset of the domain space $\mathbb{R}^n$ [114]. LTNs can be be used to do after-training reasoning over combinations of axioms on which it was not trained on (e.g., ask the truth value of queries like $\forall x(\neg mammal(x) \to species(x))$; LTNs offer a method that is entirely compositional and its prediction capabilities will will come useful in Chapter 7 where we will do comparative reasoning over aligned distributional spaces.

Combining Entity Embeddings and Logical Reasoning

In our distributional space, logical constants are represented by a vector and thus we can use them inside LTNs to learn the representations of predicates. We use entity embeddings $\mathbf{e_1}, \ldots, \mathbf{e_m}$ of the set of entities E as vectors to feed to LTNs. The truth value computed by LTNs is function not only of the parameters of the networks but also of the distributional embeddings. LTNs learn the representation of the constants from scratch, but in this setting we start from an already pre-trained representation that we keep fixed (i.e., frozen) and that we do not update over time[2].

Figure 26 shows a summary of the components of our model. We generate distributional embeddings from text and then we use axiomatic knowledge to learn the representations of predicates. After training we can use the model to reason over new axioms. An intuitive way of understanding how LTNs work is to consider the learned predicate as an area for which vectors have a truth value of 1 in cer-

2 Updating vectors will move them in the space, if at test time we want to test distributional embeddings unseen in the network, we will obtain unexpected results because that distributional vector will no longer be aligned with the other constants that have been updated

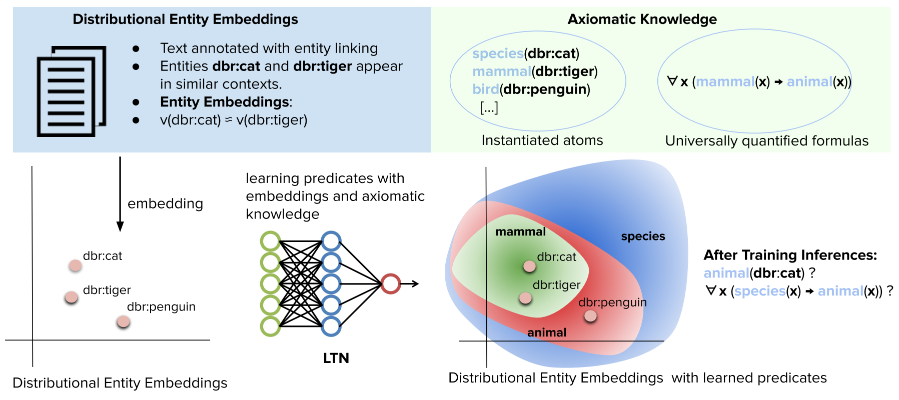

Distributional Entity Embeddings
Text annotated with entity linking
Entities dbr:cat and dbr:tiger appear in similar contexts.
Entity Embeddings:
v(dbr:cat) ≈ v(dbr:tiger)
Axiomatic Knowledge
species(dbr:cat)
mammal(dbr:tiger)
bird(dbr:penguin)
[...]
Instantiated atoms
∀x (mammal(x) → animal(x))
Universally quantified formulas
embedding
dbr:cat
dbr:tiger
dbr:penguin
Distributional Entity Embeddings
learning predicates with embeddings and axiomatic knowledge
LTN
mammal
species
animal
After Training Inferences:
animal(dbr:cat) ?
∀x (species(x) → animal(x)) ?
Distributional Entity Embeddings with learned predicates

Figure 26: We learn embeddings from text and we use LTNs to learn how to represent predicates with the network.

tain locations that decreases to values close to 0 when vectors are distant from those locations.

6.3 EXPERIMENTS

In this experimental section we report the experiment that we run in the paper in which we introduced this combination of the two methods [20], this experiment should show the interesting effects of the combination between LTNs and entity embeddings. We use 100 dimensional DBpedia entity embeddings (window = 5) that have been described in Chapter 5. Note that in this experiment we do not consider the type component of the embeddings.

6.3.1 *Reference Knowledge Base*

We create three small knowledge completion tasks for our experiments that are based on a common reference knowledge base introduced in this paper to prove the capabilities of the combination of distributional representations with LTNs.

Our main knowledge base is a subset of DBpedia and contains: a set of predicates P (e.g., $\mathtt{mammal}$); a set of constants C (e.g., dbr:cat[3]); a set I of instantiated atoms, i.e., facts such as (e.g., $\mathtt{mammal}(\text{dbr:cat})$); a set Q of universally quantified formulas that represent the sub-type dependencies DBpedia ontology (e.g., $\forall x\ \mathtt{mammal}(x) \rightarrow \mathtt{animal}(x)$); the set I^Q of formulas closed under the application of standard FOL inference to the previous set I (e.g., $\mathtt{animal}(\text{dbr:cat})$ ant others); a set of negated I^N instantiated atoms that are derived as follows: all the instantiated atoms built with predicates in P that are not in I^Q

3 We are aware that in some cases there is a subtle difference between what can be considered an instance and what is instead a type; *cat* can be for example the type of all the instances of cats. This generally depends on the granularity of the knowledge base and we think that this does not affect the scope of this experiment.

and I (e.g., $\neg \mathit{fungus}$(dbr:cat)). The reference knowledge base D is $I \cup I^Q \cup I^N$.

We extract entities (C) from DBpedia considering instance of the following classes[4]: Mammal (0.38%), Fungus (0.17%), Bacteria (0.03%), Plant (0.42%). We add the universally quantified formulas Q to derive inferences for predicates Animal, Eukaryote and Species for each atom, and apply these axioms to generate the set of instantiated axioms I^Q. Finally, we also generate all the negative instantiated atoms in I^N (e.g., $\neg \mathit{fungus}$(dbr:cat)). If we consider positive and negative axioms this reference knowledge base contains 35,133 instantiated axioms. We split the D knowledge base in three different subset that are meant to describe three different tasks with both training and test data:

1. **D1**. **Objective:** evaluate the performance of the algorithms in a task in which only positive atoms are given, not all the atoms can be inferred with logical rules (as in Figure 25). We train with 1,400 positive atoms and we ask the models to find the truth value of the other 7,077 atoms related to the entities found in the 1,400 atoms.

2. **D2**. **Objective:** evaluate the performance of the algorithms in a task in which both positive and negative atoms are given; each entity in the training set appears also in the test set. We train with 7,026 atoms (both positive and negatives) and as in D1 we ask the models to infer the truth value of 20,890 atoms (positive and negative).

3. **D3**. **Objective:** evaluate the performance of the algorithms in a task in which both positive and negative atoms are given, but the test set will also contain atoms of entities not present in the training set: The models will need to rely on the entity embeddings to make the prediction. We train with 1,756 atoms and the models are now asked to infer the truth value of 33,377 atoms (positive and negative).

6.3.1.1 *Domain Theory*

We here list the universally quantified axioms that are used by the relational learning models. In total we use 22 axioms, but note that this set is different from the set Q: the models will not have the possibility of using axioms like $\forall x : \mathit{animal}(x) \rightarrow \mathit{species}(x)$. We ask LTNs to learn something more than simple axioms when combined with distributional embeddings.

- $\forall x(\mathit{plant}(x) \rightarrow \mathit{eukaryote}(x))$

4 Note that some classes are much less represented than others

- $\forall x(\mathrm{mammal}(x) \rightarrow \mathrm{animal}(x))$
- $\forall x(\mathrm{plant}(x) \rightarrow \neg\mathrm{mammal}(x))$
- $\forall x(\mathrm{fungus}(x) \rightarrow \neg\mathrm{animal}(x))$

We refer to the model that combines LTN and EE as LTN_{EE}. **Baseline.**

- A simple LTNs model that does not use EE. This model will be useful to evalaute the capability of the model trained without embeddings.
- Probabilistic Soft Logic[5] [4] that will be trained on both atoms and universally quantified formulas. We use that standard configuration provided by the authors in the tool.
- Deep Neural Network (DNN) initialized with EEs trained to assign 0 or 1 to instantiated atoms (since this is a simple DNN we cannot use universally quantified formulas here). The DNN embeds the pre-trained representations of entities and a one-hot representation of predicates in 20 dimensions, concatenate both those representations together and apply another transformation to generate a 1 dimensional representation. We apply a sigmoid function as non-linearity and train with a binary cross entropy loss. Validation is done on 20% of the input data. Note that DNN cannot make use of the axioms of the domain theory.

6.3.2 *Results*

Table 22 shows the results of the models over by computing F1 scores on the datasets we defined above.
Experiments on D1 In this setting we compare LTN_{EE} with LTN and PSL. We exluce DNN from this setting because we do not have negative data on which to train the network on. We also add that a simple rule-based model that uses the inferential FOL rules to complete the knowledge base would be able to infer only 45% of the axioms (with a 100% precision), all the other axioms are similar to those presented in Figure 25. The LTN_{EE} approach is the best performing one.
Experiments on D2 In this experiment each entity for which we require to find other instantiated atoms appear at least one time in the training set. PSL performance is more or less similar to the one shown for the D1 dataset, instead the performance between LTN_{EE} and DNN is comparable.
Experiments on D3 From this experiment it is clear that LTN_{EE} generalizes slightly better than DNN, one possible reason is that we are

5 https://psl.linqs.org/

using a domain theory to train LTN_{EE}. While the F1 scores are comparable, the ones of the classes Fungus and Bacteria got a lower score than in the previous experiment. However, note that this might be due to the fact that the representation of elements of the class Fungus are similar to those of the class Plant while the Bacteria class as only a few instances in this experiment.

We stress the fact that LTN_{EE} can be used to do after-training logical inferences. This is a very important aspect to include in our comparative framework (in Chapter 7 we will show how this can be used to do logically compare representations).

Table 22: F1 score per tested class.

D1	A_{F1}	F_{F1}	M_{F1}	P_{F1}	B_{F1}	E_{F1}	S_{F1}
LTN_{EE}	**0.81**	**0.74**	**0.84**	**0.66**	**0.52**	**0.97**	**1.00**
LTN	0.40	0.14	0.12	0.10	0.03	0.93	**1.00**
PSL	0.54	0.19	0.15	0.14	0.07	0.93	**1.00**
D2	A_{F1}	F_{F1}	M_{F1}	P_{F1}	B_{F1}	E_{F1}	S_{F1}
LTN_{EE}	0.91	**0.86**	0.91	0.86	**0.63**	**0.99**	**1.00**
DNN	**0.93**	0.82	**0.93**	**0.87**	0.54	**0.99**	1.00
PSL	0.56	0.20	0.20	0.17	0.10	0.88	0.98
D3	A_{F1}	F_{F1}	M_{F1}	P_{F1}	B_{F1}	E_{F1}	S_{F1}
LTN_{EE}	**0.88**	**0.80**	**0.89**	**0.82**	**0.60**	**0.99**	**1.00**
DNN	0.87	0.64	0.85	0.77	0.47	0.98	1.00

Some examples

After training we can evaluate the truthfulness of axioms for which it was not specifically trained on. Table 23 reports some examples (see Chapter 7 for more examples). Note that LTNs still have some limits due to training problems, we have highlighted some of this in a recent work [16]. In general, it seems that the use of quantifiers has a big impact on the training time of LTNs and thus they should be used carefully.

Table 23: The truth values of novel axioms.

Axiom	**Truth**
$\forall x(\mathtt{species}(x) \to \mathtt{animal}(x))$	0
$\forall x(\mathtt{eukaryote}(x) \to \neg\mathtt{bacteria}(x))$	0.73
$\exists x(\mathtt{eukaryote}(x) \wedge \neg\mathtt{plant}(x))$	1

6.4 summary of this chapter

We hereby give a short summary of this chapter, by answering the research question we asked ourselves in the introduction.

- **Q6.1** can we provide a method to do logical inference over distributional representations?

In this Chapter, we have introduced a combined model for reasoning over distributional representations. With a simplified but intuitive experiment we have shown that this method is able to combine notions of first order logic with distributional elements, thus providing an account for both distributional knowledge and structured inferences (**Q6.1**). The main outcome of this chapter is that we now have a method that can be used to do logical reasoning over distributional representations, and that can be used to enhance our comparative framework.

additional resources

- [48] offers a recent analysis of novel framework for neural-symbolic learning and reasoning. [22] describes combinations of distributional semantics and logical systems. For combinations between neuro-symbolic systems and the semantic web we suggest the following related paper [64].

7 APPLICATIONS

In this chapter, we show some application of the comparative framework introduced in the previous chapters. We show how CADE can be used to do knowledge exploration with both words and TEE. Note that while in Chapter 5 we used the Skip-gram model to learn TEE in this Chapter we used the CADE model that is based on CBOW. This is because we generally rely on small amounts of text and in our experiments, CBOW seems to obtain good results in these settings; we will use CBOW to embed also the TEE representations. To use TEE on CADE we annotated text using the commercial entity linker Dandelion[1]. This entity linker is more accurate than DBpedia Spotlight, but since it is a commercial tool only a limited number of API calls to annotate text can be executed for free.

We recall that to find the corresponding token, we use the function $\phi_{D_i \to D_j}$ and that we will use the simplified version $D_1 \to D_2$ in this Chapter. We will generally look at 5-top elements in the neighborhood of the D_2 space. To show that this model is actually learning something that is not trivial, we will also show the neighborhood of the original space using the notation $D_1 \to D_1$: if the closest word to “flat” in NYT was “apartment” there would not be much to learn.

7.1 COMPARATIVE KNOWLEDGE EXPLORATION

7.1.1 *Comparative Transposition*

Another task we would like our model to solve is the mapping between words that have the same meaning, but completely different spellings in the two languages (e.g., biscuit/cookie, flat/apartment). It is difficult to define a dataset that contains pairs of these words because there are many implicit biases that we would have to make to generate a dataset like this one, for example, “which is the equivalent of the word jumper in American?” that might have multiple answers.

Newspaper Data

We thus decided to collect some words of this kind and show which are their respective embeddings in the other space; while this is not a quantitative experiment, it should give the reader the idea that the model is stable enough to be general and to map words that have the same meaning in the same space. As stated above, for each word

1 https://dandelion.eu/

we show as an example we also show the neighborhood of that word in the mapped space; we do this last step to show that the matching words are not found in the neighborhood (i.e., "flat" is not close to "apartment" in the NYT space).

Mapping	Word	Top-5
GUA → NYT	flat	'**apartment**', 'walkup', 'flat', 'upstairs', 'oneroom'
NYT → NYT	flat	'sliding', 'padding', 'rough', 'seams', 'oneroom'
GUA → NYT	petrol	'**gasoline**', 'idling', 'trucks', 'diesel', 'suv'
NYT → NYT	petrol	'lactic', 'ricocheting', 'nanoparticles', 'quill', 'squish'
GUA → NYT	garbage	'bins', 'garbage', 'litter', '**rubbish**', 'bags'
NYT → NYT	garbage	'trash', 'cans', 'bins', 'piles', 'bags'
NYT → GUA	candy	'**sweets**', 'chocolate', 'sip', 'crisps', 'jelly'
GUA → GUA	candy	'spears', 'heavenly', 'bud', 'manger', 'jasmine'
NYT → GUA	gasoline	'**petrol**', 'diesel', 'fumes', 'batteries', 'plugin'
GUA → GUA	gasoline	'pellets', 'dispose', 'tubes', 'microfibres', 'landfilled'
GUA → NYT	biscuits	'**cookies**', 'chocolate', 'pancakes', 'bread', 'noodles'
NYT → NYT	biscuits	'honey', 'vanilla', 'spinach', 'cinnamon', 'coconut'

Table 24: Qualitative examples of mapping between NYT and GUA.

The interesting result here is that we seem to be able to map well words from American English to British English and vice-versa (a more structured experiment that proved this was run in Chapter 4). It is interesting to see how the word "flat" in the NYT space has a general meaning of "something related to flat surfaces" and that when we take the word "flat" from the guardian embedding, we get as first neighbor the word "apartment".

Reddit Corpus

Reddit is an online forum divided in *boards*, main topics in which users can post related information. For example, the "TwoXChromosomes" describes itself as "a subreddit for both serious and silly content, and intended for women's perspectives.". Instead, the "sports" board is mainly used to share information about sports.

We use Reddit data[2] that was also used in a recent paper to drive domain-specific sentiment lexicons for different boards [61]. This corpus was generated by extracting data using APIs contains 1.65 billion comments (of which 350,000 are not available as reported in the online page). The comments inside this dataset were produced from October of 2007 until May of 2015.

2 https://archive.org/details/2015_reddit_comments_corpus

We select some interesting boards and we compare mappings between some words.

Boards: TwoXChromosomes - Sports

In this subsection, we show some examples related to the TwoXChromosomes board (TWX) and the Sports (SPO) board. These two boards use the word "period" in different contexts: in TWX this word is frequently used to indicate the female monthly cycle. While in the SPO board this word is generally used to refer to a period of time during sports matches. We train CADE over these two slices and their concatenation as a compass and we show results in Table 25; as expected CADE can map the respective meaning in the correct positions.

Mapping	Word	Top-5
TWX → TWX	period	'periods', 'spotting', 'cycle', 'bleeding', 'flow'
SPO → SPO	period	'periods', 'duration', 'stoppage', 'half', 'longest'
TWX → SPO	period	'samples', '**blood**', 'sprain', 'stitches', 'bruising'
SPO → TWX	period	'**lifetime**', 'span', 'time', 'continuation', 'remainder'

Table 25: Qualitative examples of mapping between TWX and SPO.

Boards: Science - Pokemon

Another example we run was on the Science (SCI) board and the Pokemon (POK) board. Pokemon have their own "mythology" and several meanings have a strong and different connotation from standard English. We used CADE to map the two spaces, results are visible in Table 26. The word "move" is generally used to refer to a Pokemon attack (also referred to as technical move (tm)). The word "ash" in the Pokemon world is heavily influenced by the fact that the protagonist of the Pokemon tv-series is named "Ash". From the SCI space, we can import the representation of the meaning of the word "ash" that is more related to a natural effect.

Another interesting example is shown in the Table: "arceus" is a Pokemon and in the POK space is close to other Pokemons. As soon as we move its vector to the SCI space we get in the vicinity of terms like "gods" and "creator". This happens because in Pokemon mythology the Pokemon Arceus is generally referred to as the creator of the Pokemon. CADE can map these semantic differences in language offering an interesting view on how to explain terms.

Our framework shows two qualities here: it can inject words from one corpus to another even when meanings are skewed (as in example of "ash") and it can also provide a method to give explanations of word embeddings as in the case of "arceus": the use of the trans-

Mapping	Word	Top-5
SCI → POK	move	’**walk**’, ’go’, ’hop’, ‘sit’, ‘rotate’
POK → POK	move	‘tm’, ‘moves’, ‘moven’, ‘superpower’, ’attacks’
POK → SCI	arceus	’**gods**’, ’sakes’, ’worship’, ‘creator’, ‘god’
POK → POK	arceus	‘mew’, ‘deoxys’, ‘giratina’, ‘celebi’, ’regigigas’
SCI → POK	ash	‘lava’, ‘moisture’, ‘air’, ‘ocean’, ‘clouds’
POK → POK	ash	‘brock’, ‘misty’, ‘serena’, ‘giovanni’, ‘gary’

Table 26: Qualitative examples of mapping between the SCI and POK.

position to the SCI space allowed us to explain the meaning of the word “arceus” with the neighbors. Something that is not possible by looking at the neighbors of the word “arceus” in POK.

7.1.2 *Comparative Temporal Transposition*

We can also use TEE to explore temporal corpora. We use CADE on the collection of temporal corpus introduced by [145]. We annotated it with the use of the entity linker Dandelion. We annotated New York Times articles from the 2000, 2005 and 2010. We will refer to this corpora as NYT-E-2000, NYT-E-2005 and NYT-E-2010. We first generate the aligned embeddings using CADE and then create Typed Entity Embeddings adding 2016 types to the entities (note that this brings a strong bias in the model, but we ignore this to show possible applications of the tool. In general, types do not change as fast as relationships (e.g., past politicians are still politicians)). We thus use Typed Entity Embeddings to represent the entities and we only consider the similarity between entities.

Mapping	entity	Top-1
NYT-E-2010 → NYT-E-2005	dbr:Barack_Obama	dbr:George_W._Bush
NYT-E-2010 → NYT-E-2000	dbr:Barack_Obama	dbr:Bill_Clinton
NYT-E-2010 → NYT-E-2005	dbr:IPhone	dbr:IPod
NYT-E-2000 → NYT-E-2010	dbr:Yugoslavia	dbr:Iraq
NYT-E-2000 → NYT-E-2010	dbr:Yahoo!	dbr:Google

Table 27: Qualitative examples of mapping between the NYT-E-2000,2005 and 2010

An interesting property of the combination of TEE and CADE is that it offers the possibility of exploring space neighborhoods by taking into consideration the types of the entities. The first example in 27 shows that moving in the space allows us to directly find president George Bush from Barack Obama. The key added value here is given by disambiguation: in Chapter 4 we experimented with temporal analogies and those analogies contain the word "bush" that is ambiguous because it can refer to two different US Presidents and also to the plant. Moreover, the example that associates Yugoslavia with Iraq is particularly interesting because it puts together two countries that in the respective years were into a civil war (note that Yugoslavia ceased to exist after that civil war). As highlighted in Chapter 5 the use of the types allows us to explore the space by considering type constraints and finding really similar entities.

Change in Similarity

Another example in which distributional representations are useful is the comparison between different vectors over time. We show some example of similarity comparison in Figure 27.

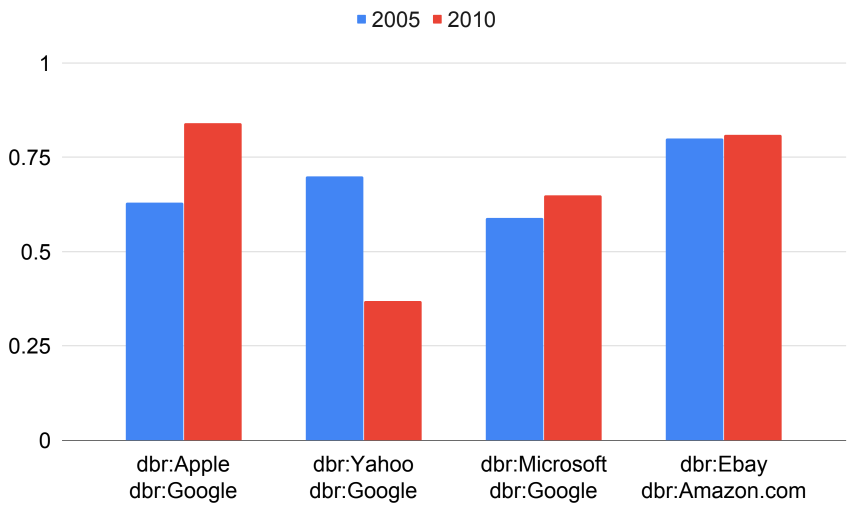


Figure 27: Similarities of different entities over different years.

See how the similarity between the entity `dbr:Google` and entity `dbr:Apple` increased over time, while the similarities between `dbr:Yahoo!` and `dbr:Google` decreased (because Google became a wider company, not only related to the search engine). It is also interesting to compare this with the results in Table 27 in which we were able to detect that the `dbr:Yahoo!` of the 2010 is `dbr:Google`.

7.2 COMPARATIVE REASONING ON CADE

A last example we want to show in this chapter is related to the combination of multiple vector spaces with reasoning systems. As described in the Preliminaries (Chapter 2), distributional representations do not support compositionality and do not have reasoning

Table 28: Learn to reason on one space, test on another space.

Axiom	Guardian Truth Value	Times Truth Value
$\mathsf{president}(\mathsf{dbr}:\mathsf{Donald\_Trump}, \mathsf{dbr}:\mathsf{Italy})$	0	0
$\mathsf{president}(\mathsf{dbr}:\mathsf{Donald\_Trump}, \mathsf{dbr}:\mathsf{United\_States})$	0.97	0.95

capabilities; while they allow us to evaluate the similarity between linguistic items. One key task for artificial intelligence is to find ways to effectively combining sub-symbolic perception with high-level reasoning [85]. This is what the neuro-symbolic [47, 48] learning and reasoning field aims to do: combining properties of symbolic reasoning and neural networks, to account both for data-driven learning and high-level reasoning, two tightly related aspects of human cognition.

7.2.1 *Transfer Reasoning and Comparative Reasoning*

Since with CADE we generate distributional aligned vector spaces, we can learn to reason on one vector space using LTNs and then used the learned neural network to reason over the other vector space. We annotated both the GUA and NYT text obtaining TEEs, we refer to this corpus as GUA-EE and NYT-EE.

TRANSFER REASONING For example, we can learn two predicates on the GUA-EE representation and see the truth value of the same predicates on the NYT-EE representation. In Table 28 we show an example where we learned the axioms $\mathsf{president}(\mathsf{dbr}:\mathsf{Donald\_Trump}, \mathsf{dbr}:\mathsf{Italy})$ and $\mathsf{president}(\mathsf{dbr}:\mathsf{Donald\_Trump}, \mathsf{dbr}:\mathsf{United\_States})$ on GUA-EE and see the truth value on NYT-EE.

The strong assumption behind this approach at modeling comparative reasoning is that the vectors of the entities in the two models are similar. Obviously, if the vectors in the two representations are identical the truth values will be the same. We evaluated this approach by considering the 100 entity names that have the most similar vectors in GUA-EE and NYT-EE by euclidean distance[3]. We then generate random predicates as in [16] and ask the model to learn, for example the predicate $\mathsf{predicate}_1(\mathsf{dbr}:\mathsf{Donald\_Trump})$ and $\mathsf{predicate}_2(\mathsf{dbr}:\mathsf{Donald\_Trump})$ over the GUA-EE embeddings and then we see if the same predicates get the same result on the NYT-EE vector space. All the predicates, once trained and transferred to the other space, give as output the same truth value once rounded to the nearest integer (0 or 1) (as in Table 28). This is again a result that comes from the fact that for similar vectors the prediction of LTNs will be the same.

3 Cosine similarity ignores the norm of the vectors that are important when the values are passed to the neural networks.

Note that this approach can be applied in the context of temporal data: we can train LTNs over multiple aligned spaces and then ask the trained model question about other aliened spaces. We consider again the aligned corpus NYT-E-2000, NYT-E-2005 and NYT-E-2010. To make an example, we trained LTNs with a predicate $\mathtt{politician}$ that puts a relation between a person, that is a politician and it's respective country.

- $\mathtt{politician}(\mathtt{dbr}:\mathtt{Barack\_Obama\_2010}, \mathtt{dbr}:\mathtt{USA\_2010})$
- $\neg\mathtt{politician}(\mathtt{dbr}:\mathtt{Barack\_Obama\_2010}, \mathtt{dbr}:\mathtt{Mexico\_2010})$
- $\mathtt{politician}(\mathtt{dbr}:\mathtt{George\_W.\_Bush\_2005}, \mathtt{dbr}:\mathtt{USA\_2005})$
- $\neg\mathtt{politician}(\mathtt{dbr}:\mathtt{George\_W.\_Bush\_2005}, \mathtt{dbr}:\mathtt{Mexico\_2005})$
- $\mathtt{politician}(\mathtt{dbr}:\mathtt{Dick\_Cheney\_2005}, \mathtt{dbr}:\mathtt{USA\_2005})$
- $\neg\mathtt{politician}(\mathtt{dbr}:\mathtt{Dick\_Cheney\_2005}, \mathtt{dbr}:\mathtt{Mexico\_2005})$
- $\mathtt{politician}(\mathtt{dbr}:\mathtt{Joe\_Biden\_2010}, \mathtt{dbr}:\mathtt{USA\_2010})$
- $\neg\mathtt{politician}(\mathtt{dbr}:\mathtt{Joe\_Biden,\_2010}, \mathtt{dbr}:\mathtt{Mexico\_2010})$

Once we learn an LTNs over this predicates we made logical queries as $\mathtt{politician}(\mathtt{dbr} : \mathtt{Bill\_Clinton,\_2000}, \mathtt{dbr} : \mathtt{Mexico\_2000})$ and $\mathtt{politician}(\mathtt{dbr} : \mathtt{Al\_Gore,\_2000}, \mathtt{dbr} : \mathtt{USA\_2000})$, obtaining the expected values.

Note a really important aspect: we can learn with LTNs on one single space or on multiple spaces thanks to the fact that the spaces in our comparative framework are aligned. This gives us great flexibility when we handle reasoning over the vector space. Even if these results are preliminary and are based on a small set of data (on which it is easy to overfit on), they are also promising.

TRUTH-VALUE DECREASE OVER TIME A last example we conducted shows the comparative analysis of the truth value of the predicates along time. We expect some predicates to become false over time. As above we consider an example with US politicians: we define a unary predicate $\mathtt{in\_charge}$ that we use to identify people governing the US during the years 2000, 2005, 2010. As before, our knowledge base contains some predicates as $\mathtt{charge}(\mathtt{dbr}:\mathtt{Barack\_Obama\_2010})$, $\mathtt{charge}(\mathtt{dbr}:\mathtt{Joe\_Biden,\_2010})$, $\mathtt{charge}(\mathtt{dbr}:\mathtt{Joe\_Biden,\_2005})$. When tested on the unseen (during LTNs training) entities $\mathtt{dbr}:\mathtt{Bill\_Clinton}$ and $\mathtt{dbr}:\mathtt{Al\_Gore}$, the value of the $\mathtt{charge}$ predicates (computed with the embedding of the entities over the years 2000, 2005 and 2010) decreases over time as shown in Figure 28.

This result is probably due to the fact that the typed entities $\mathtt{Bill\_Clinton}$ and $\mathtt{dbr}:\mathtt{Al\_Gore}$ appear more rarely in contexts in which entities related to the fact of running the government are present.

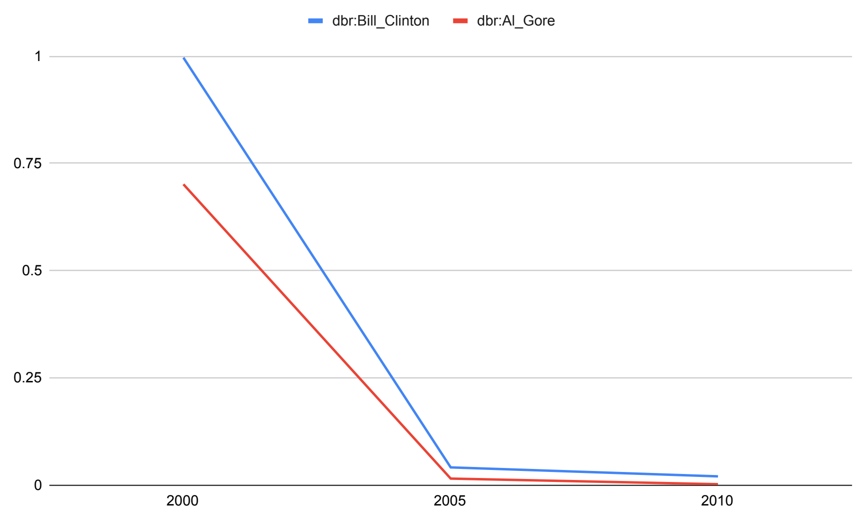


Figure 28: Example of decreasing in the truth value of the predicate in_charge over the years.

Again, these experiments, while surely preliminary, show that the combinations between LTNs, CADE and TEE allow us to analyze and compare distributional spaces with a high level of explainability given by the opportunity of using the axiom.

7.3 SUMMARY OF THIS CHAPTER

In this Chapter, we have shown how to combine all the different elements that have been introduced in this thesis, giving hints on how and when this comparison can be useful. We want to underline that there are two levels of comparison: the first one is given by the comparative framework of which CADE and TEE are the principal components while the second one inclueds logical reasoning over distributional data. With these two components, it is possible to do a comparative analysis between different corpora and evaluating the change in the similarity of different entities. On the other hand, introducing LTNs on top of this framework allowed us to provide a more structured inference over different corpora.

8 CONCLUSIONS

Natural language is one of the most important media of communication and researchers over the years have introduced models to account for the meaning of natural language expressions. These methods consider a context-based perspective on language, generating distributional representations that and have been found to be correlated with cognitive aspects of knowledge [77]. This perspective brought researchers to realize that different contexts would give us different representations (as in the temporal word embeddings domain introduced in Chapter 3), and thus it becomes important to define a general framework for comparison of these representations.

In this thesis, we have proposed a model to perform corpus-based comparison of distributional models of language and knowledge graphs. We started this research work by describing two different paradigms that accounted for the definition of meaning, logic, and distributional semantics. We discussed the main characteristics and limits of the two and we focused on the properties of distributional semantics. Distributional semantic is able to capture intrinsic characteristics of language and it has also been widely supported by researchers in the cognitive science field.

Since distributional semantics captures properties of text, we developed a method, CADE, that allows us to generate comparable representations of text that come from different sources. This allows us to study how language evolves over time and to analyze which are the words that shift in meaning when different contexts are considered.

We also treated the comparison of entity names introducing a distributional method for knowledge graph embeddings, TEE. This model is based on a purely distributional assumption and thus automatically inherits properties found by cognitive scientists in these presentations [27, 77]. These representations can be used with the CADE, allowing us to manage ambiguous natural language expressions in an effective way without losing the main linguistics and cognitive assumptions that drove the study of distributional semantics.

Finally, thanks to the introduction of entity names in the representations, we were able to combine those with a neuro-symbolic system, namely LTNs, that allows us to do more structured comparisons based on logical axioms.

In Chapter 7, we have shown some interesting applications that derive from the use of this framework. Our framework allows different levels of comparison that range from simple similarity computation of words in different slices (as in the example in which we computed the change in similarity between the entity `dbr:Google` with respect to other entities) to finding a semantic similar token in a specific slice given another token from another slice (as in the example about the word "flat"). Since we also provide a method for distributional representation, we can do operations on both entities and words. Finally, we have also shown that we can actually do comparative analysis using logical axioms by combining TEE with LTNs.

FUTURE RESEARCH DIRECTIONS This research opens up different research directions. First of all, the comparative framework that allows us to align representations can be exploited in several different ways, some of which have been exposed in Chapter 7. Possible future directions are:

- The framework is able to inject representations of words into novel contexts where they are missing. This might be interesting to treat tasks like few-shot and zero-shot learning;
- Other interesting applications of the framework are related to cognitive sciences: the framework we have introduced might help researchers study the biases that are present in different representations;
- As other work in state of the art, our method can be used to evaluate how language mutates over time and which are the reflections of this evolution in the language that people use. We can accomplish this goal with our framework by considering an analysis at an entity level and also at a logic level;
- The addition of the logical layer over the comparable representation might allow us to compare different reasoning systems in the future. This would be of great impact to the community because it will give us the ability to interpret the distributional representation under the point of view of a more structured framework built upon logic.

A NEURAL NETWORKS

Neural Networks are one of the most popular learning methods employed in Artificial Intelligence and Machine Learning. Warren McCulloch and Walter Pitts introduce them in 1943 [86], inspired by the biological nervous system: the "network" consist of a set of artificial neurons connected by weighted links, in the same way, biological neurons are connected by synapses. Research in the field stagnated at the beginning, mainly because of the lack of computer processing power capable of handling large neural networks. In the '80s, interest in the field started to grow back again: The work of Geoff Hinton, David Rumelhart, and Ronald Williams presented a way to train neural networks with many hidden layers [110] and it was proved that neural networks are capable of learning different forms of mathematical functions [33]. The core concepts defined during this period are behind the today widely used set of machine learning techniques known as *deep learning* [54, 75]. We hereby give a short introduction to the main neural network and machine learning concepts. For a more in depth introduction we refer to the deep learning book [54]. Note that we will generally make reference to a specific kind of machine learning area that is supervised learning, in which models learn to predict outputs from inputs. Other areas of machine learning are described in textbooks [54].

A.1 MACHINE LEARNING CONCEPTS

A general setting for a machine learning problem comprehends a set of training samples from which an algorithm has to *learn*. To introduce machine learning we describe an high-level general problem that is related to supervised learning.

The *training examples* consists of n pairs of inputs and target outputs (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n)[1]. An example could be a set of images (that could be represented with vectors $\mathbf{x}_i$ and a value in $\{0, 1\}$ that specify if the image contains a cat or not.

These examples are generated from a function f and this function is often unknown; this is why the goal of the network is to learn a function $\hat{f}_\theta$ where θ are the parameters that best approximates f.

Key Insight. Starting from a series of inputs-outputs examples a machine learning model has to learn a function that maps inputs to outputs.

1 Note that in general inputs and outputs can be both scalar or vectors

A.1.1 *Machine Learning Tasks*

The learning method can be either *supervised* or *unsupervised*[2]. A model learns using supervision if the desired output is provided for some given inputs. The two major challenges of this approach are: the difficulty of retrieving enough quantity of training data and the risk of *overfitting*, i.e. learning a function biased toward a particular training set.

Differently from the method specified above, machine learning models can also learn in an unsupervised way in which no target outputs are provided. Clustering algorithms are an example of an unsupervised model.

A.1.2 *Loss Function*

We define loss functions to compute the distance between the predicated value estimated by the neural networks and the target values. Different loss functions $\mathcal{L}$ can be defined[3], an example of loss function can be the mean squared error.

The learning problem of can be written in this general form:

$$\text{argmin}_\theta \frac{1}{n} \sum_{x_i, y_i}^{n} \mathcal{L}(\hat{f}_\theta(x_i), y_i) \tag{7}$$

To learn the best possible mapping between the input data and the output data we want to minimize the loss function. That is, we want to find the best configuration of the parameters „ that minimize the loss function $\mathcal{L}$.

Key Insight. A machine learning problem is defined with respect to a loss function. The objective is to minimize this loss function in such a way that the models predictions are as near as possible to the one provided by the function f.

A.2 NEURAL NETWORKS

A.2.1 *Deep Neural Networks*

Deep neural networks are generally explained considering neurons and weighted connections between them that makes them assume the form of actual networks (that is why we generally refer to this model as neural network).

2 Consider that there is also another field in machine learning called reinforcement learning, but that is subtly different from the two mentioned here

3 Note that for learning reasons these functions must be differentiable.

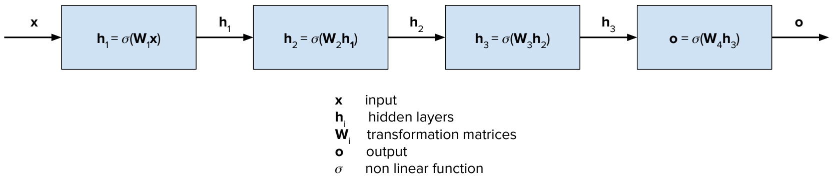


Figure 29: Example of neural network chaining

Nevertheless, the most direct way to view neural networks is as a model that does function composition through a series of matrix transformation from an input space to an output space. In fact, each layer in the neural network can be viewed as a matrix multiplication that transforms the input. A general layer assumes the form of an operation $\mathbf{h} = \sigma(\mathbf{Wx})$. Where σ denotes an activation function function, x denotes the input of the layer; the $\mathbf{W}$ matrix components are often referred to as weights; these weights are part of the parameters „ that the network has to adjust to minimize the loss function.

In Figure 29 we show how the neural network computes the output. Starting from an input $\mathbf{x}$ a first matrix multiplication is applied to the input $\mathbf{W}_1\mathbf{x}$, and then a non-linearity function is applied[4], generating the values of what is generally called the first hidden layer and that will be the input for the next layer. This procedure is repeated until the last layer, where the network generates the final output. The final output is thus the result of $\hat{f}_\theta(x_i)$ that will be used to compute the value of the loss function.

Adding more layers (i.e., matrix multiplications) makes the network deeper, hence the name deep learning that has been given to the field.

Key Insight. Neural networks generate predictions through a series of matrix multiplications that are followed by non-linear function applications.

A.2.2 *Activation functions*

Non-linearities are also called activation functions. This definition comes from the earliest architecture of artificial neural networks, composed of a single computational neuron, where the activation functions was used to decide decide which connection of the neuron should be active and which should not be active. As outlined above, this functions are also useful between layers to make neural networks

4 Non-linearities are added because the composition of linear functions is a linear function and hence a network with many layers would be reduced to a network with only one hidden layer.

able to learn non-linear functions. We hereby show some of the activation functions present in literature.

SIGMOID $\sigma(x) = \frac{1}{1+e^{-x}}$, range $[0, 1]$

HYPERBOLIC TANGENT $tanh(x) = \frac{e^{z}-e^{-x}}{e^{z}+e^{-x}}$, range $[-1, 1]$

RECTIFIED LINEAR UNIT $ReLU(x) = max(0, x)$, range $[0, \infty]$

SOFTMAX $\sigma(\mathbf{x})_j = \frac{e^{x}_{j}}{\sum_{i}^{n} e^{x}_{k}}$, range $[0, 1]$

Each activation function as specific uses, for example the sigmoid can be used in context of binary classification. The softmax function is a generalization of the sigmoid function to the multi-dimensional case: it takes in input a vector and transform it in a vector of pseudo-probabilities: each value is normalized in the interval $[0, 1]$ and the sum of the values is 1.

A.2.3 *Backpropagation and Optimization*

The loss function can be defined with respect to all the errors generated over the entire training set or with respect to a batch of training examples. Optimization is done by computing the gradient of the loss function and updating the parameters to minimize the loss function [54].

The method to learn from a loss function in deep learning is the *backpropagation* [54]. The loss function can be seen as a function of weights, the *backpropagation* computes the gradient of the loss and updates the values of the weights favoring the minimizaion of the function. This process is repeated for each input example, a common technique to speed up the computation and perform more general *backpropagation* is to apply the *backpropagation* as the mean of gradients computed on a batch of examples with size r. This prevents that outliers examples will affect the generalization process. Note also that there are different ways to actually updates the weights of a neural network, these are often called optimizers.

A.2.4 *Overfitting*

A very important problem that has to be faced during a neural network training (and even using other machine learning models) is the problem *overfitting* [54]. This is a phenomenon which causes bad performances of the model even if, during the training, performances seems to be good. The overfitting appears when the model *memorizes* the training distribution instead of *generalizing*, so is not able to recognize any new input which shows a similar distribution. There are different methods to tackle and to reduce the effects of overfitting in neural networks: for example, regularization techniques help in constraining the value the weights of the network assume. On the other hand, techniques like the dropout (i.e., randomly deactivating some weights in the network at each iteration) have been found effective in neural networks [54].

BIBLIOGRAPHY

[1] Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. "Domain adaption of named entity recognition to support credit risk assessment." In: *Proceedings of the Australasian Language Technology Association Workshop 2015*. 2015, pp. 84–90.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. "Dbpedia: A nucleus for a web of open data." In: *The semantic web*. Springer, 2007, pp. 722–735.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[4] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. "Hinge-Loss Markov Random Fields and Probabilistic Soft Logic." In: *Journal of Machine Learning Research* 18 (2017), pp. 1–67.

[5] Robert Bamler and Stephan Mandt. "Dynamic Word Embeddings." In: *ICML*. 2018, pp. 380–389.

[6] David Bamman, Chris Dyer, and Noah A Smith. "Distributed Representations of Geographically Situated Language." In: *ACL*. Vol. 2. 2014, pp. 828–834.

[7] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors." In: *ACL (1)*. 2014, pp. 238–247.

[8] Roberto Camacho Barranco, Raimundo F Dos Santos, and M Shahriar Hossain. "Tracking the Evolution of Words with Time-reflective Text Representations." In: *arXiv preprint arXiv:1807.04441* (2018).

[9] Pierpaolo Basile, Annalina Caputo, Roberta Luisi, and Giovanni Semeraro. "Diachronic Analysis of the Italian Language exploiting Google Ngram." In: *CLiC-it*. 2016.

[10] Pierpaolo Basile, Annalina Caputo, Gaetano Rossiello, and Giovanni Semeraro. "Learning to rank entity relatedness through embedding-based features." In: *NLDB*. Springer. 2016, pp. 471–477.

[11] Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. "Representing Meaning with a Combination of Logical and Distributional Models." In: *Computational Linguistics* 42.4 (2016), pp. 763–808.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. "A Neural Probabilistic Language Model." In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[13] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, Jaz Kandola, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor. "A Neural Probabilistic Language Model." In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[14] Tim Berners-Lee, James Hendler, and Lassila. "The semantic web." In: *Scientific american* 284.5 (2001), pp. 28–37.

[15] Sudeep Bhatia. "The semantic representation of prejudice and stereotypes." In: *Cognition* 164 (2017), pp. 46–60.

[16] Federico Bianchi and Pascal Hitzler. "On the Capabilities of Logic Tensor Networks for Deductive Reasoning." In: *AAAI Spring Symposium MAKE*. 2019.

[17] Federico Bianchi, Matteo Palmonari, and Debora Nozza. "Towards Encoding Time in Text-Based Entity Embeddings." In: *International Semantic Web Conference*. 2018.

[18] Federico Bianchi, Matteo Palmonari, Marco Cremaschi, and Elisabetta Fersini. "Actively Learning to Rank Semantic Associations for Personalized Contextual Exploration of Knowledge Graphs." In: *Extended Semantic Web Conference*. Springer. 2017, pp. 120–135.

[19] Federico Bianchi, Mauricio Soto, Matteo Palmonari, and Vincenzo Cutrona. "Type Vector Representations from Text: An Empirical Analysis." In: *Deep Learning for Knowledge Graphs and Semantic Technologies Workshop, co-located with the Extended Semantic Web Conference*. 2018.

[20] Federico Bianchi, Matteo Palmonari, Pascal Hitzler, and Luciano Serafini. "Complementing Logical Reasoning with Sub-Symbolic Commonsense." In: *International Joint Conference on Rules and Reasoning*. 2019.

[21] Gemma Boleda. "Distributional Semantics and Linguistic Theory." In: *Annu. Rev. Ling.* (2020).

[22] Gemma Boleda and Aurélie Herbelot. "Formal distributional semantics: Introduction to the special issue." In: *Computational Linguistics* 42.4 (2016), pp. 619–635.

[23] Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. "Man is to Computer Programmer As Woman is to Homemaker? Debiasing Word Embeddings." In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. USA: Curran Associates Inc., 2016, pp. 4356–4364. ISBN: 978-1-5108-3881-9.

[24] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. "Learning Structured Embeddings of Knowledge Bases." In: *AAAI*. AAAI Press, 2011.

[25] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating embeddings for modeling multi-relational data." In: *Advances in neural information processing systems*. 2013, pp. 2787–2795.

[26] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. "A large annotated corpus for learning natural language inference." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 632–642.

[27] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. "Semantics derived automatically from language corpora contain human-like biases." In: *Science* 356.6334 (2017), pp. 183–186.

[28] K W Church. "Emerging Trends: Word2Vec." In: *Natural Language Engineering* (2017). ISSN: 14698110.

[29] Kenneth Ward Church and Patrick Hanks. "Word association norms, mutual information, and lexicography." In: *Proceedings of the 27th annual meeting on Association for Computational Linguistics -*. 1989. ISBN: 0891-2017.

[30] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. "Natural Language Processing (Almost) from Scratch." In: *Journal of Machine Learning Research* 12 (2011), pp. 2493–2537.

[31] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. "Word Translation Without Parallel Data." In: *arXiv preprint arXiv:1710.04087* (2017).

[32] D Alan Cruse. "Antonymy revisited: Some thoughts on the relationship between words and concepts." In: *Frames, Fields, and Contrasts, Hillsdale, NJ, Lawrence Erlbaum associates* (1992), pp. 289–306.

[33] G. Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 0932-4194.

[34] Luc De Raedt and Angelika Kimmig. "Probabilistic (logic) programming concepts." In: *Machine Learning* 100.1 (2015), pp. 5–47.

[35] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. "Indexing by latent semantic analysis." In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[37] Valerio Di Carlo, Federico Bianchi, and Matteo Palmonari. "Training Temporal Word Embeddings with a Compass." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[38] Ivan Donadello, Luciano Serafini, and Artur D'Avila Garcez. "Logic tensor networks for semantic image interpretation." In: *IJCAI*. 2017.

[39] Lisa Ehrlinger and Wolfram Wöß. "Towards a Definition of Knowledge Graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48 (2016).

[40] Vyvyan Evans. "Lexical concepts, cognitive models and meaning-construction." In: *Cognitive Linguistics* 17.4 (2006), pp. 491–534.

[41] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. "Entity Disambiguation by Knowledge and Text Jointly Embedding." In: *CoNLL*. 2016, pp. 260–269.

[42] Manaal Faruqui and Chris Dyer. "Improving vector space word representations using multilingual correlation." In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014, pp. 462–471.

[43] Jose Ferreiros. "The Road to Modern Logic-An Interpretation." In: *The Bulletin of Symbolic Logic* 7.4 (2001), pp. 441–484. ISSN: 10798986.

[44] Lorenzo Ferrone and Fabio Massimo Zanzotto. "Symbolic, distributed and distributional representations for natural language processing in the era of deep learning: a survey." In: *arXiv preprint arXiv:1702.00764* (2017).

[45] John R Firth. "A synopsis of linguistic theory, 1930-1955." In: *Studies in linguistic analysis* (1957).

[46] John R. Firth. *Papers in Linguistics*. London: Oxford University Press., 1957.

[47] Artur d'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

[48] Artur d'Avila Garcez, Marco Gori, Luis C Lamb, Luciano Serafini, Michael Spranger, and Son N Tran. "Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning." In: *arXiv preprint arXiv:1905.06088* (2019).

[49] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. "Word embeddings quantify 100 years of gender and ethnic stereotypes." In: *Proceedings of the National Academy of Sciences* 115.16 (2018), E3635–E3644.

[50] Dan Garrette, Katrin Erk, and Raymond Mooney. "A formal approach to linking logical form and vector-space lexical semantics." In: *Computing meaning*. Springer, 2014, pp. 27–48.

[51] Pierdaniele Giaretta and Nicola Guarino. "Ontologies and knowledge bases towards a terminological clarification." In: *Towards very large knowledge bases: knowledge building & knowledge sharing* 25.32 (1995), pp. 307–317.

[52] Nabeel Gillani and Roger Levy. "Simple dynamic word embeddings for mapping perceptions in the public sphere." In: *arXiv:1904.03352 [cs]* (Apr. 2019). arXiv: 1904.03352. (Visited on 05/14/2019).

[53] Asunción Gómez-Pérez and Richard Benjamins. "Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods." In: IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings. 1999.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[55] Sergio Greco and Cristian Molinaro. "Datalog and logic databases." In: *Synthesis Lectures on Data Management* 7.2 (2015), pp. 1–169.

[56] Anthony G Greenwald, Debbie E. McGhee, and Joe L. Schwartz. "Measuring individual differences in implicit cognition: the implicit association test." In: *Journal of personality and social psychology* 74 6 (1998), pp. 1464–80.

[57] Kristina Gulordava and Marco Baroni. "A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus." In: *GEMS Workshop*. Association for Computational Linguistics, 2011, pp. 67–71. ISBN: 978-1-937284-16-9.

[58] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. "Jointly embedding knowledge graphs and logical rules." In: *EMNLP*. 2016, pp. 192–202.

[59] Kilem Li Gwet. "Computing inter-rater reliability and its variance in the presence of high agreement." In: *British Journal of Mathematical and Statistical Psychology* 61.1 (2008), pp. 29–48.

[60] William L Hamilton, Jure Leskovec, and Dan Jurafsky. "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change." In: *ACL*. Vol. 1. 2016, pp. 1489–1501.

[61] William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. "Inducing domain-specific sentiment lexicons from unlabeled corpora." In: *EMNLP*. Vol. 2016. 2016, p. 595.

[62] Zellig Harris. "Distributional Structure." In: *The Philosophy of Linguistics*. New York: Oxford University Press, 1964.

[63] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1." In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. MIT Press, 1986. Chap. Distributed Representations, pp. 77–109.

[64] Pascal Hitzler, Federico Bianchi, Monireh Ebrahimi, and Md Kamruzzaman Sarker. "Neural-symbolic integration and the Semantic Web." In: *Semantic Web* (Oct. 2019), 1–9.

[65] Keith J Holyoak. "Analogy and relational reasoning." In: *The Oxford handbook of thinking and reasoning* (2012), pp. 234–259.

[66] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. "Sensembed: Learning sense embeddings for word and relational similarity." In: *ACL*. 2015, pp. 95–105.

[67] Shoaib Jameel and Steven Schockaert. "Entity Embeddings with Conceptual Subspaces as a Basis for Plausible Reasoning." In: *ECAI*. 2016.

[68] Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. "How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks." In: *arXiv preprint arXiv:1702.02170* (2017).

[69] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. "A latent factor model for highly multi-relational data." In: *NIPS*. 2012, pp. 3176–3184.

[70] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. "Knowledge Base Completion: Baselines Strike Back." In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 2017, pp. 69–74.

[71] Kate Kearns. *Semantics*. Palgrave, 2011, p. 269. ISBN: 9780230232303.

[72] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. "Temporal Analysis of Language through Neural Language Models." In: *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. 2014, pp. 61–65.

[73] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. "Statistically significant detection of linguistic change." In: *WWW*. 2015, pp. 625–635.

[74] Matt Le, Stephen Roller, Laetitia Papaxanthos, Douwe Kiela, and Maximilian Nickel. "Inferring Concept Hierarchies from Text Corpora via Hyperbolic Embeddings." In: *arXiv preprint arXiv:1902.00913* (2019).

[75] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836.

[76] Robert Leaman and Graciela Gonzalez. "BANNER: an executable survey of advances in biomedical named entity recognition." In: *Biocomputing*. World Scientific, 2008, pp. 652–663.

[77] Alessandro Lenci. "Distributional semantics in linguistic and cognitive research." In: *Italian journal of linguistics* 20.1 (2008), pp. 1–31.

[78] Alessandro Lenci. "Distributional models of word meaning." In: *Annual review of Linguistics* 4 (2018), pp. 151–171.

[79] Omer Levy and Yoav Goldberg. "Linguistic Regularities in Sparse and Explicit Word Representations." In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. 2014. ISBN: 9781941643020.

[80] Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. "Biomedical named entity recognition based on extended recurrent neural networks." In: *2015 IEEE International Conference on bioinformatics and biomedicine (BIBM)*. IEEE. 2015, pp. 649–652.

[81] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning Entity and Relation Embeddings for Knowledge Graph Completion." In: *AAAI*. 2015, pp. 2181–2187.

[82] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. "Topical Word Embeddings." In: *AAAI*. 2015, pp. 2418–2424.

[83] John Lyons. *Language and linguistics : an introduction*. Cambridge University Press, 1981, p. 356. ISBN: 9780521297752.

[84] Gengchen Mai, Krzysztof Janowicz, and Bo Yan. "Combining Text Embedding and Knowledge Graph Embedding Techniques for Academic Search Engines." In: *4th Workshop on Semantic Deep Learning (SemDeep-4)*. 2018.

[85] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. "DeepProbLog: Neural Probabilistic Logic Programming." In: *arXiv preprint arXiv:1805.10872* (2018).

[86] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 0007-4985.

[87] David McDonald and Shan He. "HEAT: Hyperbolic Embedding of Attributed Networks." In: *arXiv preprint arXiv:1903.03036* (2019).

[88] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. "DBpedia spotlight: shedding light on the web of documents." In: *Proceedings of the 7th international conference on semantic systems*. ACM. 2011, pp. 1–8.

[89] Albert Meroño-Peñuela, Ashkan Ashkpour, Marieke Van Erp, Kees Mandemakers, Leen Breure, Andrea Scharnhorst, Stefan Schlobach, and Frank Van Harmelen. "Semantic technologies for historical research: A survey." In: *Semantic Web* 6.6 (2015), pp. 539–564.

[90] Carolyn B Mervis and Maria A Crisafi. "Order of acquisition of subordinate-, basic-, and superordinate-level categories." In: *Child Development* (1982), pp. 258–266.

[91] Ivan Meza-Ruiz and Sebastian Riedel. "Jointly identifying predicates, arguments and senses using Markov logic." In: *NAACL*. ACL. 2009, pp. 155–163.

[92] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." In: *hlt-Naacl*. Vol. 13. 2013, pp. 746–751.

[93] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *NIPS*. 2013, pp. 3111–3119.

[94] George A. Miller and Walter G. Charles. "Contextual correlates of semantic similarity." In: *Language and Cognitive Processes* 6.1 (Jan. 1991), pp. 1–28. ISSN: 0169-0965.

[95] Marvin L Minsky. "Logical versus analogical or symbolic versus connectionist or neat versus scruffy." In: *AI magazine* 12.2 (1991), p. 34.

[96] Allen Newell and Herbert A. Simon. "Computer science as empirical inquiry: symbols and search." In: *Communications of the ACM* 19.3 (Mar. 1976), pp. 113–126.

[97] Maximilian Nickel and Douwe Kiela. "Poincaré Embeddings for Learning Hierarchical Representations." In: *NIPS*. 2017, pp. 6341–6350.

[98] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data." In: *Proceedings of ICML-11*. 2011, pp. 809–816.

[99] Bo Pang, Lillian Lee, et al. "Opinion mining and sentiment analysis." In: *Foundations and Trends® in Information Retrieval* 2.1–2 (2008), pp. 1–135.

[100] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[101] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." In: *NAACL*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. ACL, 2018, pp. 2227–2237.

[102] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[103] Paul Portner. *What is meaning? : fundamentals of formal semantics*. Malden, MA: Blackwell Pub, 2005.

[104] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. "RDF2Vec: RDF graph embeddings and their applications." In: *Semantic Web* Preprint (2018), pp. 1–32.

[105] Tim Rocktäschel and Sebastian Riedel. "End-to-end differentiable proving." In: *NIPS*. 2017, pp. 3788–3800.

[106] Xin Rong. "word2vec Parameter Learning Explained." In: Nov. 2014.

[107] Marco Rovera, Federico Nanni, Simone Paolo Ponzetto, and Anna Goy. "Domain-specific named entity disambiguation in historical memoirs." In: *CEUR Workshop Proceedings*. Vol. 2006. RWTH. 2017, Paper–20.

[108] Maja Rudolph and David Blei. "Dynamic Embeddings for Language Evolution." In: *WWW*. 2018, pp. 1003–1011.

[109] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. "Exponential family embeddings." In: *NIPS*. 2016, pp. 478–486.

[110] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 0028-0836.

[111] Frederic Sala, Christopher De Sa, Albert Gu, and Christopher Ré. "Representation Tradeoffs for Hyperbolic Embeddings." In: *ICML*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4457–4466.

[112] Evan Sandhaus. *The New York Times Annotated Corpus*. 2008.

[113] Adriaan MJ Schakel and Benjamin J Wilson. "Measuring word significance using distributed representations of words." In: *arXiv preprint arXiv:1508.02297* (2015).

[114] Luciano Serafini and Artur S d'Avila Garcez. "Learning and reasoning with logic tensor networks." In: *AI*IA*. Springer. 2016, pp. 334–348.

[115] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. "Offline bilingual word vectors, orthogonal transformations and the inverted softmax." In: *arXiv preprint arXiv:1702.03859* (2017).

[116] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. "Reasoning with neural tensor networks for knowledge base completion." In: *Advances in neural information processing systems*. 2013, pp. 926–934.

[117] Sparck Jones K. "A Statistical Interpretation of Term Specificity and its Application in Retrieval." In: *Journal of Documentation* (1972).

[118] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP." In: (July 2019), pp. 3645–3650.

[119] Ryota Suzuki, Ryusuke Takahama, and Shun Onoda. "Hyperbolic Disk Embeddings for Directed Acyclic Graphs." In: *arXiv preprint arXiv:1902.04335* (2019).

[120] Terrence Szymanski. "Temporal Word Analogies: Identifying Lexical Replacement with Diachronic Word Embeddings." In: *ACL*. Association for Computational Linguistics, 2017.

[121] Matt Taddy. "Document Classification by Inversion of Distributed Language Representations." In: *IJCNLP*. Vol. 2. 2015, pp. 45–49.

[122] Jean-Pierre Thibaut, Robert French, and Milena Vezneva. "Cognitive load and semantic analogies: Searching semantic space." In: *Psychonomic bulletin & review* 17.4 (2010), pp. 569–574.

[123] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. "Poincar\'e GloVe: Hyperbolic Word Embeddings." In: *arXiv preprint arXiv:1810.06546* (2018).

[124] Marco Del Tredici, Malvina Nissim, and Andrea Zaninello. "Tracing Metaphors in Time through Self-Distance in Vector Spaces." In: *CLiC-it*. 2016.

[125] Rocco Tripodi, Massimo Warglien, Simon Sullam, and Deborah Paci. "Tracing Antisemitic Language Through Diachronic Embedding Projections: France 1789-1914." In: Jan. 2019, pp. 115–125.

[126] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex embeddings for simple link prediction." In: *ICML*. 2016, pp. 2071–2080.

[127] Peter D. Turney and Patrick Pantel. "From frequency to meaning: Vector space models of semantics." In: *JAIR* (2010).

[128] Jean Van Heijenoort. *From Frege to Gödel: a source book in mathematical logic, 1879-1931*. Vol. 9. Harvard University Press, 1967.

[129] Manuel Vimercati, Federico Bianchi, Mauricio Soto, and Matteo Palmonari. "Mapping Lexical Knowledge to Distributed Representations for Ontology Concept Invention." In: *AIXIA (to appear)*. 2019.

[130] Stella Vosniadou. "Similarity and Analogical Reasoning." In: New York, NY, USA: Cambridge University Press, 1989. Chap. Analogical Reasoning As a Mechanism in Knowledge Acquisition: A Developmental Perspective, pp. 413–437. ISBN: 0-521-38935-6.

[131] Hao Wang. "The Axiomatization of Arithmetic." In: *J. Symbolic Logic* 22.2 (June 1957), pp. 145–158.

[132] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. "Knowledge graph embedding: A survey of approaches and applications." In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743.

[133] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge Graph and Text Jointly Embedding." In: *EMNLP*. Vol. 14. 2014, pp. 1591–1601.

[134] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge graph embedding by translating on hyperplanes." In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[135] Zhigang Wang and Juan-Zi Li. "Text-Enhanced Representation Learning for Knowledge Graph." In: *IJCAI*. 2016, pp. 1293–1299.

[136] Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. "Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1711–1721.

[137] Matthijs Westera and Gemma Boleda. "Don't Blame Distributional Semantics if it can't do Entailment." In: *arXiv preprint arXiv:1905.07356* (2019).

[138] *Wiki2Vec*. https://github.com/idio/wiki2vec. URL: `https://github.com/idio/wiki2vec`.

[139] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 2009.

[140] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. "SSP: semantic space projection for knowledge graph embedding with text descriptions." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[141] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. "Representation learning of knowledge graphs with entity descriptions." In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[142] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. "Normalized word embedding and orthogonal transform for bilingual word translation." In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1006–1011.

[143] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. "Wikipedia2Vec: An Optimized Tool for Learning Embeddings of Words and Entities from Wikipedia." In: *arXiv preprint 1812.06280* (2018).

[144] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases." In: *arXiv preprint arXiv:1412.6575* (2014).

[145] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. "Dynamic Word Embeddings for Evolving Semantic Discovery." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM. 2018, pp. 673–681.

[146] Shaodian Zhang and Noémie Elhadad. "Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts." In: *Journal of biomedical informatics* 46.6 (2013), pp. 1088–1098.

[147] Yating Zhang, Adam Jatowt, Sourav S. Bhowmick, and Katsumi Tanaka. "The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time." In: *IEEE Transactions on Knowledge and Data Engineering* 28.10 (Oct. 2016), pp. 2793–2807.

[148] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. "Learning Gender-Neutral Word Embeddings." In: *EMNLP*. 2018.

[149] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. "Learning continuous word embedding with metadata for question retrieval in community question answering." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 250–259.

[150] Hanqing Zhou, Amal Zouaq, and Diana Inkpen. "A Comparison of Word Embeddings and N-gram Models for DBpedia Type and Invalid Entity Detection." In: *Information* 10.1 (Dec. 2018), p. 6.

[151] Ganggao Zhu and Carlos A Iglesias. "Computing semantic similarity of concepts in knowledge graphs." In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2017), pp. 72–85.