A Computational Measure for the Semantic Readability of Segmented Texts

Valentino Santucci[1(✉)], Umberto Bartoccini[1], Paolo Mengoni[2], and Fabio Zanda[1]

[1] University for Foreigners of Perugia, Perugia, Italy
{valentino.santucci,umberto.bartoccini,fabio.zanda}@unistrapg.it
[2] School of Communication and Film, Hong Kong Baptist University, Hong Kong, China
pmengoni@hkbu.edu.hk

Abstract. In this paper we introduce a computational procedure for measuring the semantic readability of a segmented text. The procedure mainly consists of three steps. First, natural language processing tools and unsupervised machine learning techniques are adopted in order to obtain a vectorized numerical representation for any section or segment of the inputted text. Hence, similar or semantically related text segments are modeled by nearby points in a vector space, then the shortest and longest Hamiltonian paths passing through them are computed. Lastly, the lengths of these paths and that of the original ordering on the segments are combined into an arithmetic expression in order to derive an index, which may be used to gauge the semantic difficulty that a reader is supposed to experience when reading the text. A preliminary experimental study is conducted on seven classic narrative texts written in English, which were obtained from the well-known Gutenberg project. The experimental results appear to be in line with our expectations.

Keywords: Semantic readability of texts · Natural Language Processing · Unsupervised machine learning · Hamiltonian path

1 Introduction

As is widely known, some texts are more difficult to read than others. Books like *Ulysses* and *Finnegans Wake*, by the Irish writer James Joyce, or *Infinite Jest*, by the American author David Foster Wallace, are challenging reads also for the most voracious and experienced readers.

The readability of a written text is defined as a measure of how easy it is to understand the content of the text from a reader's point of view. It has been investigated in various works taking into account different aspects. The richness of the vocabulary, the syntactic structure of the discourse, and how the text is presented to the reader (i.e., character, background, material, etc.) are features

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13377, pp. 107–119, 2022.
https://doi.org/10.1007/978-3-031-10536-4_8

that have an impact on the readability of the texts. Considering these factors, researchers introduced of a set of qualitative and quantitative readability metrics.

Qualitative metrics are usually based on surveys with readers [15,18], where the readability is assessed by interviewing the readers on aspects such as the ease of understanding the content and the text presentation. The results of the surveys are then interpreted and assessed by experts.

Quantitative measures aim to assess the readability using semantic and linguistic features of the texts. This kind of measures can be computed automatically using Natural Language Processing (NLP) techniques [28,31].

Semantic and linguistic measures are computed by using supervised learning approaches. The learning procedure requires an annotated dataset where class labels are assigned to the texts segments by experts or are automatically discovered [12,27]. Various syntactic and lexical features are considered to assign the class labels. Syntactic features include basic attributes and properties of the texts such as word length, word count, and word frequency in sentences and in higher-level text structures (e.g., paragraphs, chapters, etc.). Lexical features are unigrams, bigrams, and the surface form of the target word. Afterwards a NLP classifier using Decision Trees, Random Forest, Support Vector Machines, or Artificial Neural Networks learn to automatically assign a complexity label to the texts [11].

Many readability measures consider the text segments (i.e. the phrases, paragraphs, chapters, etc.) as isolated parts. A few works in literature take in account the sequence of presentation of the various aprts of the text to the readers [5]. However, the structure and organization of the different segments to form high level structures has been proved to improve the readability [10]. Stories that are presented linearly are more readable than stories where the discourse, the plot, and the storylines interleave. Therefore, in this work we will try to fill this gap by defining a readability measure that take in account the sequence of presentation of the various segments that compose a text.

The rest of the paper is organized as follows. The high level description of the readability measure is provided in Sect. 2, while the detailed operational procedures for the computation of the text segment embeddings, Hamiltonian paths and the final arithmetic expression for the readability index are presented in Sects. 3, 4 and 5, respectively. Experiments are described and discussed in Sect. 6, while conclusions are drawn in Sect. 7, where future lines of research are also depicted.

2 The Semantic Readability Index for Segmented Texts

Most of the texts naturally present a shallow structure formed by linearly ordered segments. For instance, this is the case of most of narrative books, film scripts, textbooks, scientific articles and several other kinds of texts.

By denoting $[n] = \{1, 2, \ldots, n\}$, a segmented text T may be formally defined as a sequence of textual segments, i.e., $T = \langle t_1, t_2, \ldots, t_n \rangle$, where n is the total number of segments of T, and t_i, for any $i \in [n]$, denotes the i–th segment of T.

In the case of a narrative text, such segments are naturally identified with the actual chapters of the text.

Informally speaking, one way to define the semantic readability of a segmented text is to measure somehow the semantic jump between two consecutive segments and sum up these measurements for all the pairs of adjacent segments. A formal and procedural definition of this idea allows to define a quantitative semantic readability index which gauges how smooth the reading of the text is.

With the recent growth of Natural Language Processing (NLP) techniques [6], it has now become possible to numerically tackle a considerable number of semantic aspects which, in the past, were only qualitatively approached. In particular, text and word embeddings (see e.g. [7–9,19,22,29]) are among the main NLP methodologies which allow to semantically handle text data.

Given a segmented text $T = \langle t_1, t_2, \ldots, t_n \rangle$, here we define a computational procedure for computing its semantic readability $\texttt{SemRead}(T) \in [0, 1]$. Therefore, $\texttt{SemRead}(T)$ represents a semantic readability index which measures the semantic smoothness a hypothetical reader is supposed to experience when reading the text T. At the edge values, we obtain $\texttt{SemRead}(T) = 0$ when T is very difficult to read, and $\texttt{SemRead}(T) = 1$ when the reading of T is very smooth.

The procedure `SemRead` consists of three main computational stages, as depicted in Fig. 1 and described below.

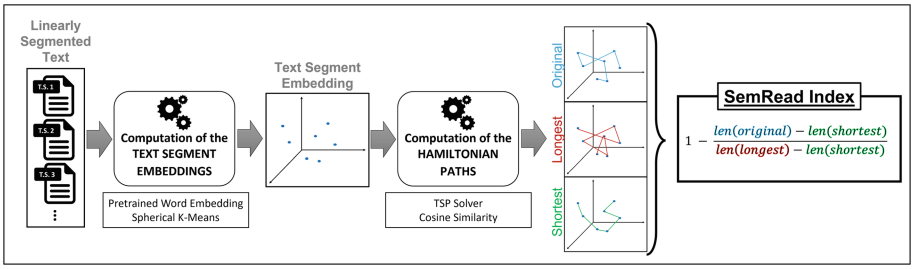

Fig. 1. High level architecture of the computational system

1. **Computation of the text segment embeddings**
   Given the segmented text $T = \langle t_1, t_2, \ldots, t_n \rangle$ in input, the objective of this stage is to compute an embedding $v_i \in \mathbb{R}^m$, for any textual segment $t_i \in T$, in such a way that two segment embeddings v_i and v_j are geometrically close – in the high dimensional embedding space $\mathbb{R}^m$ – when their corresponding text segments t_i and t_j are semantically related. Substantially, v_i is the numerical representation of t_i in the vector space $\mathbb{R}^m$, which may be obtained by exploiting one of the available word embedding models that were pretrained on very large datasets (see e.g. [13,22,24]). Further details of this computational phase are provided in Sect. 3.

2. **Computation of the Hamiltonian paths**
   By considering a suitable distance function d, this computational phase starts by computing a distance value $d(v_i, v_j) \geq 0$, for all the pairs of vectorized segments v_i, v_j. Then, these distances are used to compute the shortest and longest Hamiltonian paths [23] passing through all the vectorized segments $v_1, v_2, \ldots, v_n$ (seen as points in the multidimensional space $\mathbb{R}^m$). The rationale is that the shortest (longest) Hamiltonian path represents the smoothest (most rugged) ordering on the segments of T. Further details of this computational phase are provided in Sect. 4.

3. **Computation of the `SemRead` index**
   Given the shortest and longest Hamiltonian paths, denoted respectively as α and ω, their total lengths $\texttt{len}(\alpha)$ and $\texttt{len}(\omega)$ are computed on the basis of the distance function d. Moreover, the total length $\texttt{len}(T)$ of the actual ordering of segments (i.e., the canonical ordering $v_1, v_2, \ldots, v_n$) is computed as well. Therefore, the semantic readability index of the text T is given by the following arithmetic expression:

$$\texttt{SemRead}(T) = 1 - \frac{\texttt{len}(T) - \texttt{len}(\alpha)}{\texttt{len}(\omega) - \texttt{len}(\alpha)}. \tag{1}$$

   A more precise description of this computational phase is provided in Sect. 5.

3 Computation of the Text Segment Embeddings

This computational phase takes in input a segmented text $T = \langle t_1, t_2, \ldots, t_n \rangle$ and produces the corresponding segment vectors $\langle v_1, v_2, \ldots, v_n \rangle$, with $v_i \in \mathbb{R}^m$, by exploiting one of the nowadays available word embedding models which were pretrained on very large datasets. The rationale behind this approach is that the geometric closeness between the vectors v_i and v_j correlates with the semantic similarity between the text segments t_i and t_j.

A word embedding model [21] is a mapping from words to their vector representations, computed starting from a set of text data in a totally unsupervised way. Today, there are a variety of publicly available word embedding models pretrained on very large datasets.

In this work, we considered the `glove-wiki-gigaword-100` model[1] which is formed by 400 000 word vectors trained on a dataset formed by 6 billions of tokens from the full English Wikipedia pages and the English Gigaword corpus [14]. This model is obtained by means of the well-known GloVe algorithm [22], which exploits the probabilities on the observed words co-occurrences and matrix factorization operations in order to transform the one-hot word encodings to their continuous vector representations in the multidimensional space $\mathbb{R}^k$, where k is a user defined parameter. The model `glove-wiki-gigaword-100` is pretrained with $k = 100$.

[1] `glove-wiki-gigaword-100` is available with the Python module Gensim [17] at the url https://github.com/RaRe-Technologies/gensim-data.

We now depict a procedure to aggregate the pretrained word vectors of all the words belonging to a text segment in order to obtain a semantically meaningful mathematical representation of the text segment.

Therefore, for each text segment $t_i \in T$, we perform the following steps:

1. for any word $s \in t_i$, which is not a stop word, we obtain its word vector $w \in \mathbb{R}^k$ from the pretrained word embedding model (words not included in the model are discarded);
2. the word vectors $w_1, w_2, \ldots$ are then clustered into c groups through the Spherical K-Means algorithm [3], i.e., a variant of K-Means which considers the cosine distance measure [20] instead of the Euclidean distance, because cosine distance is known to be more appropriate for word embeddings [30];
3. the vectors of the centroids of the c groups are concatenated in order to form the final $(c \cdot k)$-dimensional representation for the text segment t_i.

In this work we have decided to set $c = 10$ and, by also recalling that $k = 100$, we have that any text segment t_i, of the inputted text T, is converted to a vector $v_i \in \mathbb{R}^{1000}$. Note anyway that any other setting for k, c and any other choice of the pretrained word embedding model could potentially be adopted, even for a different language.

For the sake of completeness, the pseudo-code of the segment embedding procedure is provided in Algorithm 1.

Algorithm 1. Computation of the segment embeddings

```
function EMBED(T = ⟨t_1, ..., t_n⟩, WEM)      ▷ WEM is a pretr. word emb. model
    T^emb ← ⟨ ⟩                                ▷ The embedding of T, initially empty
    for i ← 1 to n do
        w ← ⟨ ⟩                                ▷ List of word vectors, initially empty
        for each token tok ∈ t_i do
            convert tok to lowercase
            if tok is a stop word or tok ∉ WEM then
                skip tok and continue with the next token
            append the word vector WEM[tok] to w
        centroids ← SphericalKMeans(w)
        v_i ← concatenate the centroids into a vector
        append the segment embedding v_i to T^emb
    return T^emb
```

4 Computation of the Hamiltonian Paths

After the previous computational phase, the n text segments are represented by n vectors $v_1, \ldots, v_n \in \mathbb{R}^m$, with $m = 1000$ for the setting here adopted.

It is now possible to compute a distance matrix D such that $D_{i,j}$ contains a distance measure between v_i and v_j. As previously discussed, the cosine distance is known to be the most suitable distance measure for mathematical text representations [30]. Therefore, we set

$$D_{i,j} = 1 - \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||}, \tag{2}$$

where: the numerator $v_i \cdot v_j$ is the dot product between the vectors v_i and v_j, $||v||$ is the Euclidean norm of the vector v, while the second term in Eq. (2) is subtracted to 1 because it is the *cosine similarity* – defined in $[0, 1]$ – which we want to complement in order to have the *cosine distance*. Clearly, $D_{i,j} \in [0, 1]$, $D_{i,j} = D_{j,i}$ and $D_{i,i} = 0$, for $i, j \in \{1, \ldots, n\}$.

Since the distance between the text embeddings correlates with their semantic similarity, we can now measure the semantic smoothness of a sequence of text segments as the total distance traveled by linearly moving through their embeddings.

Formally, an ordering of the n text segments can be represented by a permutation $\pi \in \mathcal{S}_n$ of the set of indices $[n] = \{1, 2, \ldots, n\}$. Hence, $\pi_i \in [n]$ is the index of the i-th segment in the ordering π, while the total semantic length $\texttt{len}(\pi)$ of the ordering π is given by the sum of the distances between all the pairs of segment vectors which are adjacent in π, i.e., more formally:

$$\texttt{len}(\pi) = \sum_{i=1}^{n-1} D_{\pi_i, \pi_{i+1}}. \tag{3}$$

For a segmented text $T = \langle t_1, t_2, \ldots, t_n \rangle$ in input, the actual ordering of segments is represented by the identity permutation $\iota = \langle 1, 2, \ldots, n \rangle$, whose semantic length is $\texttt{len}(\iota)$.

Moreover, for the purposes of this work, it is important to compute the shortest and longest orderings $\alpha, \omega \in \mathcal{S}_n$ such that

$$\alpha = \arg\min_{\pi \in \mathcal{S}_n} \texttt{len}(\pi), \tag{4}$$

$$\omega = \arg\max_{\pi \in \mathcal{S}_n} \texttt{len}(\pi). \tag{5}$$

The orderings α and ω are the shortest and longest Hamiltonian paths passing through the points $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$ and such that their distances are defined as in Eq. (2).

Interestingly, the Hamiltonian Path Problem (HPP) [23], i.e., the problem of computing the shortest Hamiltonian path, can be easily reduced to the Traveling Salesman Problem (TSP) [2,16], i.e., the problem of computing the shortest Hamiltonian cycle (also known as "TSP tour"). This reduction allows to exploit the very efficient TSP solvers nowadays available that, though the NP-hardness of TSP, are often able to exactly solve a TSP instance formed by dozens or

hundreds of points in few seconds. In this work, we adopt the well-known exact TSP solver called Concorde[2] [1].

Given an HPP instance, the corresponding TSP instance is obtained by simply adding a fake point with null distance from all the other points. Formally, an HPP instance is defined by the distance matrix $D \in [0,1]^{n \times n}$ of Eq. (2), thus the corresponding TSP instance $D' \in [0,1]^{(n+1)\times(n+1)}$ contains D and has one extra row and one extra column, that we both identify with the index 0, such that

$$D'_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ D_{i,j} & \text{otherwise.} \end{cases} \tag{6}$$

The TSP solver is executed on the TSP instance D' and returns the shortest TSP tour, which is formed by all the indexes in $\{0, 1, \ldots, n\}$. Then, the tour is broken and unrolled at the position of the fake index 0 which is removed as well, thus to obtain the permutation $\alpha \in \mathcal{S}_n$, which is guaranteed to be the shortest Hamiltonian path for the HPP instance D, as required by Eq. (4).

Finally, the longest Hamiltonian path is computed in a similar way, but considering an additional reduction step – known as "MAX-TSP to TSP reduction" [4] – which mainly consists in complementing the TSP matrix. Formally, the TSP solver is executed on the TSP instance $D'' \in [0,1]^{(n+1)\times(n+1)}$ defined as follows:

$$D''_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ 1 - D_{i,j} & \text{otherwise.} \end{cases} \tag{7}$$

The shortest TSP tour for D'' is the longest tour for D', therefore removing the fake point from the tour gives rise to the permutation $\omega \in \mathcal{S}_n$, which represents the longest Hamiltonian path for the HPP instance D, as required by Eq. (5).

5 Computation of the Readability Index

By using the notation provided in Sect. 4, it is possible to rewrite Eq. (1) as follows:

$$\texttt{SemRead}(T) = 1 - \frac{\texttt{len}(\iota) - \texttt{len}(\alpha)}{\texttt{len}(\omega) - \texttt{len}(\alpha)} = \frac{\texttt{len}(\omega) - \texttt{len}(\iota)}{\texttt{len}(\omega) - \texttt{len}(\alpha)}. \tag{8}$$

Clearly, since the length of the actual ordering of segments ι cannot be lesser than the length of the shortest segments ordering α and greater than the length of the longest segments ordering ω, i.e., $\texttt{len}(\alpha) \leq \texttt{len}(T) \leq \texttt{len}(\omega)$, we have that $\texttt{SemRead}(T) \in [0,1]$.

Therefore, Eq. (8) states that $\texttt{SemRead}(T)$ is a normalized value representing how close $\texttt{len}(\iota)$ is to $\texttt{len}(\alpha)$ and how far it is from $\texttt{len}(\omega)$. At one extreme, when $\texttt{len}(\iota) = \texttt{len}(\alpha)$, the passing through the text segments results in a very

[2] Concorde is available from https://www.math.uwaterloo.ca/tsp/concorde.html.

smooth "semantic walk", thus T is supposed to have a high readability and $\texttt{SemRead}(T) = 1$. Conversely, when $\texttt{len}(\iota) = \texttt{len}(\omega)$, $\texttt{SemRead}(T) = 0$ and the "semantic walk" through the text segments appears very rugged.

6 Experiments

A series of computational experiments was conducted in order to validate our proposal. The texts considered for this experimental analysis are described in Sect. 6.1. Moreover, for the sake of reproducibility, Sect. 6.2 provides all the details of the algorithmic settings adopted in the experimentation. Finally, the experimental results are presented and discussed in Sect. 6.3.

6.1 Selected Texts

For this first series of experiments, we decided to select seven very popular texts from the classic literature, written by native English authors.

The texts were obtained from the well-known public repository of the Gutenber project [32], which mainly includes the plain text versions of a variety of out-of-copyright books – under the US copyright law (i.e., published 95 or more years ago).

The list of the seven selected texts is presented in Table 1 which includes: the identifier of the text used later on in this paper, the title and the author of the text, and the number of text segments in which it is divided.

Table 1. Texts considered in the experimentation.

Id	Title	Author	Segments
`AliceWonderland`	Alice's Adventures in Wonderland	L. Carrol	12
`SherlockHolmes`	The Adventures of Sherlock Holmes	A.C. Doyle	12
`GreatGatsby`	The Great Gatsby	F.S. Fitzgerald	9
`DorianGray`	The Picture of Dorian Gray	O. Wilde	20
`TimeMachine`	The Time Machine	H.G. Wells	16
`WizardOz`	The Wonderful Wizard of Oz	L.F. Baum	24
`Ulysses`	Ulysses	J. Joyce	18

These texts are very popular narrative books which are already subdivided into parts, or chapters, by their respective authors, therefore these original chapters are considered as segments in this work. It is worthwhile to note that this choice relieved us of the task of arbitrarily splitting the texts into linear segments.

Finally, the number of segments in the selected texts, as shown in Table 1, goes from 9 ("The Great Gatsby") to 24 ("The Wonderful Wizard of Oz").

6.2 Experimental Setting

For the sake of reproducibility, we provide in Table 2 the algorithmic settings of the computational procedures of `SemRead` which were adopted in this experimentation.

Table 2. Settings for the algorithmic components used in the experimentation.

Tokenization	Tokenizer from Gensim 4.1.2 every token is converted to lower case English stop words from NLTK 3.7
Word embedding	Gensim model = glove-wiki-gigaword-100 dimensionality = 100
Spherical K-Means	Python module = sphere cluster 0.1.7 no. of clusters = 10 no. of executions = 10 max iterations = 300 tolerance = 10^{-4}
Distance measure	Cosine distance
TSP solver	Concorde 3.12.19 precision = 6 digits default settings

The word embedding model and distance measure adopted were already discussed in the previous sections. The widely adopted tokenizer of the Gensim library [17] was considered. An execution of the Spherical K-Means procedure [3] depends on the initial random choice of the clusters' centroids thus, to increase the robustness of the clusterization, ten executions are carried out and the best clusterization is used to build up the segment embeddings. Finally, the Concorde TSP solver [1] is run with its default parameter setting and, since it only accepts an integer distance matrix, the real values in $[0, 1]$ of our distances are multiplied by 10^6 and rounded up to the closest integer.

6.3 Experimental Results

The semantic readability measure `SemRead` was computed on the seven texts indicated in Table 1 by adopting the algorithmic settings described in Table 2. Therefore, for any text T, the value `SemRead`(T) is provided in Table 3 together with additional information such as: the number of segments in T and the lengths of the actual, shortest and longest segment orderings, `len`(ι), `len`(α) and `len`(ω), respectively. The results are ordered from the most readable to the less readable text.

The results shown in Table 3 may be commented as follows.

Table 3. Experimental results.

Text	SemRead	len(ι)	len(α)	len(ω)	Segments
GreatGatsby	0.53	4.49	3.78	5.29	9
WizardOz	0.47	13.86	11.06	16.35	24
DorianGray	0.46	11.49	8.84	13.70	20
TimeMachine	0.45	9.14	7.18	10.76	16
SherlockHolmes	0.45	5.92	4.77	6.84	12
Ulysses	0.36	12.51	9.12	14.38	18
AliceWonderland	0.26	7.11	5.61	7.64	12

GreatGatsby appears to be the most readable text in our analysis, while Ulysses and AliceWonderland have the lowest readability scores. This appears to be in line with our expectations. In fact, although a fully objective validation is not inherently possible in this case, it is somehow commonly accepted that the GreatGatsby represents a straightforward reading, while Ulysses and AliceWonderland have a more complex semantic structure.

The highest readability score is 0.53, quite far from the theoretical upper bound of 1. This may indicate that the text genre considered in this work, i.e., classic narrative books, does not represent the most readable kind of texts. However, further experimentation with other text genres is required to corroborate this hypothesis.

Four out of seven texts have a readability score in the very narrow range $[0.45, 0.47]$, thus plausibly indicating that their readability is nearly identical.

Finally. the lengths len(ι), len(α), len(ω) and the number of the segments does not have a significant influence on the readability score. In fact, a quick inspection to Table 3 shows that their respective columns have a low correlation with the SemRead column.

7 Conclusion and Future Work

In this work we introduced a novel readability measure SemRead that take in account the sequence of presentation of the various text segments that compose a text. The readability index computation is based on an unsupervised learning approach [33] that first proceeds with the computation of the text segment's embeddings and subsequently compute the Hamiltonian paths connecting the various segments. Experiments have been performed on a series of very popular texts from the classic literature. The assessment of the measurement performance shows that the results are in line with the expectations. The SemRead measure is able to distinguish between texts that are easy to read from the ones that present more complex storylines and semantic structures.

Although the findings of the present study are very useful, the semantic readability score was computed on books which are not necessarily meant to be

read in a different order of chapters than the one that they were given by their authors when completed. It would therefore be of great value to carry out a follow up study considering books which were indeed devised with an increasing semantic - yet not only semantic - complexity in mind, i.e. the so-called *graded readers*. Graded readers are easy-to-read books addressing L2 language learners and are often presented split up into chapters. They can be found addressing different language proficiency level readers and their vocabulary is frequently "limited" to a certain lemma frequency threshold, according to the proficiency of the targeted readership. Furthermore, computing a semantic readability score could be a farther validating element to be taken into account when publishing new graded readers.

Finally, another interesting line of research is to encode the problem as an instance of the well-known Linear Ordering Problem [25,26], thus to take into account all pairwise relationships among the text segments.

References

1. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: TSP cuts which do not conform to the template paradigm. In: Jünger, M., Naddef, D. (eds.) Computational Combinatorial Optimization. LNCS, vol. 2241, pp. 261–303. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45586-8_7
2. Baioletti, M., Milani, A., Santucci, V., Bartoccini, U.: An experimental comparison of algebraic differential evolution using different generating sets. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, pp. 1527–1534. (2019). https://doi.org/10.1145/3319619.3326854
3. Banerjee, A., Dhillon, I.S., Ghosh, J., Sra, S., Ridgeway, G.: Clustering on the unit hypersphere using von Mises-Fisher distributions. J. Mach. Learning Res. **6**(9), 1345–1382 (2005)
4. Barvinok, A., Gimadi, E.K., Serdyukov, A.I.: The maximum TSP. In: Gutin, G., Punnen, A.P. (eds.) The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, vol. 12, pp. 585–607. Springer, Boston (2007). https://doi.org/10.1007/0-306-48213-4_12
5. Calfee, R.C., Curley, R.: Structures of prose in content areas. In: Understanding Reading Comprehension, pp. 161–180 (1984)
6. Chowdhary, K.: Natural language processing. Fundamentals of Artificial Intelligence, pp. 603–649 (2020)
7. Church, K.W.: Word2vec. Nat. Lang. Eng. **23**(1), 155–162 (2017)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Dieng, A.B., Ruiz, F.J., Blei, D.M.: Topic modeling in embedding spaces. Trans. Assoc. Comput. Linguis. **8**, 439–453 (2020)
10. DuBay, W.H.: The principles of readability. Online Submission (2004)
11. Forti, L., Grego Bolli, G., Santarelli, F., Santucci, V., Spina, S.: MALT-IT2: a new resource to measure text difficulty in light of CEFR levels for Italian L2 learning. In: Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, May 2020, pp. 7204–7211. European Language Resources Association (2020). https://aclanthology.org/2020.lrec-1.890

12. Forti, L., Milani, A., Piersanti, L., Santarelli, F., Santucci, V., Spina, S.: Measuring text complexity for Italian as a second language learning purposes. In: Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. Florence, Italy, August 2019, pp. 360–368. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/W19-4438
13. Gourru, A., Guille, A., Velcin, J., Jacques, J.: Document network projection in pretrained word embedding space. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 150–157. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_19
14. Graff, D., Kong, J., Chen, K., Maeda, K.: English Gigaword. Linguis. Data Consortium Philadelphia **4**(1), 34 (2003)
15. Jones, M.J., Shoemaker, P.A.: Accounting narratives: a review of empirical studies of content and readability. J. Acc. Lit. **13**, 142 (1994)
16. Jünger, M., Reinelt, G., Rinaldi, G.: The traveling salesman problem. In: Handbooks in Operations Research and Management Science, vol. 7, pp. 225–330 (1995)
17. Khosrovian, K., Pfahl, D., Garousi, V.: GENSIM 2.0: a customizable process simulation model for software process evaluation. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 294–306. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79588-9_26
18. Kwolek, W.F.: A readability survey of technical and popular literature. Journalism Q. **50**(2), 255–264 (1973). https://doi.org/10.1177/107769907305000206
19. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine learning, pp. 1188–1196. PMLR (2014)
20. Li, B., Han, L.: Distance weighted cosine similarity measure for text classification. In: Yin, H., et al. (eds.) IDEAL 2013. LNCS, vol. 8206, pp. 611–618. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41278-3_74
21. Li, Y., Yang, T.: Word embedding for understanding natural language: a survey. In: Srinivasan, S. (ed.) Guide to Big Data Applications. SBD, vol. 26, pp. 83–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-53817-4_4
22. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
23. Rahman, M.S., Kaykobad, M.: On Hamiltonian cycles and Hamiltonian paths. Inf. Process. Lett. **94**(1), 37–41 (2005)
24. Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, pp. 15–18 (2019)
25. Santucci, V., Baioletti, M., Milani, A.: An algebraic differential evolution for the linear ordering problem. In: Companion Material Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 1479–1480 (2015). https://doi.org/10.1145/2739482.2764693
26. Santucci, V., Ceberio, J.: Using pairwise precedences for solving the linear ordering problem. Appl. Soft Comput. **87**, 105998 (2020). https://doi.org/10.1016/j.asoc.2019.105998
27. Santucci, V., Forti, L., Santarelli, F., Spina, S., Milani, A.: Learning to classify text complexity for the Italian language using support vector machines. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12250, pp. 367–376. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58802-1_27

28. Santucci, V., Santarelli, F., Forti, L., Spina, S.: Automatic classification of text complexity. Appl. Sci. **10**(20) (2020). https://doi.org/10.3390/app10207285, https://www.mdpi.com/2076-3417/10/20/7285
29. Santucci, V., Spina, S., Milani, A., Biondi, G., Di Bari, G.: Detecting hate speech for Italian language in social media. In: EVALITA 2018, Co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), vol. 2263 (2018)
30. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 298–307 (2015)
31. Smith, E.A., Kincaid, J.P.: Derivation and validation of the automated readability index for use with technical materials. Hum. Factors **12**(5), 457–564 (1970). https://doi.org/10.1177/001872087001200505
32. Stroube, B.: Literary freedom: project Gutenberg. XRDS: Crossroads, ACM Mag. Students **10**(1), 3–3 (2003)
33. Yeoh, J.M., Caraffini, F., Homapour, E., Santucci, V., Milani, A.: A clustering system for dynamic data streams based on metaheuristic optimisation. Mathematics **7**(12), 1229 (2019)