Can We Replace Reads by Numeric Signatures? Lyndon Fingerprints as Representations of Sequencing Reads for Machine Learning*

Paola Bonizzoni[1], Clelia De Felice[3], Alessia Petescia[1], Yuri Pirola[1], Raffaella Rizzi[1], Jens Stoye[2], Rocco Zaccagnino[3], and Rosalba Zizza[3]

[1] Dip. di Informatica, Sistemistica e Comunicazione, University of Milano-Bicocca, viale Sarca 336, 20126 Milan, Italy
{paola.bonizzoni, yuri.pirola, raffaella.rizzi}@unimib.it, a.petescia@campus.unimib.it

[2] Faculty of Technology and Center for Biotechnology, Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany
jens.stoye@uni-bielefeld.de

[3] Dip. di Informatica, University of Salerno, via Giovanni Paolo II 132, 84084 Fisciano, Italy
{cdefelice, rzaccagnino, rzizza}@unisa.it

Abstract. Representations of biological sequences facilitating sequence comparison are crucial in several bioinformatics tasks. Recently, the Lyndon factorization has been proved to preserve common factors in overlapping reads [6], thus leading to the idea of using factorizations of sequences to define measures of similarity between reads. In this paper we propose as a signature of sequencing reads the notion of *fingerprint*, i.e., the sequence of lengths of consecutive factors in Lyndon-based factorizations of the reads. Surprisingly, fingerprints of reads are effective in preserving sequence similarities while providing a compact representation of the read, and so, k-mers extracted from a fingerprint, called k-fingers, can be used to capture sequence similarity between reads.

We first provide a probabilistic framework to estimate the behaviour of fingerprints. Then we experimentally evaluate the effectiveness of this representation for machine learning algorithms for classifying biological sequences. In particular, we considered the problem of assigning RNA-Seq reads to the most likely gene from which they were generated. Our results show that fingerprints can provide an effective machine learning interpretable representation, successfully preserving sequence similarity.

Keywords: Sequence analysis · Lyndon factorization · Read representation · Machine learning · Sequence mining

* The final authenticated version is available online at https://doi.org/10.1007/978-3-030-74432-8_2.

1 Introduction

In the Big Data era, characterized by a massive data growth, mining *sequence* data has attracted a lot of attention, since knowing useful patterns from sequences can benefit many applications, such as, event prediction, pattern discovery, time-aware recommendation, DNA detection, and feature embedding [12].

In the specific context of *biological sequences*, finding machine-interpretable representations for sequences that can increase performance of machine learning algorithms, is a challenging task. Indeed, even the most sophisticated algorithms would perform poorly with inappropriate features, while simple methods can potentially perform when fed with appropriate features. Most of the approaches proposed in literature adopt existing methods in Natural Language Processing with the goal to discover functions encoded within biological sequences [1,11,16]. As well as the common techniques in Bioinformatics to study sequences, also such methods involves fixed-length overlapping n-grams [18,20].

A main question addressed in this paper is whether there exists a "similarity signature" that may be used to represent a sequencing read and that can be easily detected while reading the read itself. We answer to this question by exploiting the *Lyndon factorizations* for a collection of reads. The Lyndon factorization is one of the most well-known factorizations in combinatorics on words: it is unique for a word and it can be computed in linear time [7,14]. The notion of Lyndon word is not novel in the field of Bioinformatics, since it was used to locate short motifs [8] and more recently it was explored in the development of bijective Burrows-Wheeler Tranforms [13]. Such a factorization has a main desired property: a read shares a set of consecutive common Lyndon factors with the Lyndon factorization of a superstring of the read itself [4,6]. Surprisingly, in this paper we discover that the length of factors in a Lyndon factorization is enough to define a notion of signature that captures sequence similarity: this is our notion of *fingerprint* of a read. Given a fingerprint f which is a sequence of integer numbers, we extract k-fingers, i.e. k-mers of f. Then collections of k-fingers are used as a main signature to analyze a sample of reads.

To show the effectiveness of such a novel representation approach, we explore their use in the framework of RNA-Seq data classification consisting in assigning each read in a collection to the origin gene. A similar problem for RNA-seq data analysis in trascriptomics, which is filtering reads by origin genes, has been recently considered in [9] where we refer for the main reference literature. Results of such a preliminary evaluation study, show that the fingerprint can be used successfully to provide a machine-interpretable representation and opens up new perspectives on the possibility of defining a new biological sequence embedding technique. This paper is organized as follows. In section 2, we present the factorizations used to deal with the double-stranded nature of sequencing reads. In section 3 we explore results motivating the use of k-fingers by extending probabilistic results for sequence similarity based on k-mers. Section 4 provides the details about the experiments and related methodology carried out to assess our approach. Finally, Section 5 discusses the results and future directions.

2 Lyndon Factorizations and Overlapping Reads

Let Σ be a finite alphabet and let $s = c_1 \cdots c_n$ be a sequence of n characters drawn from Σ; we say that s is a *string* over Σ of length n. The *length* and the character c_i (at position i) will be denoted by $|s|$ and $s[i]$, respectively. The *substring* of s from position i to position j will be denoted by $s[i:j]$. A *prefix* or *suffix* of s are substrings $s[1:j]$ and $s[i:|s|]$, respectively (denoted also by $s[:j]$ and $s[i:]$). A prefix or a suffix of s is *proper* if $j \neq |s|$ and $i \neq 1$. In the following, we will use the notation $s < v$ ($s > v$) to specify that string s is lexicographically smaller (greater) than string v, and the notation $s \ll v$ to specify that $s < v$ and additionally s is not a proper prefix of v.

Now we introduce the definitions of *factorization* and *fingerprint*, which are the main ingredients we use to capture the overlap between two reads. A *factorization* of a string s is a sequence $F(s) = \langle f_1, f_2, \ldots, f_n \rangle$ of factors, such that $s = f_1 f_2 \ldots f_n$. The *fingerprint* of s with respect to $F(s)$ is the sequence $\mathcal{L}(s) = \langle |f_1|, |f_2|, \ldots, |f_n| \rangle$ of the factor lengths. Given a fingerprint $\mathcal{L}(s) = \langle l_1, l_2, \ldots, l_n \rangle$, a *k-finger* is any subsequence $\langle l_i, l_{i+1}, \ldots, l_{i+k-1} \rangle$ of k consecutive lengths, that is, a k-mer of $\mathcal{L}(s)$. The substring $f_i f_{i+1} \ldots f_{i+k-1}$ will be called as *supporting string* of the k-finger.

In order to capture overlaps between reads, we use Lyndon factorizations, namely composed of *Lyndon Words* [15,3]. A string s is a *Lyndon word* if and only if it is strictly lexicographically smaller than any of its proper suffixes. For example, $s = aabbab$ over alphabet $\{a, b\}$ is a Lyndon word, whereas string $s' = abaabb$ is not a Lyndon word, since the suffix $aabb$ is smaller than s'. The Chen-Fox-Lyndon's Theorem [7] states that any nonempty string s has a unique standard Lyndon factorization $F(s) = \langle f_1, f_2, \ldots, f_n \rangle$ such that $f_1 \geq f_2 \geq \cdots \geq f_n$. Such a Lyndon factorization is called CFL from the authors' names. The Duval algorithm [10] allows to compute CFL in linear time and constant space.

A property of the CFL factorization [6], which is crucial in our framework, is the following.

Conservation Property: let s be a string such that $\mathsf{CFL}(s) = \langle f_1, f_2, \ldots, f_n \rangle$ and let x, y be substrings of s such that x and y share a common overlap z where $z = f'_l f_{l+1} \ldots f_t f'_{t+1}$ for some indexes l, t with $1 < l, t < n$. Then, $\mathsf{CFL}(x)$ and $\mathsf{CFL}(y)$ will share the consecutive factors $f_{l+1}, \ldots f_t$, more precisely $\mathsf{CFL}(x) = \langle \mathsf{CFL}(x'), f_{l+1}, \ldots f_t, \mathsf{CFL}(f'_{t+1}) \rangle$ and $\mathsf{CFL}(y) = \langle \mathsf{CFL}(f'_l), f_{l+1}, \ldots f_t, \mathsf{CFL}(y') \rangle$, where $x = x'z$ and $y = zy'$. Similarly, $\mathcal{L}(x)$ and $\mathcal{L}(y)$ will share the consecutive lengths $|f_{l+1}|, \ldots, |f_t|$. It follows that two overlapping strings s and s' will share consecutive common Lyndon factors in their Lyndon factorizations, while the fingerprints of s and s' will share common k-fingers for a suitable k, where $1 \leq k \leq t$, if s and s' share t common Lyndon factors. In this paper, we propose to use the fingerprint as a signature of common overlapping strings. Another type of Lyndon factorization is based on the notion of Inverse Lyndon word: a string s is an *Inverse Lyndon word* if each proper suffix is strictly smaller than s [5]. Then, $F(s) = \langle f_1, f_2, \ldots, f_n \rangle$ is an *Inverse Lyndon factorization* for a string s, if each factor f_j is an Inverse Lyndon word. Bonizzoni et al. in [5] propose a linear time algorithm to compute a special Inverse Lyndon factorization which is

unique for s and is called *Canonical Inverse Lyndon factorization* (referred in the following by ICFL). While uniqueness and linear time computation is guaranteed by both CFL and ICFL guarantee, a fundamental property of ICFL is that it splits Lyndon words, thus allowing to further factorize too long Lyndon factors of a Lyndon factorization, if needed. We will call CFL_ICFL the factorization obtained by applying first the Standard Lyndon Factorization CFL, and then the Canonical Inverse Lyndon factorization ICFL to factors (in CFL) longer than a given threshold T. In other words, given $\mathsf{CFL}(s) = \langle f_1, f_2, \ldots, f_n \rangle$, we obtain $\mathsf{CFL\_ICFL}(s)$ by replacing each f_i longer than T with $\mathsf{ICFL}(f_i)$. Remarkably, CFL_ICFL has the main advantage of producing many factors, thus enriching the set of k-fingers to use for detecting the overlaps. Observe that, while the conservation property holds for CFL_ICFL, since ICFL splits the same Lyndon factors it is an open problem to formally prove the property for ICFL, which seems to hold also in this case, as suggested in the preliminary experiments.

Moreover, since we want to take into account the double-stranded nature of the reads, we are interested in signatures that are invariant with respect to a read and its reverse and complement. For this purpose, given any factorization F, we define a *double-stranded version* of F, denoted by F^d, having the following property: given $F^d(s) = \langle f_1, f_2, \ldots, f_n \rangle$ then $F^d(\overline{s}) = \langle \overline{f}_n, \overline{f}_{n-1}, \ldots, \overline{f}_1 \rangle$, where $\overline{f}_i$ and $\overline{s}$ are the reverse and complement of f_i and s, respectively. Observe that (when dealing with a double-stranded factorization) the fingerprint $\mathcal{L}(s)$ is the reverse of the fingerprint $\mathcal{L}(\overline{s})$. As a consequence, the two k-fingers supported by the same genomic region on two opposite overlapping reads, are one the reverse of the other. To overcome this fact, we *normalize* the k-fingers, meaning that, given a k-finger $\langle l_1, l_2, \ldots, l_k \rangle$, we take the lexicographically smallest sequence between $\langle l_1, l_2, \ldots, l_k \rangle$ and its reverse $\langle l_k, l_{k-1}, \ldots, l_1 \rangle$, by considering k-fingers as sequences over the alphabet of the natural numbers. In this way, the fingerprints of two reads extracted from the same locus on two different strands share the same normalized k-fingers. We omit the details for space constraints.

3 Probabilistic Behaviour of Fingerprints

In this section, we explore probabilistic results to be used to estimate how k-fingers capture the similarity of two sequences: we first define and analyze the collision phenomenon by estimating the probability that two sequences share some common k-fingers w.r.t. to the corresponding notion of k-mers. In the following, let Σ_s be the alphabet of the lengths of the fingerprint of a sequence W, where $|\Sigma_s| = s$, with s being the length of the sequence w. Clearly a k-finger is a k-mer of a fingerprint. Let us define the notion of *k-finger collision* reflecting the fact that a k-finger can be supported by distinct strings.

Definition 1 (k-finger collision). *Let $x, y \in \Sigma^*$ be two sequences. Let F be a Lyndon-based factorization. Let $f(x)$ and $f(y)$ be the fingerprints for x and y with respect to F, respectively. Let $k(x)$ be a k-finger of $f(x)$ and $k(y)$ be a k-finger of $f(y)$. Let $s_{k(x)}$ and $s_{k(y)}$ be two substrings of x and y supporting $k(x)$*

and $k(y)$, respectively. If $k(x) = k(y)$ and $s_{k(x)} \neq s_{k(y)}$, then we say that there exists a collision *between $s_{k(x)}$ and $s_{k(y)}$.*

Since k-fingers of w are k-mers over the alphabet Σ_s, the collision phenomenon can be studied by exploiting results already obtained in literature in the case of k-mers [2,17]. Observe that, the length of the fingerprints can vary for each read considered, and so, to extend those results to our case we will indicate with s the *mean length* of a generic set of reads considered.

Let x and y be two random sequences, both of length n, and x_s, y_s the corresponding fingerprints. Then, the probability P_r that x_s and y_s will share a k-finger by chance and not because of a real similarity can be defined as:

$$P_r = 1 - (1 - s^{-k})^{n-k+1} \tag{1}$$

Observe that since s is far greater than the usual alphabets on which reads are defined (e.g. DNA), using k-fingers drastically reduce the occurrence of random matches differently from the usage of k-mers. However, using k-fingers has an intrinsic and unavoidable probability of collision, which is related to the dimension of the set considered and can be computed as illustrated below and easily taken in consideration to limit the effect of this phenomenon in any formula used. Now we will show how to use P_r to compute the expected number of k-fingers shared by two fingerprints of arbitrary length. To this, we first define the probability that a collision occurs, named *collision probability*, and then the probability that none of the reads are corrupted by any error.

Collision Probability. One factorization can be view as a function which randomly maps a sequence to a certain number of integers with a uniform distribution. This means that all the integers have the same probability to be picked up. According to this assumption, the probability of k-finger collision is a generalization of the well known "birthday problem"[1]. Observe that for each item of a k-finger we have s possible values, and so the space of possible values for the k-uple is s^k. Let suppose to pick a single value. After that, there are $s^{k-1}(s-1)$ remaining possibilities that are unique from the first. Therefore, the probability of randomly generating two k-uple that are unique form each other is $\frac{s^{k-1}(s-1)}{s^k} = \frac{s-1}{s}$. After that, there are $s^{k-1}(s-2)$ remaining possibilities that are unique for the first two, which means that the probability of randomly generating three k-uple that are unique is $\frac{s^{k-1}(s-1)}{s^k} \times \frac{s^{k-1}(s-2)}{s^k} = \frac{s-1}{s} \times \frac{s-2}{s}$, and so on. Based on this argument, we can give the following proposition:

Proposition 2. *[k-finger collision probability] Let s be the mean length of the set of sequences considered and let M be the total number of k-fingers generated. Then, the probability that at least two of them are equal is:*

$$1 - \frac{s^{k-1}(s-1)}{s^k} \times \frac{s^{k-1}(s-2)}{s^k} \times \cdots \times \frac{s^{k-1}(s-M-1)}{s^k} \tag{2}$$

Which can be approximated by $P_c = 1 - e^{\frac{-M(M-1)}{2s^k}}$

[1] https://preshing.com/20110504/hash-collision-probabilities/

Detecting Similarity Using P_c and P_r. In order to detect similarity between two sequence using fingerprints it is necessary to be able of correctly distinguish if a common k-finger is coming from a random match, a collision or a shared region. An effective solution could be calculating a threshold using the *Bernoulli distribution*. Let f_1, f_2, be two fingerprints sharing a number x of k-fingers, and let a false match be a single match generated by a random match or a collision. We define as success the event of having one false positive match (a random matching or a collision), to which corresponds the probability $P_c + P_r$ as calculated before. Respectively, we consider as failure both a matching coming from a common region or the case of not having a match, and as number of the experiments the number of k-fingers of the longest fingerprint between the two considered.

Given a dataset of reads, we can use the Bernoulli formula to compute the minimum number x of k-fingers matching needed to say that two reads share a common region. Specifically, first we set a threshold δ such that $\binom{n}{x} p^x \cdot (1-p)^{n-x} < \delta$. Then we compute the solution x which represent the minimum number x of k-fingers matching needed to say that two reads are in overlap. So, let f_1, f_2, be two fingerprints sharing a number x' of k-fingers, if $x' > x$ then we can say that f_1 and f_2 are similar with a probability depending on δ.

4 The Methodology and Experiments

In this section we provide the details of the experiments we carried out to assess the effectiveness of fingerprints and k-fingers as machine-interpretable representation for sequencing reads. We assume the reader is familiar with the basic notions of machine learning (see [19] for further details). More precisely, the following question is addressed: can fingerprints and k-fingers be used to assign RNA-Seq reads to the correct origin gene? To this aim, we explore two machine learning approaches to classify RNA-seq reads; the first one is based on fingerprints and the second one is based on k-fingers. We implemented the following five-step methodology, in relation to the input data used in our experiments:

1. *training data collection*: given a `FASTA` file containing the transcript sequences annotated for 6040 genes from chromosomes `1`, `17` e `21` (`havana` and `ensembl_havana`; 3 transcripts per gene on average), we have collected the set of all the 100-long substrings extracted from the transcripts of 100 randomly selected genes (we have considered at most 4 randomly selected transcripts for each gene), obtaining a set $\mathcal{C}$ containing 797407 substrings. For each substring we have also collected the `ID` of the origin gene;
2. *feature extraction*: we have considered the double-stranded factorization algorithms CFL^d, ICFL^d and $\mathsf{CFL\_ICFL}^d$ (with threshold $\mathtt{T} \in \{10, 20, 30\}$) for a total of 5 factorization algorithms. For each algorithm, we have computed the fingerprints of the strings in $\mathcal{C}$ obtaining 5 datasets (each one to be used as feature dataset in the fingerprint-based approach), and then, for each dataset of fingerprints, we have extracted all the *k-fingers* for k from 3 to 8

obtaining 30 datasets (to be used as features in the k-finger based approach); we remark that, in the fingerprint-based (resp. k-fingers-based) approach, one fingerprint (resp. one k-finger) represents one *sample* in the feature dataset, and each length in such a fingerprint (resp. k-finger) is a *feature*; we remark that the choice of values for k and T is the result of a series of observations on preliminary experiments, and is not linked to the optimization phase of the machine learning models described below.

3. *labeling*: each fingerprint (or k-finger) in a feature dataset is labeled by the `ID` of the origin gene in order to have a class for each input gene;
4. *validation and classification*: each feature dataset is split into a *training set* (80% of the samples) and a *testing set* (the remaining 20%); the data have been normalized by using the `minmaxscaler` technique; the k-fold cross-validation was performed to validate different machine learning models; some of the most used classification models have been tested on the testing set with the best parameters found during the previous step;
5. *testing*: in order to simulate reads from the set of our 100 input genes in a reliable setting, we have considered a set with 10Million RNA-Seq reads simulated with Flux Simulator with different gene-expression levels by considering 9403 Human genes from chromosomes 1, 17, and 21; then we have extracted from such set the reads originated from the 100 input genes, thus obtaining 285628 reads; we used the best classification model obtained in the previous step to classify the reads by the gene locus; we note that the set of reads obtained was unbalanced, i.e., 142266 reads simulated from the most expressed gene (`ID ENSG00000132517`), and only 2 reads simulated from the least expressed one (`ID ENSG00000116205`); due the multiclass nature of the problem considered, we first calculated the precision, recall, and f-score values for each of the 100 genes and then the average scores.

The last three steps are detailed in the following two sections with respect to the considered approach (fingerprint based or k-finger based). We implemented our methodology in Python by using the *scikit-learn* library[2]. All the input and output files and the Python scripts are available online[3].

4.1 Fingerprint-based Approach

Validation and Classification. By performing a 5-fold cross-validation using the `gridsearchcv` method, we have obtained the best parameters to train and test the following chosen classification models: *Random Forest* (`RF`), *Logistic Regression* (`LR`), and *Multinomial Naive Bayes* (`MNB`). We remark that the goal of this step was to chose the model having the highest accuracy, precision and recall scores for all classes.

As a result, we observed that the `RF` model always outperforms the other models, so in the following we report only its results (Table 1). In general the

[2] https://scikit-learn.org/
[3] https://github.com/rzaccagnino/DeepShark

Table 1. Results obtained with RF model for each type of double-stranded factorization

Factorization		Accuracy	Precision	Recall	F-score
CFL^d		0.72	0.72	0.72	0.72
ICFL^d		0.85	0.85	0.85	0.85
CFL_ICFL^d	T=10	0.92	0.93	0.92	0.93
	T=20	**0.93**	**0.94**	**0.93**	**0.94**
	T=30	0.92	0.93	0.92	0.92

Table 2. Results obtained on CFL_ICFL^d for T = 20 with MNB, LR, and RF

Model	Accuracy	Precision	Recall	F-score
Multinomial Naive Bayes (MNB)	0.42	0.23	0.30	0.44
Logistic Regression (LR)	0.44	0.43	0.30	0.45
Random Forest (RF)	**0.93**	**0.94**	**0.93**	**0.94**

RF models trained with fingerprints obtained by the factorizations CFL_ICFL^d outperforms the other RF models. Moreover, the best result have been obtained by using the factorization CFL_ICFL^d with T = 20, with accuracy 0.93, precision 0.94, recall 0.93, and f-score 0.94. Furthermore, for all the factorizations CFL_ICFL^d the performance does not vary significantly as the parameter T changes. This could be due to the fact that the longest factors to break were the same, and they were most probably longer than 30 bases. In Table 2 we also report the average results obtained by all the models with CFL_ICFL^d with T = 20.

Testing. Best results were obtained using the RF model with CFL_ICFL^d and $T = 20$. More precisely, we obtained an average weighted precision of 0.85, a recall of 0.42, and an f-score of 0.55. Precision and recall are computed for each investigated gene G as $TP/(TP+FP)$ and $TP/(TP+FN)$ (respectively), where TP is the number of reads simulated from G which are correctly assigned to G, FP is the number of reads simulated from a different gene which are erroneously assigned to G, and FN is the number of reads simulated from G which are erroneously assigned to a different gene.

4.2 *k*-finger-based Approach

Validation and Classification. As observed before, also in this approach the RF model always outperforms the other models and so, we only report its results. Results of RF classification tests are shown in Table 3.

The best results have been obtained by using the factorization CFL_ICFL^d with T = 30 and k = 8, i.e., accuracy 0.94, precision 0.94, recall 0.93 and f-score 0.94. To compare it with the other models, in Table 4 we report the average results obtained by all the models with CFL_ICFL^d with T = 30 and k = 8. We further analyzed different choices of parameter k and different factorizations (details omitted for space constraints). We obtained results ranging from 0.38 to 0.94

Table 3. Best results obtained with RF model on CFL^d, ICFL^d and CFL_ICFL^d, T = 5

Factorization	K value	Accuracy	Precision	Recall	F-score
CFL^d	8	0.90	0.93	0.86	0.89
ICFL^d	5	0.91	0.92	0.87	0.88
CFL_ICFL^{d}30	8	0.94	0.94	0.93	0.94

Table 4. Results obtained on the best factorization CFL_ICFL^d for T = 30 and k = 8

Model	Accuracy	Precision	Recall	F-score
Multinomial Naive Bayes (MNB)	0.42	0.35	0.33	0.46
Logistic Regression (LR)	0.41	0.65	0.71	0.50
Random Forest (RF)	**0.94**	**0.94**	**0.93**	**0.94**

accuracy, according to the type of factorization and the k value. We can notice that in most cases, with increasing values of k the accuracy improves for any type of factorization. Furthermore, for k values between 3 and 6 the best accuracy is always obtained with ICFL^d, while the best accuracy (0.94) was achieved by CFL_ICFL^d with T = 30 and $k = 8$. We also observed that experiments conducted with k ≤ 5 show that CFL_ICFL^d factorizations (for T=10,20,30) always result in lower accuracy compared to other methods.

Testing. To evaluate the effectiveness of the k-fingers approach to classify RNA-Seq data by the gene locus, we have defined a *Rule-based read classifier* [19]. During several preliminary test experiments we have defined many *criteria* to deduce the classification of a read by the classification of its k-fingers. By empirical observations two possible criteria have been selected: *Majority* and *Threshold*. According to the *Majority* criteria, a gene g reaches the majority for a given read if at least half of the read k-fingers are classified to g; therefore the read is classified to g. The idea of the *Threshold* criteria, instead, is to use the lowest probability whereby a k-finger was correctly classified to a gene. For classifying a read, *(i)* first this value is subtracted from the classification probabilities of each k-finger extracted from the read (*margins*), *(ii)* then the k-finger reaching the highest margin is selected, and *(iii)* finally the read is classified to the gene for which such a margin has been reached by the selected k-finger.
We tested such criteria in different orders and best results were obtained when combined in the following way: *(1)* if the majority is reached, then the read is classified to the gene which achieves majority, otherwise *(2)* the read is classified to the gene which achieves the highest threshold. In Algorithm 1 we describe the classifier based on such techniques. It takes as input the read to classify (`read`), the k-fingers of the read (`k_fingers`), a k-finger classifier (`classifier`), and a list containing, for each gene, the lowest probability to be classified by `classifier` to such a gene (`genes_thresholds`). As output, it returns the ID of the gene to which `read` is classified (`ID_gene`). First, the lowest probability is computed by `genes_thresholds` (line 1). Then, `classifier` is used to classify `k_fingers`

Algorithm 1: Rule-based read classifier

```
   Input  : read, k_fingers, classifier, genes_thresholds.
   Output : ID_gene
 1 min_threshold ← min(genes_thresholds);
 2 classes ← classify(k_fingers, classifier);
 3 best_classes, frequence ← most_frequent(classes);
 4 if (size(best_classes) == 1) and frequence >= size(k_fingers/2) then
 5 |   // Majority criteria;
 6 |   ID_gene = best_classes[0];
 7 else
 8 |   // Threshold criteria;
 9 |   probability_for_gene ← probability_classify(k_fingers,
   |    classifier);
10 |   max_index_list ← [];
11 |   max_margin_list ← [];
12 |   foreach sample ∈ probability_for_gene do
13 |   |   sample_margin ← [(p - min_threshold) for p ∈ sample];
14 |   |   max_index ← argmax(sample_margin);
15 |   |   max_index_list.append(max_index);
16 |   |   max_margin ← amax(sample_margin);
17 |   |   max_margin_list.append(max_margin);
18 |   max_index ← argmax(max_margin_list);
19 |   best_class ← max_index_list[max_index];
20 |   ID_gene ← best_class;
21 return ID_gene;
```

and so the most frequent genes (`best_classes`), and the number of occurrences of such genes are extracted by the results of the k-fingers classification (line 2). If `best_classes` contains one only a gene and such a gene reached the majority then it was returned (line 6). Otherwise, for each k-finger `sample` in `k_fingers`, given the probabilities to be classified to each gene (line 9), the algorithm first computes the classification margin for each gene (line 13) and then selects the gene for which the highest margin has been reached by `sample` (lines 15-17). After having computed such a value for each k-finger `sample` in `k_fingers` the algorithm returns the first of such genes (line 20).

We used the k-finger classifiers trained in the previous step to implement the rule-based read classifier defined in Algorithm 1. Due to the multiclass nature of the problem considered, we first calculated the precision, recall, and f-score values for each of the 100 genes and then the average scores. Unlike the result obtained during the training step, in which the `RF` model with $\mathsf{CFL\_ICFL}^d$, `T = 30` and `k = 8` achieved the best scores respect to the k-finger classification, the result obtained for the read classification show that the best scores were achieved by using the `RF` model with ICFL^d and `k = 5`. This is because, by reducing the k value, we are able to consider more local parts of each read, obtaining more classifications which make the rule-based read classifier more robust respect to

the presence of errors in the read. As first result, we obtained average weighted precision of 0.91, recall 0.77, and f-score 0.82. Given the substantial difference respect the scores obtained with the approach of Section 4.1, we have decided to perform a more in-deep analysis of rule-based read classifier performance. We have observed that the precision value was directly proportional to the support value, which is the number of reads assigned to the given gene, with average precision value from 0.99 for the most expressed gene (`ID ENSG00000132517`) to 0.001 for the less expressed (`ID ENSG00000116205`). However, the average recall was always greater than 0.62, and in general very high for the less expressed genes (about 1). This means that the classification of the few samples related to such genes is almost always exact, and so the problem of the low precision value was attributable to the misclassification of the reads related to the most expressed genes. To assess our hypothesis we repeated the same experiment on several balanced subsets of the 285628 reads: *(1)* the subset of reads related to the 31 most expressed genes which have at least 1000 samples (1000 reads $\times$ 31 genes), *(2)* the subset of reads related to the 15 genes which have at least 14 samples and less of 100 reads (14 reads $\times$ 15 genes), *(3)* 4 reads for each of the most expressed 98 genes (4 reads $\times$ 98 genes), and finally *(4)* 2 reads for each of the 100 genes (2 read $\times$ 100 genes). As result, we obtained arithmetic average precision of 0.90, recall 0.77, and f-score 0.77, that by construction corresponds also to the weighted average scores. This result confirm our hypothesis and so we can consider the average weighted precision of 0.91, recall 0.77, and f-score 0.82 as representative results of classifier performance.

5 Discussion

In this paper we investigate the notion of fingerprint as novel signature for representing sequencing reads. While this notion could be used also in a combinatorial setting for comparing biological sequences, being a numeric representation of sequences we decided to explore its use in a machine learning approach to classify sequencing reads. We experimented different notions of Lyndon-based factorizations, i.e. CFL^d, ICFL^d and $\mathsf{CFL\_ICFL}^d$ to evaluate the best representation. We assess its potentiality in assigning RNA-Seq reads to their origin gene by carrying out several read classification experiments. The training-test steps show that both notions in our method allow to achieve comparable high scores. Nevertheless, differences among the use of the two notions emerged during the testing step, where the performances were tested on a simulated RNA-Seq sample. In this case, the presence of errors in the reads and the different level of gene expression, leading to an unbalanced number of reads per gene, produce differences in the performance when using fingerprint versus k-fingers of the read. This significant difference between the results in terms of recall and F-score can be explained with the presence of errors in the reads. Since the fingerprint method considers the whole read, this is sensitive to the presence of errors. Conversely, dividing the reads in multiple k-fingers enables to get a more robust classification, allowing to better isolate the corrupted portion of reads, and to identify the correct

class with the conjunct application of the rule-based classifier. In conclusion, the results prove the effectiveness of the notions of fingerprint and k-finger as machine learning interpretable representation. This work still leave many open questions that need to be addressed, including how fingerprints perform compared to other machine learning representations of biological sequences [1,11] and how the probabilistic analysis presented in the paper is reflected on real sequencing data. Finally, could be fingerprints used in a combinatorial approach to detect the overlap of sequencing reads?

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement number [872539].

References

1. Asgari, E., Mofrad, M.R.: Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS ONE **10**(11), e0141287 (2015)
2. Berlin, K., Koren, S., Chin, C.S., Drake, J.P., Landolin, J.M., Phillippy, A.M.: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature Biotechnology **33**(6), 623–630 (2015)
3. Berstel, J., Perrin, D.: The origins of combinatorics on words. European Journal of Combinatorics **28**(3), 996–1022 (2007)
4. Bonizzoni, P., De Felice, C., Zaccagnino, R., Zizza, R.: Lyndon words versus inverse lyndon words: Queries on suffixes and bordered words. In: LATA 2020. LNCS, vol. 12038, pp. 385–396. Springer (2020)
5. Bonizzoni, P., De Felice, C., Zaccagnino, R., Zizza, R.: Inverse Lyndon words and inverse Lyndon factorizations of words. Adv. in App. Math. **101**, 281–319 (2018)
6. Bonizzoni, P., De Felice, C., Zaccagnino, R., Zizza, R.: On the longest common prefix of suffixes in an inverse Lyndon factorization and other properties. Theoretical Computer Science (in press)
7. Chen, K.T., Fox, R.H., Lyndon, R.C.: Free differential calculus, iv. the quotient groups of the lower central series. Annals of Mathematics **68**(1), 81–95 (1958)
8. Delgrange, O., Rivals, E.: Star: an algorithm to search for tandem approximate repeats. Bioinformatics **20(16)**, 2812–2820 (2004)
9. Denti, L., Pirola, Y., Previtali, M., Ceccato, T., Della Vedova, G., Rizzi, R., Bonizzoni, P.: Shark: fishing relevant reads in an RNA-Seq sample. Bioinformatics (2021)
10. Duval, J.P.: Factorizing words over an ordered alphabet. Journal of Algorithms **4**(4), 363–381 (1983)
11. Kimothi, D., Soni, A., Biyani, P., Hogan, J.M.: Distributed representations for biological sequence analysis. arXiv preprint arXiv:1608.05949 (2016)
12. Kumar, P., Kumar, P., Krishna, P.R., Raju, S.B.: Pattern discovery using sequence data mining: applications and studies. IGI Publishing (2011)
13. Köppl, D., Hashimoto, D., Hendrian, D., Shinohara, A.: In-Place Bijective Burrows-Wheeler Transforms. In: Combinatorial Pattern Matching (2020)

14. Lothaire, M.: Combinatorics on words. Cambridge University Press (1967)
15. Lyndon, R.C.: On burnside's problem. Transactions of the American Mathematical Society **77(2)**, 202–215 (1954)
16. Motomura, K., Fujita, T., Tsutsumi, M., Kikuzato, S., Nakamura, M., Otaki, J.M.: Word decoding of protein amino acid sequences with availability analysis: a linguistic approach. PLoS ONE **7**(11), e50039 (2012)
17. Ondov, B.D., Treangen, T.J., Melsted, P., Mallonee, A.B., Bergman, N.H., Koren, S., Phillippy, A.M.: Mash: fast genome and metagenome distance estimation using minhash. Genome Biology **17**(1), 132 (2016)
18. Srinivasan, S.M., Vural, S., King, B.R., Guda, C.: Mining for class-specific motifs in protein sequence classification. BMC Bioinformatics **14**(1), 96 (2013)
19. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to data mining. Pearson Education India (2016)
20. Vries, J.K., Liu, X.: Subfamily specific conservation profiles for proteins based on n-gram patterns. BMC Bioinformatics **9**(1), 72 (2008)