Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Telles Werner

A fast and space-economical approach to Word Mover's Distance

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Eduardo Sany Laber

Rio de Janeiro
April 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Telles Werner

A fast and space-economical approach to Word Mover's Distance

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Eduardo Sany Laber
Advisor
Departamento de Informática – PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática – PUC-Rio

Prof. Raúl Pierre Rentería
Departamento de Informática – PUC-Rio

Rio de Janeiro, April 24th, 2019

Matheus Telles Werner

Bachelor's in Computer Engineering (2016) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio).

Bibliographic data

Werner, Matheus Telles

A fast and space-economical approach to Word Mover's Distance / Matheus Telles Werner; advisor: Eduardo Sany Laber. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2019.

v., 55 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Distância entre documentos. 2. Word Embeddings. 3. Word Mover's Distance. I. Laber, Eduardo Sany. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First and foremost, I would like to thanks my advisor Eduardo Sany Laber for his guidance, professionalism and valuable discussions during all this work. To Alexandre Renteria, whose key insight even if unintentionally, helped this work to reach its present state. To the Informatics Department and its professors at PUC-Rio. To all my friends at Galgos, in special to Georges and Luisa. Finally, to my family, for helping and supporting my decision.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Werner, Matheus Telles; Laber, Eduardo Sany (Advisor). **A fast and space-economical approach to Word Mover's Distance**. Rio de Janeiro, 2019. 55p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The Word Mover's Distance (WMD) proposed in Kusner et. al. [ICML,2015] is a distance between documents that takes advantage of semantic relations among words that are captured by their Word Embeddings. This distance proved to be quite effective, obtaining state-of-the-art error rates for classification tasks, but also impracticable for large collections or documents because it needs to compute a transportation problem on a complete bipartite graph for each pair of documents.
By using assumptions, that are supported by empirical properties of the distances between Word Embeddings, we simplify WMD so that we obtain a new distance whose computation requires the solution of a max flow problem in a sparse graph, which can be solved much faster than the transportation problem in a dense graph. Our experiments show that we can obtain a performance gain up to 3 orders of magnitude over WMD while maintaining the same error rates in document classification tasks.

Keywords

Document Distance; Word Embeddings; Word Mover's Distance.

Resumo

Werner, Matheus Telles; Laber, Eduardo Sany. **Uma abordagem rápida e econômica para Word Mover's Distance**. Rio de Janeiro, 2019. 55p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O Word Mover's Distance (WMD) proposto por Kusner et al. [ICML,2015] é uma função de distância entre documentos que se aproveita das relações semânticas entre palavras extraidas por suas Word Embeddings. Essa função de distância se mostrou bastante eficaz, obtendo taxas de erro estado da arte para problemas de classificação, porém ao mesmo tempo inviável para largas coleções ou grandes documentos devido a ser necessário computar um problema de transporte em um grafo bipartido completo para cada par de documentos.

Assumindo algumas hipóteses, que são respaldadas por propriedades empíricas das distâncias entre as Word Embeddings, nós simplificamos o WMD de forma a obter uma nova função de distância o qual requer a solução de um problema de fluxo máximo em um grafo esparço, que pode ser resolvido mais rapidamente do que um problema de transporte em um grafo denso. Nossos experimentos mostram que conseguimos obter ganhos de performance até 3 ordens de magnitude acima do WMD enquanto mantendo as mesmas taxas de erro na tarefa de classificação de documentos.

Palavras-chave

Distância entre documentos; Word Embeddings; Word Mover's Distance.

Table of contents

List of figures

List of tables

Nothing travels faster than light with the possible exception of bad news, which obeys its own special rules.

Douglas Adams, *The Hitchhiker's Guide to the Galaxy.*

1
Introduction

Documents comparison is a fundamental step in a number of applications, such as recommendation, clustering, search, and categorization. In its simplest version, this task consists of computing the distance between a single pair of documents.

The document representation is a crucial factor in the definition of a distance. Arguably, the most employed document representations due to their simplicity and good results are the Bag-of-Words (BOW) and the Term Frequency - Inverse Document Frequency (TFIDF). These representations are based on word counting and so may lose information that is relevant for some applications, as the ordering among words in a document, co-occurrence and semantic relations between different words. Therefore, richer representations that take into account some of this information have been proposed [1, 2, 3].

Up to a few years ago, no representation used semantic relations because there was no clear methodology of how to obtain them. Though they eventually started using ontologies [4] as a palliative but making them dependent on an external knowledge base. Another concern of that approach was that finding such bases was also impossible for many languages and domains. However, this restricted scenario changed with the emergence of Word2Vec [5, 6] and its variants [7], a class of methods that allow to efficiently identify the relationship between words and embed them into vectors, called word embeddings. As a result, researchers have been looking for ways to combine these embeddings with methods already proposed in the literature to refine them. Results of these efforts can already be seen in works such as [8, 9, 10, 11] and indeed improvements are obtained.

In particular, Kusner et al. [9] proposes the Word Mover's Distance (WMD), an application of the classic Earth Mover's Distance (EMD) [12] for the domain of documents, that takes advantage of the semantic relations captured by the embeddings associated with their words. The idea is to compute the minimum cost required to transform one document representation into another by using the distance between embeddings as the cost of transforming words. In fact, the distance is given by the cost of an optimal solution of a transportation problem defined on a bipartite complete graph where the

nodes correspond to the distinct words of the documents. In the same paper, they show that this approach obtained very good results on document classification tasks, outperforming a number of alternatives. The major drawback of WMD, however, is its high computational cost since solving the transportation problem in a complete bipartite graph is costly, requiring super cubic time.

Motivated by this scenario, the focus of our work is to develop a new distance function that is as effective as WMD with the advantage of being much more efficient with regards to computational resources consumption.

1.1 Our Contributions

To achieve this goal, in contrast to other approaches available, we explore properties of the application domain, more specifically the distribution of distances among word embeddings. Our starting point is showing that instead of considering all the n^2 distances between the embeddings of a vocabulary with n words, we can focus on a much smaller set. This set consists of the distances between *related words*, that is, words that are close in the embedding space, for a suitable definition of closeness. This observation, supported by empirical data, can be used to dramatically reduce the memory required to cache the distances between embeddings. Such cache is essential for the fast computation of WMD and related distances as we show in our experiments. Moreover, the space savings is highly desirable for handling large vocabularies.

Our main contribution, which is also built upon the previous observation, is a new distance function, namely Max Flow Word Mover's Distance (MF-WMD), whose computation relies on the solution of a max flow problem on a sparse graph, a problem that can be solved much faster than the transportation problem in a dense graph. The key idea for this reduction is the observation, perhaps surprisingly, that we can consider only two distinct values to represent the distances between words from a pair of documents, without losing much information: One representing the related words while the other representing the unrelated ones. The selection of related words can be defined via a global parameter that allows to trade-off speed and accuracy. Experiments reported over 8 datasets show that MF-WMD yields test error as good as WMD for document classification task with a significant gain in terms of execution time. Moreover, it is competitive with variants of the WMD such as Relaxed Word Mover's Distance (RWMD) [9] while consuming significantly less space.

As the reduction from all distances to two distances can be seen as too aggressive, we also propose variants of the WMD and RWMD grounded on those same assumptions, although only replacing the distances among

unrelated words by a single distance. Reproducing the previous experiments show that these variants achieve similar gains to the MF-WMD.

In addition, to contributing to the state-of-the-art of the topic under discussion, we believe that the approach taken here could be useful to optimize algorithms for other applications (for example, clustering) that involve the computation of distances between embeddings.

1.2 Related Work

Our work is closely related with some approaches that have been proposed to circumvent the high computational cost of WMD [9, 13].

Kusner et al. [9] proposes the RWMD, a distance that is defined over a relaxation of the transportation problem in which some constraints are dropped. Given the matrix distance between the words embeddings of documents D and D', the RWMD can be calculated in $O(|D| \times |D'|)$ time, where $|D|$ and $|D'|$ are the number of distinct words of D and D', respectively. Thus, the bottleneck of RWMD is to compute the distance matrix which costs $O(|D| \times |D'| \times d)$ time, where d is the dimension of the word embeddings space. Such cost can be prevented by caching the $O(n^2)$ distances between all the n vocabulary words, an approach that could be prohibitive for large n. Experiments from Kusner et al. [9] shows that RWMD achieves test error competitive with WMD for document classification tasks while incurring a lower computational cost, even without using cache.

Atasu et al. [13] shows how to compute RWMD for any two documents D and D' from a collection $\mathcal{C}$ in $O(|D| + |D'|)$ time. To achieve this running time they need to pre-compute and store the distance of word w to the nearest word in document D, for each w in the vocabulary and each D in the collection. Thus, it consumes $O(n|\mathcal{C}|)$ memory, where $|\mathcal{C}|$ is the number of documents in $\mathcal{C}$, which may be infeasible for large collections. Furthermore, this linear time complexity does not hold for dynamic collections since the method requires $O(|D^{new}| \times n \times d)$ for computing the distance matrix before calculating the RWMD from a new document D^{new} to some document D.

The main advantage of MF-WMD over RWMD is that it obtains competitive results in terms of both test error and computational performance while requiring significantly less memory, which makes it more suitable to handle large vocabularies/collections as well as dynamic collections.

Our work is also related with some proposals to speed up EMD [14, 15]. Pele et al. [14] presents an optimized solution of the EMD for instances in which the costs of the edges satisfies some properties that are motivated by the

way human perceive distances.The optimization introduced by this approach consists of reducing the number of edges in the transportation network and, as a consequence, the running time. This works resembles ours in the sense that both optimize the time complexity to solve the transportation problem by taking into account how the costs behave in the domains under consideration. In contrast to Pele et al. [14], our new distance relies on the computation of the max flow problem, which can be solved much faster than the transportation problem as aforementioned.

Cuturi et al. [15] uses an entropic regularization term to smooth out the transportation problem so that it can be solved much faster via Sinkhorn's matrix scaling algorithm. This algorithm has $O(|D| \times |D'|)$ empirical time according to [15] and it was used in a supervised version of WMD [16]. However, similar to the RWMD, this method needs an $O(n^2)$ space cache in order to prevent the $O(|D| \times |D'| \times d)$ time required to compute the distances between the words in D and D'. If not possible, then in reality the running time of this method is $O(|D| \times |D'| \times d)$ instead of $O(|D| \times |D'|)$.

1.3 Dissertation Organization

The dissertation is organized as follows. In Chapter 2, we introduce our notation and discuss some background that is important to the understanding of our work. In the next chapter we develop our approach. In Chapter 4, we present our experimental study comparing the new distance function with WMD and RWMD both in terms of test error in classification tasks as well as in terms of computational performance. Finally, Chapter 5 is regarded to our final remark and discussion of future work.

2
Background

In this chapter, we overview the concepts and methods needed to understand the rest of this work. In Section 2.1, we briefly describe how we can obtain the Word Embeddings and some of their nice properties that are exploited by many of the distance functions used here. Next, in Section 2.2, we discuss the documents representations employed in this work. In Section 2.3, we formally introduce the Maximum flow problem and Minimum cost flow problem that, as already mentioned, are required to explain the distances described in Section 2.4 and to develop our approaches in Chapter 3.

2.1
Word Embeddings

2.1.1
Neural Network Language Model

Bengio et al. [17] introduced the key idea of how to represent words as continuous vectors using Neural Networks during its presentation of a Neural Network Language Model (NNLM). The objective of a language model is to compute the probability $p(w_i|w_1, \ldots, w_m)$ of each word w_i appearing right after a sequence of m words $w_1, \ldots, w_m$. Figure 2.1 displays an example with $m = 2$.

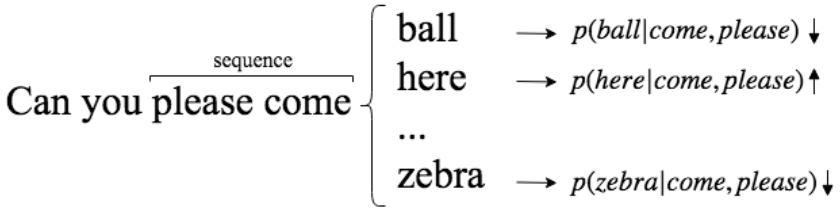

Figure 2.1: Language Model example.

The state-of-the-art algorithm at the time, the n-gram, suffered from the lack of generalization because it could only compute the probability distribution of sequences already seen during training. To overcome this limitation, Bengio et al. [17] suggested a Neural Network model that learns how the words in the sequence relate to the next word being predicted and

how their positions disturb its probability distribution. This new approach surpassed the previous state-of-the-art results, but also unfold an issue related to the computational cost required for training such complex model.

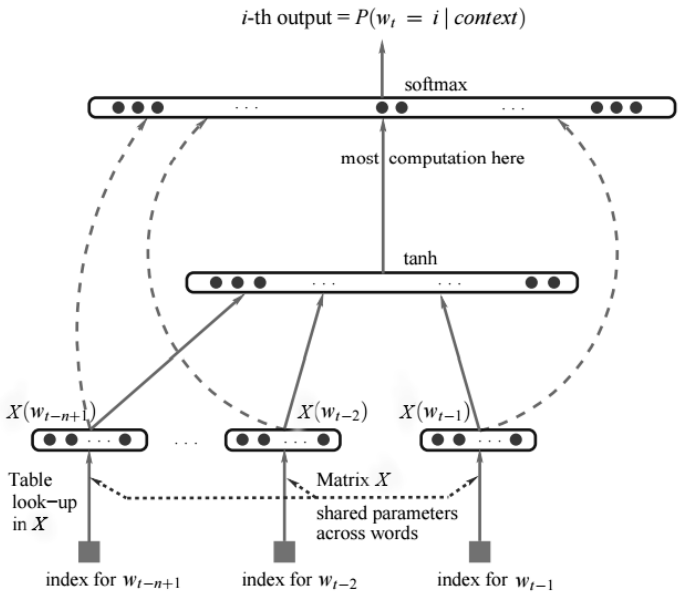

Figure 2.2: Neural Network Language Model.

The NNLM architecture (Figure 2) consists of three components denominated layers: The first layer (bottom) is responsible for getting the embedding of each word w in the input sequence $\{w_1, \ldots, w_m\}$. Next, the second layer (middle) uses them to infer more meaningful features to the task. Then, the third layer (top) consumes these new features to estimate the probability $p(w_i|w_1, \ldots, w_m)$ of each word i of the vocabulary be next in the sequence. Lastly, the model assigns the word with the highest probability as the following word.

For tuning this model, we follow the supervised learning strategy. We give the model a collection $\mathcal{C}$ of annotated sequences. And, for each one of them, the model predicts the next word and checks whether it got it right. If yes, it goes to the next sequence. Otherwise, we update the layers guided by the gradient of the error. We repeat this procedure a few times through the collection, commonly until convergence to a plateau or time limit.

The high computational cost as to do with the obligation of computing the probability of all n words for each sequence, which makes the model costs $\Omega(E \times |\mathcal{C}| \times h \times n)$, where E is how many times we go through them, $|\mathcal{C}|$ is the number of sequences in the collection $\mathcal{C}$ and h is the number of features inferred by the second layer.

2.1.2 Word2Vec

The NNLM can be seen as an algorithm that solves two tasks at the same time. The first task is to learn how to represent words into a vector space and the second task is to build a generic language model using them as features. Due to this clear division, follow-up studies [5, 6, 18] attempted to attack them separately.

In special, Milokov et al. [5, 6] focused solely on developing an efficient procedure for generating, in an unsupervised fashion, a vector space that captures the semantic relations among words since its usually assumed to be a quite time-consuming task.

Inspired by the ideas laid by Bengio et al. [17], Mikolov et al. [5] assumes that semantically close words should frequently appear within a context window of size c of each other which corresponds to the c nearest words before and after a central word w_t in the sentence. Figure 2.3 displays an example with $c = 2$ where "jumps" is considered semantically similar to "brown", "fox", "over" and "the".

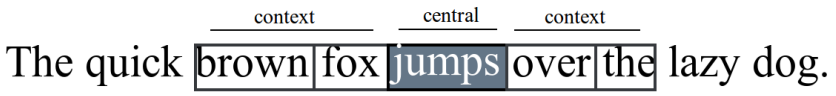

Figure 2.3: Context window example.

Mikolov repeats this process for all existing sentences and contexts windows inside them for each document in a document collection, resulting in an extensive list of similar words as displayed in Figure 2.4. Then he uses this list as input to a shallow Neural Network for learning how to represent each word as a vector, its *Word Embedding*, in the new vector space.

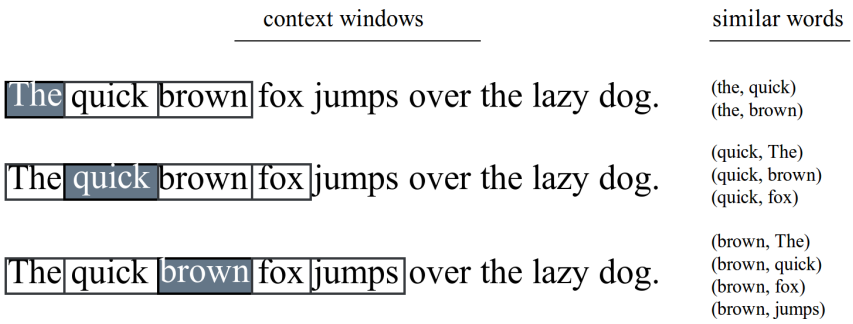

Figure 2.4: Word2Vec training samples.

Mikolov et al. [5] proposes two versions of this shallow Neural Network. The first model called Continuous Bag-Of-Words (CBOW) uses the context words $w_{t-c}, \ldots, w_{t-1}$ and $w_{t+1}, \ldots, w_{t+c}$ to predict the central word w_t while the second model called Skip-Gram do the opposite, it uses the central word w_t to predict the context words $w_{t-c}, \ldots, w_{t-1}$ and $w_{t+1}, \ldots, w_{t+c}$ individually. Figure 2.5 displays a plain version of both models.

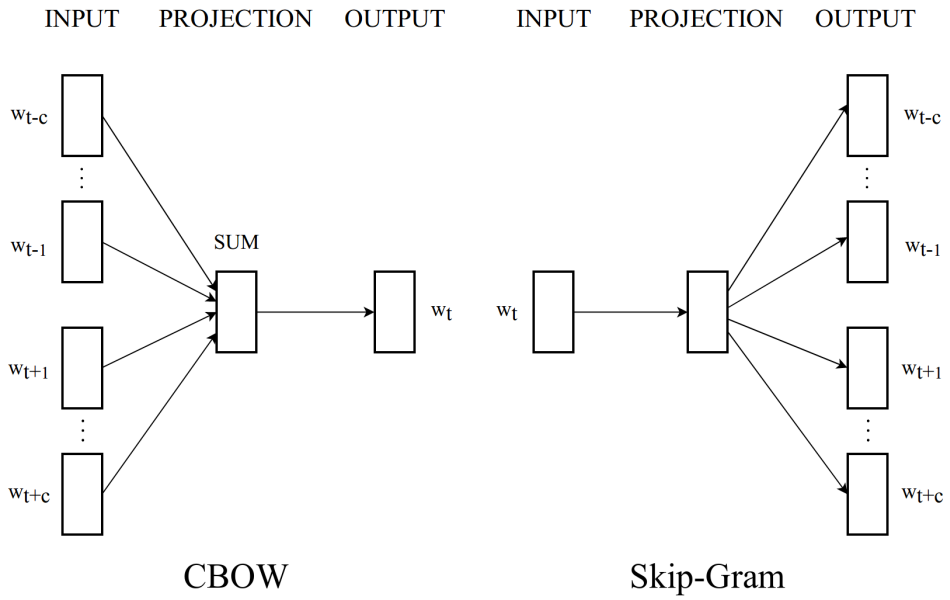

Figure 2.5: Word2Vec Models.

In both architectures each word i in the vocabulary is represented by vectors $x_i^I \in X^I$ and $x_i^O \in X^O$, where the matrices $X^I, X^O \in \mathbb{R}^{d \times n}$ store the d-dimensional word embedding of each one of the n words. The former encodes word i between input and projection layers while the later encodes it between projection and output layers. At the end of the training phase, the word embedding for word i is merely $x_i = x_i^I + x_i^O$. This double representation serves to symbolize both central and context word versions of the word i.

For the CBOW, the first layer (input) gets the embeddings in X^I of the context words $w_{t-c}, \ldots, w_{t-1}$ and $w_{t+1}, \ldots, w_{t+c}$. The second layer (projection) sum these $2c$ vectors and create an aggregated word vector. Next, the third layer (output) takes the dot product between the resultant "artificial" word from previous layer and the vector x_i^O of each word i in the vocabulary and then compute the probability of each one using the softmax function. Finally, similarly to the NNLM, the model assigns the word with the highest probability as the predicted word. The computational cost of one training example is then $O(d \times n)$.

For the Skip-Gram, the first layer gets the word vector $x^I_{w_t}$ and the second layer just copies it. Then the third layer do the same computation done by the CBOW. However, this time we repeat the same process $2c$ times, predicting each context word separately. The computational cost of one training example is then $O(c \times d \times n)$.

Although the Skip-Gram is more computationally expensive, it performs better in external applications and so is more used in practice. Nonetheless, both are still expensive since training one example continues to be proportional to the number of words in the vocabulary. As a solution, Mikolov et al. [6] proposed applying a Noise Contrastive Estimation (NCE) which approximately maximizes the log probability of the softmax but replaces the n by a k factor where $n \gg k$. The idea is to slightly change the goal from "perfectly" learning the probability of each pair of words to learning how to distinguish a real pair of similar words from k fake ones drawn from the noise distribution $P_n(w)$.

Together with this gain in efficiency, the Word2Vec gained general attention because of the nice properties of the generated vector space. Word vectors that are close in the space tend to be semantically close even if they do not co-occur frequently in the documents, while distant vectors, in general, have no semantic relation. Furthermore, they also noticed that in the vector space one can find interesting relations like $vec(German) + vec(airlines) \approx vec(airline\ Lufthansa)$ and $vec(king) - vec(man) + vec(woman) \approx vec(queen)$. Figure 2.6 displays one of the relations found by Mikolov et al. [6].

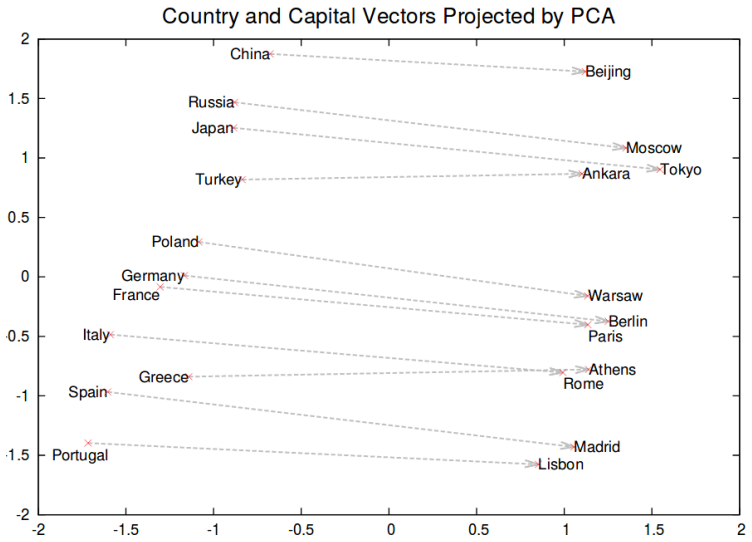

Figure 2.6: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities.

2.2 Document representations

When working with documents, the first step is to establish a representation of them that can be easily manipulated by computers. Ultimately, this representation is a simplification of the original that stores its relevant information in a data structure, such as a vector or dictionary.

For that, we need to define a vocabulary determining all tokens – unique words and phrases – that are going to be taken into account for the representation. Usually the lexicon is just all words present in a given collection of documents, however, it is a well-established practice in the field to apply a preprocessing step before it to standardize these words. This is useful because helps to reduce the noise in the documents, improving the quality of their representations. In any case, we are not going to enter in details about this step. Just consider that the n tokens identified after it composes the vocabulary.

2.2.1 Bag-of-Words

The Bag-of-Words (BOW) is a simple representation of a document where only the frequencies of the words in the documents are taken into account. This representation disregards both semantic and order among words. Formally, the BOW stores in a feature vector the frequency of each word w of the vocabulary in document D.

For example,

D: Obama speaks to the media in Illinois

D': The President greets the press in Chicago

after the preprocessing step results in the following BOW representations:

	D	D'
Obama	1	0
speaks	1	0
media	1	0
Illinois	1	0
President	0	1
greets	0	1
press	0	1
Chicago	0	1

Due to the number of words in a document is much smaller than the number of words in the vocabulary, it is a common practice to employ sparse

vectors, instead of dense ones, for saving space.

Arguably, the BOW is the oldest and most fundamental document representation. Many other document representations derived from it, such as LSI[2] and LDA[3].

2.2.2 Enhanced Bag-of-Words

The Enhanced Bag-of-Words is an extension of the BOW. For the best of our knowledge, a formal definition does not exist for it, but, regardless, we informally define it as the inclusion of any additional information from the documents to the BOW representation, such as the word order.

In our case, we are interested on including the Word Embeddings as side information for the BOW which erases one of the known weakness of that representation. However, this inclusion imposes an update on the document representation. Figure 2.7 displays the updated version of the examples of the Section 2.2.1.

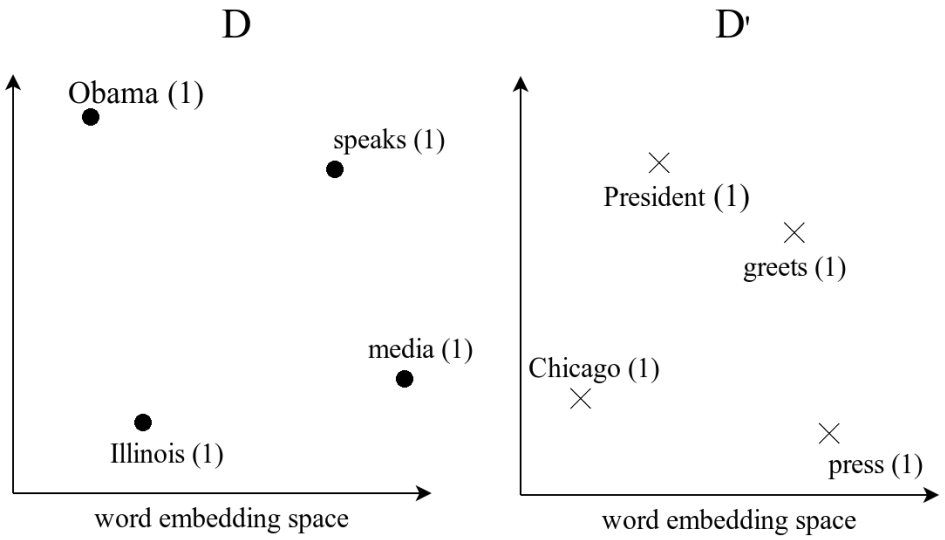

Figure 2.7: Examples of Enhanced Bag-of-Words representations. Each document is represented by its collection of words (points) in the word embedding space weighted by their respective frequencies (between parentheses).

Again, for the sake of space efficiency, all Word Embeddings are stored together in a Matrix $X \in \mathbb{R}^{d \times n}$. The vector $x_i \in \mathbb{R}^d$, corresponding to the i-th column of X, represents the vector of word i of the vocabulary in dimension d. When a document needs to access one of its embeddings, it does it through that matrix. This way, we avoid replicating the embeddings throughout the representations.

2.3 Network flow problems

Network flow problems are a class of computational problems which the input is some flow network $G = (V, E)$, where V is the set of nodes in the graph and E is the set of edges with numerical properties (for example, costs and capacities), and the goal is to construct a flow F between these nodes that optimizes a given objective function subject to a set of constraints.

2.3.1 Maximum flow problem

In this context, the Maximum flow problem is a problem whose goal is to find the maximum amount of flow that can go from a source s to a sink t without exceeding the flow capacity D_{ij} of each edge $e_{ij} \in E$. Formally, the Maximum flow problem follows the Linear programming formulation below:

$$\max \sum_{j|(s,j)\in E} f_{s,j} \tag{2-1}$$

$$\text{s.t.:} \sum_{j|(i,j)\in E} f_{i,j} = \sum_{j|(j,i)\in E} f_{j,i} \qquad \forall i \in V - \{s,t\} \tag{2-2}$$

$$0 \leq f_{i,j} \leq D_{ij} \qquad \forall (i,j) \in E \tag{2-3}$$

The literature provides a vast number of methods for solving this formulation. One of the most commonly used is the Push-relabel algorithm proposed by Golberg et al. [19] that costs $O(|V|^2 \times |E|)$, while the fastest algorithm developed, for the best of our knowledge, was proposed by Orlin et al. [20] and costs $O(|V| \times |E|)$.

2.3.2 Minimum-cost flow problem

The Minimum-cost flow problem is similar to the Maximum flow problem with 2 twists: The amount of flow F is already known from the start; Each edge $e_{i,j} \in E$ begins to charge $c_{i,j}$ per unit flow passing by. Thereby, the idea is to find the cheapest way possible of sending this amount F through the flow network. Formally, the Minimum-cost flow problem follows the Linear programming formulation below:

$$\min \sum_{(i,j)\in E} c_{i,j} f_{i,j} \tag{2-4}$$

$$\text{s.t.:} \sum_{i|(s,i)\in E} f_{s,i} = F \tag{2-5}$$

$$\sum_{j|(j,t)\in E} f_{j,t} = F \tag{2-6}$$

$$\sum_{j|(i,j)\in E} f_{i,j} = \sum_{j|(j,i)\in E} f_{j,i} \qquad \forall i \in V - \{s,t\} \tag{2-7}$$

$$0 \leq f_{i,j} \leq D_{ij} \qquad \forall (i,j) \in E \tag{2-8}$$

$$\tag{2-9}$$

Once again, the literature provides multiple methods for also solving this formulation. However, this time, the fastest algorithm changes depending on the characteristics of the network flow. Even so, arguably, the most commonly used is the cost-scaling push-relabel algorithm proposed by Golberg et al. [21] that costs $O(|V|^2 \times |E| \times log(|V| \times C))$ where C is the value of the largest edge cost in the graph.

2.4 Document distances

One of the by-products of the document representation is the ability of computing distance between representations, which is an elementary operation of any application that needs to compare documents. Naturally, these distance functions are directly dependent of the representation chosen. Thus, we describe next distance functions that are compatible with the representations presented in Section 2.2.

In this section we assume that we have a vocabulary of n words $\{1, \ldots, n\}$ and a collection of documents $\mathcal{C}$. For a document D, let $|D|$ be the number of its distinct words. To explain how to define the distance between two documents D and D' we assume that the set of distinct words of D and D' are, respectively, $\{w_1, \ldots, w_{|D|}\}$ and $\{w'_1, \ldots, w'_{|D'|}\}$. Moreover, we assume that D and D' are represented as normalized sparse Bag-of-Words so that $D = (D_1, \ldots, D_{|D|})$ and $D' = (D'_1, \ldots, D'_{|D'|})$, with $\sum_i D_i = \sum_i D'_i = 1$, where D_i and D'_i are, respectively, the normalized frequency of w_i and w'_i in documents D and D'.

As an example, reusing the two documents of Section 2.2 as our document collection, we have vocabulary

$$\begin{array}{llll} \{ & 1: \text{“Obama”}, & 2: \text{“speaks”}, 3: \text{“media”}, & 4: \text{“Illinois”}, \\ & 5: \text{“President”}, & 6: \text{“greets”}, 7: \text{“press”}, & 8: \text{“Chicago”} \quad \} \end{array}$$

with documents D and D' containing the set of words

$$\{w_1, w_2, w_3, w_4\} = \{1, 2, 3, 4\}$$

and

$$\{w'_1, w'_2, w'_3, w'_4\} = \{5, 6, 7, 8\}$$

respectively. Their respective sparse Bag-of-Words representations are

$$\begin{aligned} D\text{:} \quad & \{D_1, D_2, D_3, D_4\} = \{1, 1, 1, 1\} \\ D'\text{:} \quad & \{D'_1, D'_2, D'_3, D'_4\} = \{1, 1, 1, 1\} \end{aligned}$$

while their normalized versions are

$$\begin{aligned} D\text{:} \quad & \{D_1, D_2, D_3, D_4\} = \{0.25, 0.25, 0.25, 0.25\} \\ D'\text{:} \quad & \{D'_1, D'_2, D'_3, D'_4\} = \{0.25, 0.25, 0.25, 0.25\}. \end{aligned}$$

Furthermore, we assume that the words are represented by its embeddings in a vector space of dimension d and we use $c(i, j)$ to denote the euclidean distance between the embeddings of words w_i and w'_j.

2.4.1 Cosine Distance

The Cosine distance is a widely used distance function in practice that is the complement of the cosine of the angle between two vectors. Arguably, its popularity is due to simplicity, reasonable results and the fact that the "accuracy" of a distance is more associated with the document representation chosen than the distance function itself. Formally, the Cosine distance between documents D and D' is defined as:

$$COSINE(D, D') = 1 - \frac{D \cdot D'}{||D|| \times ||D'||}$$

It is also interesting to recall that the Cosine Distance and the Euclidean distance are equivalent for ranking applications when using unit vectors.

2.4.2 Word Mover's Distance

In the traditional document distance computation, documents are vectors encoded with their BOW representations or derived representations of them that use together some pattern detection technique in the documents collection such as the Latent Dirichlet Allocation (LDA) [3] or Latent Semantic Indexing (LSI) [2]. However, as discussed earlier, it could be argued either way that they

could still be further improved if considering other relevant information, among them the relations between words. Thus, a possible solution is replacing the usual BOW by its enhanced version discussed in Subsection 2.2.2. Still, this raises the question of how to compute distances using them (as displayed in Figure 2.8) since we are only used to work with vectors.

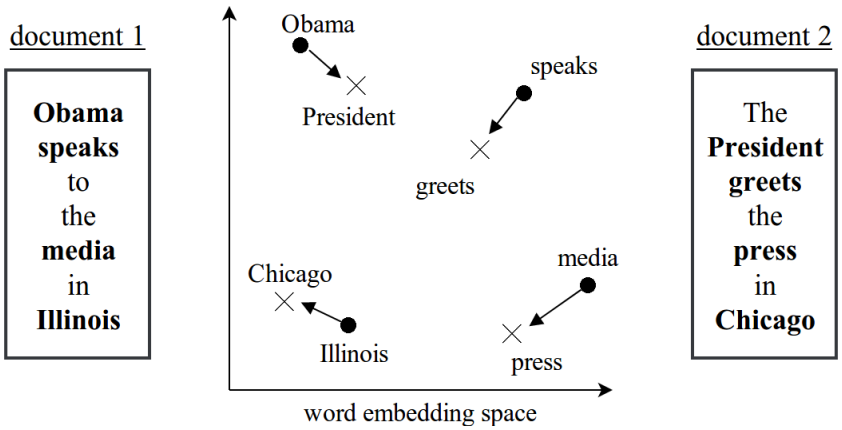

Figure 2.8: Pair of documents in word embedding space.

Eventually, Kusner et al. [9] was able to propose a distance, namely Word Mover's Distance (WMD), employing these representations based on the work of Rubner et al. [12] in which they define a metric for measuring the distance between probability distributions over a region. In our case, the idea is to compute the minimum cost required to transform the words of one document D into the words of another document D', where the cost $c(i, j)$ of transforming the word w_i into word w'_j is given by the distance between the word embeddings of w_i and w'_j and limited by their representations D_i and D'_j of documents D and D'. Figure 2.9 visually expresses this strategy.

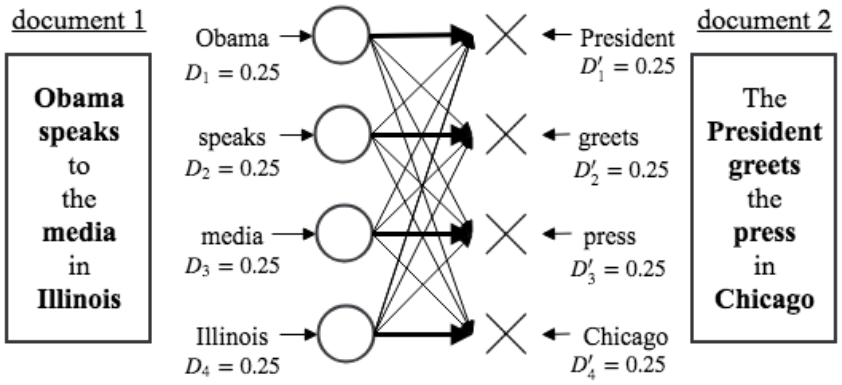

Figure 2.9: Distance between all pair of words from a pair of documents.

Formally, the WMD between two documents D and D' is defined as the value of the optimal solution of the following transportation problem:

$$\min \sum_{i=1}^{|D|} \sum_{j=1}^{|D'|} c(i,j) T_{i,j} \tag{2-10}$$

$$\text{s.t.:} \sum_{j=1}^{|D'|} T_{i,j} = D_i \qquad \forall i \in \{1, \ldots, |D|\} \tag{2-11}$$

$$\sum_{i=1}^{|D|} T_{i,j} = D'_j \qquad \forall j \in \{1, \ldots, |D'|\} \tag{2-12}$$

$$T_{i,j} \geq 0 \qquad \text{for all} \quad i,j \tag{2-13}$$

In the above formulation $T_{i,j}$ is the flow between w_i and w_j.

The WMD, although well defined, suffers from efficiency problems since solving the transportation problem on a complete bipartite graph is costly, requiring super cubic time using the best-known minimum cost flow algorithms or linear programming solvers [9].

2.4.3 Relaxed Word Mover's Distance

To overcome the high computational cost of solving the transportation problem, Kusner et al. [9] proposed the Relaxed Word Mover's Distance (RWMD), a variation of WMD in which relaxations of the transportation problem are solved in order to obtain the distance between two documents. These relaxations are obtained by either ignoring the set of constraints (2-11) or the set (2-12). Let ℓ_1 and ℓ_2 be, respectively, the optimum values of the relaxed problems that are obtained by ignoring constraints (2-11) and (2-12). The RWMD between documents is defined as the maximum between $\ell_1$and ℓ_2.

The advantage of RWMD is that, given the distance matrix, the optimal solution of the relaxation can be found in $O(|D| \times |D'|)$ time, a significant improvement over WMD in terms of computational efficiency. In fact, if we drop the constraints (2-11), the optimal flow T^* works as follows: $T^*_{i,j} = D'_j$ if $c_{i,j}$ is the minimum cost of an edge that reaches j and $T^*_{i,j} = 0$, otherwise. When the constraints (2-12) are dropped the optimal flow is defined analogously.

Therefore, the bottleneck of RWMD is the computation of the distance matrix between the words of D and D' which can be prevented by using a possibly huge cache.

3
An efficient method for calculating the distance between documents via word embeddings

WMD presents two sources of complexity: (i) the cost of calculating the distances between the $|D| \times |D'|$ pairs of words from documents D and D' and (ii) the cost of solving a minimum cost flow problem.

In this chapter we argue that we can avoid the $|D| \times |D'|$ distance evaluations and we can replace the minimum cost flow problem with a maximum flow problem, which yields a significant gain in terms of computational speed.

3.1
On the distances between word embeddings

In this section, we discuss the assumptions in which our new distance function relies on. For that, we present examples of distances between words that employ word embeddings. All of them, used here and in the next chapters, were made available by Google [1]. They trained the vectors with $d = 300$ using the Word2Vec template of Word Embeddings [6] on top of a Google News document base containing altogether about 100 billion words and 3 million tokens. We also refer to some datasets that will be detailed in Chapter 4.

From a semantic perspective, it is reasonable to consider that, in general, words are closely related to only a few other words. As the word embeddings were designed to simulate semantic relations, it is expected that they present a similar behavior, that is, each vector should be close to a few other vectors and far away from the remaining ones.

As an example, if the words are ranked according to their distances to the embedding corresponding to "cat" one should expect "dog" and "rabbit" preceding both "moon" and "guitar". However, it is not clear whether "moon" or "guitar" comes first in the ranking since neither of them has an obvious relation with "cat". Figure 3.1 illustrates this behaviour by displaying the distances between the embedding for "cat" and the embeddings from the words of Amazon dataset (described in Section 4.1) sorted by increasing order of distance. We note that there are few words with small distance while the vast majority has distance concentrated in the range $[1.2, 1.4]$.

[1] https://code.google.com/archive/p/word2vec/

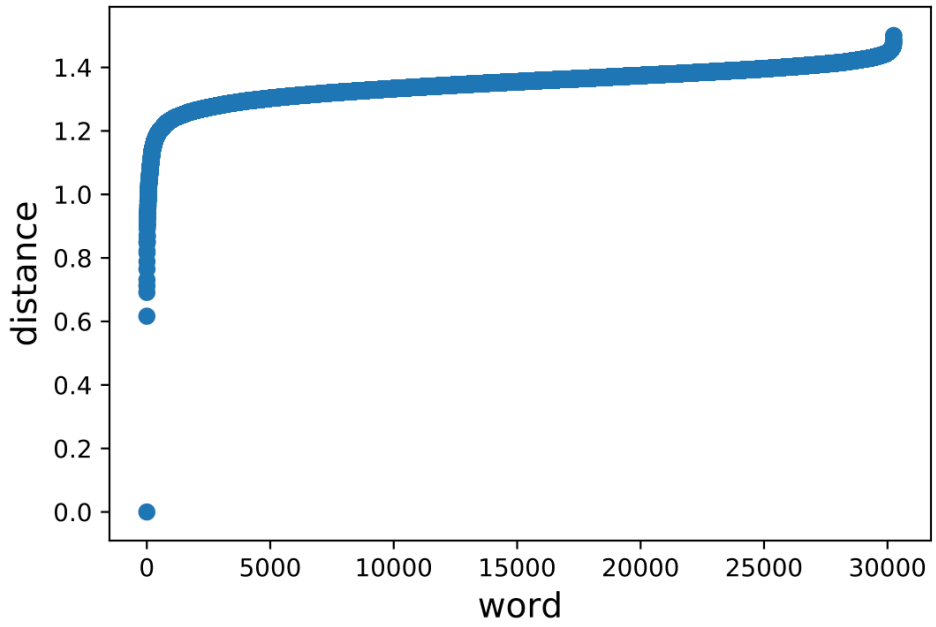

Figure 3.1: Distances from the embeddings of the words in Amazon dataset to word "cat".

For checking whether this behavior repeats in general, we sampled 2000 pairs of documents containing at least 50 tokens from the Reuters dataset (described in Section 4.1) and computed the distance between the embeddings of each document pair. Figure 3.2 shows the distribution of these distances clustered in bins for better visualization. Once again, we observe a high concentration of the distances around the interval $[1.2, 1.4]$, behaving similarly to a Normal distribution.

Based on this discussion, we make the following assumptions:

(a) Given a word w, the remaining words can be split into two groups: $\texttt{RELATED}(w)$ and $\texttt{UNRELATED}(w)$, with the former (latter) containing the words related (unrelated) with w;

(b) The distances from every word in $\texttt{UNRELATED}(w)$ to w, for every w, is a same "large" value c_{max}.

For (a), we assume that given the behavior displayed in Figure 3.1 the related words with respect to each word should be the ones with distance smaller than the 'elbow point' distance in the graph while the unrelated ones should have distances greater than it. Additionally, as already mentioned before and displayed in Figure 3.2, this unrelated group seems to behave like a Normal

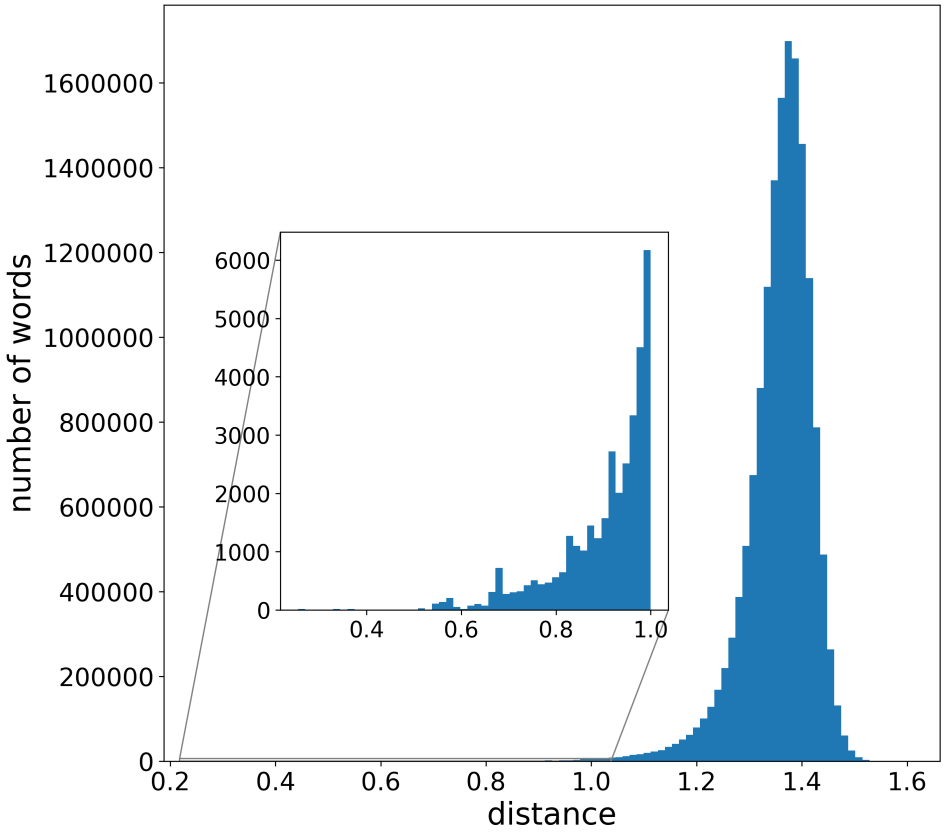

Figure 3.2: Distribution of the distances between words from 2000 pairs of documents.

distribution, concentrating most of its distances around a single value, which suggests that replacing them by its average should have little impact as implied by (b).

To define which words lie in each group we use a parameter $r(w)$ so that the $r(w)$ closest words to w with respect to the distance of their embeddings lie in $\texttt{RELATED}(w)$ and the remaining ones lie in $\texttt{UNRELATED}(w)$. In Chapter 4 we discuss how to set $r(w)$.

3.2 Algorithms exploiting distance assumptions

An algorithm with the flavor of WMD can benefit from our assumptions as follows: first, the algorithm could include a preprocessing step to cache the distance from w to its $r(w)$ closest words for each word w in the vocabulary. By doing so the algorithm prevents calculating the distance between the same pair of words more than once. Note that this approach, without our assumption, could be infeasible due to the large size of the cache (e.g. for the entire Google collection containing 3M words we would need dozens of Petabytes). In

addition, as we explain further, the algorithm also benefits from the assumption that all large distances are equal to c_{max} since it allows to significantly reduce the number of edges in the flow problem.

In the next subsections we describe three algorithms that diminish the WMD computation by relying on these assumptions. Apart from providing an expected speed up with respect to the WMD, these algorithms do not lose in terms of accuracy, as we show in Chapter 4.

We follow the same set up laid out on Section 2.4, where we assume that every document D consists of a set of words $\{w_1, \ldots, w_{|D|}\}$ and is represented as a sparse normalized BOW so that $D = \{D_1, \ldots, D_{|D|}\}$, with $\sum_i D_i = 1$.

3.2.1 Preprocessing Phase

Formally, we are given a matrix of word embeddings $X \in \mathbb{R}^{d \times n}$ for a vocabulary of n words. The vector x_i, corresponding to the ith column of X, represents the embedding of word i in dimension d.

Recall that $r(w)$ is defined as the number of words related to w. The preprocessing phase works as follows. For each word i, we compute its Euclidean distance $\|x_i - x_j\|_2$ to every other word j and we add (i, j) as well as its distance to a cache C if and only if $\|x_i - x_j\|_2 \leq t_i$, where t_i is the distance between i and its $r(i)$-th closest word. The cache C requires $O(n\bar{r})$ space, where $\bar{r}$ is the average of all $r(i)$. Meanwhile, all distances superior to t_i are accumulated and then average, resulting in the c_{max} value discussed in the previous section.

For efficiently computing the cache C and c_{max}, we implemented this preprocessing phase using a min-heap data structure, limiting its size to $r(w)$. For each word i, we compute its distance to the remaining vocabulary and then push each distance to the heap H_i if its less than the greater distance this heap, otherwise we added it to a global accumulator A. After we repeat this n times, the cache $C = \bigcup_{i=1}^{n} H_i$ and $c_{max} = \sum_{j}^{|A|} \frac{A_j}{|A|}$. The first part of this algorithm costs $O(nd)$ while the second part cost $O(n \log r(w_i))$ per word, thus the total cost of this phase is $O(n^2(d + \log \bar{r}))$.

3.2.2 Related Word Mover's Distance

The Related Word Mover's Distance (Rel-WMD) between D and D' is defined as the optimum value of the transportation problem given by equations (2-10)-(2-13), where the costs of the edges are as follows:

$$c(i,j) = \begin{cases} 0, & \text{if } w_i = w_j \\ \|x_i - x_j\|_2, & \text{if } (w_i, w_j) \in C \\ c_{max}, & \text{otherwise} \end{cases} \tag{3-1}$$

Due to the saturation of many edges to c_{max}, it turns out possible to considerably reduce the number of comparisons between edges during the computation of the minimum cost flow problem. The key idea is that the algorithm only needs to compare the pairs of words contained in C during that computation. As the remaining pairs have the same value c_{max}, it implies that using any of them will lead to the same optimal value as long as they respect the constraints of the problem. Thus, the algorithm can just randomly select any pair, excluding the necessity of having to compare them.

The Rel-WMD formulation follows the formulation proposed by Pele et al. [14] as a faster alternative for the EMD [12] based on the same assumptions described in Section 3.1. Essentially, the solution of the WMD can be infeasible if we only focus on comparing the pairs of words in C. Because of it, we have to include an "external" node t as an alternative path between any pair of words but costing c_{max} — to only employ it when actually needed — which has the effect described in the last paragraph.

$$\min \sum_{i=1}^{|D|} \sum_{j=1}^{|D'|} c(i,j) T_{i,j} + \sum_{i=1}^{|D|} c_{max} T_{i,t} \tag{3-2}$$

$$\text{s.t.:} \sum_{j|(i,j)\in C} T_{i,j} + T_{i,t} \leq D_i \qquad \forall i \in \{1, \ldots, |D|\} \tag{3-3}$$

$$\sum_{i|(i,j)\in C} T_{i,j} + T_{t,j} \leq D'_j \qquad \forall j \in \{1, \ldots, |D'|\} \tag{3-4}$$

$$\sum_{i=1}^{|D|} T_{i,t} - \sum_{j=1}^{|D'|} T_{t,j} = 0 \tag{3-5}$$

$$T_{i,j}, T_{i,t}, T_{t,j} \geq 0 \qquad \text{for all} \quad i, j \tag{3-6}$$

3.2.3 Related Relaxed Word Mover's Distance

The Related Relaxed Word Mover's Distance (Rel-RWMD) is a variation of the Rel-WMD, in which we drop constraints of the original formulation in order to obtain a relaxation that can be computed more efficiently, similarly to WMD and RWMD.

Instead of computing the optimal value of the problem defined by (3-2)-(3-6), we ignore either the set of constraints (3-3) or (3-4) and compute the

optimal value of the relaxation as explained in Subsection 2.4.3. In the event of a node not having any edge, we force it to select c_{max} from the constraint (3-5) as their lowest cost edge.

Due to our assumptions, the computation of the optimal solution for the relaxation decreases from $O(|D| \times |D'|)$ to $O(\bar{r} \times |D| + \bar{r} \times |D'|)$.

3.2.4 Max Flow Word Mover's Distance

Our assumptions so far have allowed us to split the distances between words embeddings into two groups of related and unrelated words and replace all the distances in the latter by a single distance. That raises the question of whether the same is also applicable for the former. If so, we could exploit this new assumption together with the others to increase the computational speed even more.

So now we turn our attention to the related words. On one hand, it is clear that some words in $\texttt{RELATED}(w)$ must be closer to w than others: arguably "kitten" is more related to "cat" than "rabbit" to "cat". On the other hand, it may not be clear whether "feline" or "dog" is closest to "cat". In this case, the answer depends on the context. In addition, it is also not clear whether "cat" is more related to "dog" than "piano" to "guitar".

Taking into account the difficulty of precisely defining a distance between related words and *mainly* motivated by performance issues, we add a new assumption:

(c) When evaluating the distances between documents D and D' all the distances between words $w \in D$ and $w' \in D'$ for $w' \in \texttt{RELATED}(w)$ or $w \in \texttt{RELATED}(w')$ are at the same distance $c_{rel}(D, D')$ that depends on both D and D'.

For two documents D and D' let $c_{rel}(D, D')$ be the average distance of the related words from D and D'. Similarly to the Rel-WMD, The MF-WMD between D and D' is defined as the optimum value of the transportation problem given by equations (2-10)-(2-13), where the costs of the edges are as follows:

$$c(i,j) = \begin{cases} 0, & \text{if } w_i = w_j \\ c_{rel}(D, D'), & \text{if } (w_i, w_j) \in C \\ c_{max}, & \text{otherwise} \end{cases} \tag{3-7}$$

In this case, a key observation is that the transportation problem can be reduced to the maximum flow problem presented below. In this formulation,

s and t are the source and the sink of the network, respectively. Moreover, $E_i = D_i$ if word w_i does not appear in document D' and, otherwise, $E_i = \max\{D_i - D'_j, 0\}$ where w'_j is the word in D' equals to w_i. E'_j is analogously defined for document D'.

$$\max \sum_{i \in D} f_{s,i} \tag{3-8}$$

$$\text{s.t.:} \quad f_{s,i} \leq E_i \qquad i \in \{1, \ldots, |D|\} \tag{3-9}$$

$$f_{j,t} \leq E'_j \qquad j \in \{1, \ldots, |D'|\} \tag{3-10}$$

$$f_{s,i} = \sum_{j|(i,j) \in C} f_{i,j} \qquad i \in \{1, \ldots, |D|\} \tag{3-11}$$

$$\sum_{i|(i,j) \in C} f_{i,j} = f_{j,t} \qquad j \in \{1, \ldots, |D'|\} \tag{3-12}$$

$$f_{s,i}, f_{i,j}, f_{j,t} \geq 0, \qquad \text{for all } i, j \tag{3-13}$$

The next lemma relates the transportation problem with the maximum flow problem.

Lemma 3.1 *The optimal value of the problem defined by (2-10)-(2-13), with edge costs given by equation (3-7), is equal to the optimal solution of the maximum flow problem defined by (3-8)-(3-13)*

Proof. First we note that there exists an optimal solution T^* with $T_{x,y} = min\{D_x, D'_y\}$ for every edge (x, y) with $c(x, y) = 0$. Thus, if $w_x = w_y$, we can fix $T_{x,y} = min\{D_x, D'_y\}$ and then replace the constraints

$$\sum_{j=1}^{|D'|} T_{x,j} = D_x \text{ and } \sum_{i=1}^{|D|} T_{i,y} = D'_y$$

with the constraints

$$\sum_{j \neq y} T_{x,j} = \max\{D_x - D'_y, 0\} \text{ and } \sum_{i \neq x} T_{i,y} = \max\{D'_y - D_x, 0\},$$

respectively.

Thus, the new transportation problem has edges connecting w_i to w_j if and only if $w_i \neq w_j$. Note that the total amount of flow send through the arcs is the same for every feasible solution. Let F be this amount. Since there are only two possibilities for the costs, c_{max} and $c_{rel}(D, D')$, then the cost of a solution that sends F_{cheap} units of flow through arcs of cost $c_{rel}(D, D')$ is

$$F_{cheap} \cdot c_{rel}(D, D') + (F - F_{cheap}) \cdot c_{max}.$$

Thus, an optimal solution is one that sends the maximum amount of flow through arcs of cost $c_{rel}(D, D')$, that is, one that is optimal for the maximum flow problem defined by (3-8)-(3-13) ■

This formulation has the following advantages w.r.t the WMD formulation. First, the number of edges required by the maximum flow formulation is the number of distinct words in the documents plus the number of pairs of related word between the documents. This quantity is in general significantly smaller than $|D| \times |D'|$, the number of edges required by WMD's formulation. The second advantage is that the maximum flow problem in a sparse graph can be solved much faster than the transportation problem in a complete bipartite graph.

Let f^* be a vector where each component corresponds to the flow of an edge in the optimal solution of the maximum flow problem. The MF-WMD between D and D' is defined as

$$c_{rel}(D, D') \sum_{(i,j) \in C} f^*_{i,j} + c_{max} \left(1 - \sum_{(i,j) \in C} f^*_{i,j} \right).$$

4
Experiments

In this chapter we report our experimental study whose main goal is to assess the quality of the Rel-WMD, Rel-RWMD and MF-WMD described in Chapter 3.

Our experimental setting follows Kusner et al. [9], where different distances are evaluated according to their performance when they are employed by the k-nearest neighbors (k-NN) method to address the document classification task.

We run the k-NN using $k = 19$. In case of ties, k is divided by 2 until there are no more ties. This is slightly different from Kusner et al. [9] where the best value of k in the set $\{1, 3, \ldots, 19\}$ is employed. Motivations for using this evaluation approach, based on k-NN, include its reproducibility and simplicity.

The methods were implemented in C++. The Eigen library [1] was used for matrix manipulation and Linear Algebra while the OR-Tools library [2] was used for the resolution of flow problems. All experiments were executed using a single core of an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, with 8 GB of RAM.

The code and the datasets are available in `https://github.com/anonymous-1234/mf-wmd` and `https://www.dropbox.com/sh/9rnx8vwjvwjirsp/AADGwbT-aCqzZG7C_L1uNsY1a?dl=0`, respectively.

4.1
Datasets description

For the document classification task we used the following 8 preprocessed datasets provided by Kusner et al. [9] as well as its partitions, for the purpose of cross validation:

- **20NEWS:** Posts on discussion boards for 20 different topics.
- **AMAZON:** Product reviews from Amazon for 4 product categories.
- **BBCSPORT:** BBC Sport sports section articles for 5 sport between 2004 and 2005.

[1] eigen.tuxfamily.org/index.php
[2] https://developers.google.com/optimization/

- **CLASSIC:** Sentences from academic works from 4 different publishers.
- **OHSUMED:** Medical summaries categorized by different cardiovascular diseases. For computational performance issues, only the first 10 categories of the database were used.
- **RECIPE:** Culinary recipes separated by 15 regions of origin.
- **REUTERS:** News from the Reuters news agency in 1987. The original database contains 90 classes, however, due to problems of imbalance between them, a reduced version with only the 8 most frequent ones was created [22].
- **TWITTER:** Collection of "tweets" labeled by feelings "negative", "positive" and "neutral".

In a preprocessing phase, all documents had their words converted to lowercase or removed if contained in a stopwords list [3] due to their little semantic value. Next, non-alphanumeric characters were removed. Lastly, what remained from the documents was split on the whitespaces, producing a set of unique words that compose their respective vocabularies. For performance reasons, the 20NEWS had all tokens with less than 5 occurrences on the collection dropped, and each document limited to its 500 most frequent tokens. Table 4.1 presents relevant statistics for each of the datasets.

Table 4.1: Datasets statistics.

Name	#Docs	#Vocabulary	Average tokens per doc	Classes
20news	18820	22439	69.3	20
amazon	8000	30249	44.5	4
bbcsport	737	10103	116.5	5
classic	7093	18080	38.6	4
ohsumed	9152	19954	60.2	10
recipe	4370	5225	48.3	15
reuters	7674	15115	36.0	8
twitter	3108	4489	9.9	3

[3] http://www.lextek.com/manuals/onix/stopwords2.html

4.2 Distances

All the distances considered in our experiments make use of the Bag-of-Words (BOW) representation. We concentrate our comparison on the following distance functions:

COSINE: a widely used measure that is a natural baseline for other distances. It is also an alternative to the Euclidean distance, which presented poor results in the experiments reported in Kusner et al. [9].

WMD: it is also a natural competitor since one of the main goals of our work is to achieve quality comparable to WMD spending significantly less computational resources.

RWMD: the method that uses a relaxation of the WMD. Experiments from Kusner et al. [9] suggest that RWMD has accuracy comparable to WMD for the document classification task. In addition, it can be viewed as an alternative to the Sinkhorn Distance [15] since both have a low computational cost.

4.3 Results

We discuss the results with respect to test error, computational performance and memory requirements. As we shall see, we approach each one of these aspects from different angles, covering the critical factors that influence the results of the distance functions.

Due to the number of experiments, we only report in the the body of this work the most meaningful results during the analysis. In any case, the complete list of experiments with their respective results can be seen in the Appendix A.

During all experiments, unless stated otherwise, denote Rel-WMD, Rel-RWMD and MF-WMD with $r(w)$ set to 32, as their test errors had achieved the lowest around that value and there is no reason to use larger values as we will see in the next subsections.

4.3.1 Test Error

In this subsection, we present the behavior of the test error obtained by the distances under consideration over the 8 datasets.

For better readability of all tables, the distance with lowest error is assigned to 1 for each dataset, while the others get values equal to their error over the error achieved by the best distance. Furthermore, for the sake of future

reference, the lowest test errors are presented in an extra column in the table, denominated BEST.

4.3.1.1 By dataset

Table 4.2 compares the test errors obtained by our competitors against the lowest errors obtained on average by our methods with a same $r(w)$ over the described datasets.

First, as expected, there are slightly differences between the values obtained here and those reported in Kusner et al. [9]. One reason is the different usage of the parameter k that is employed to select the number of neighbors for classification.

Some observations are in order: clearly, COSINE is the worst of the distances, a behavior that is not surprising since it does not capture semantic relation between different words. Among WMD and its variants, there is a balance in terms of test error, which collaborates with our assumptions that only a small number of distances are required during the computation of WMD. The variance of the results for BBCSPORT, essentially for WMD and RWMD, should have to do with the small number of samples.

Table 4.2: The behavior of test error for different distances/datasets.

Dataset	BEST	COSINE	WMD	RWMD	REL-WMD	REL-RWMD	MF-WMD
20news	24.03	1.27	1.00	1.03	1.01	1.01	1.00
amazon	6.91	1.87	1.04	1.00	1.14	1.11	1.16
bbcsport	4.36	1.10	1.23	1.19	1.04	1.12	1.00
classic	2.81	2.26	1.08	1.00	1.11	1.11	1.11
ohsumed	40.58	1.13	1.06	1.07	1.00	1.02	1.02
recipe	43.07	1.06	1.08	1.08	1.00	1.00	1.00
reuters	3.84	2.33	1.00	1.06	1.06	1.07	1.04
twitter	28.78	1.11	1.01	1.01	1.01	1.00	1.00
Average	—	1.52	1.06	1.05	1.05	1.06	1.04

4.3.1.2 By changing embeddings

Table 4.3 presents the same experiments of 4.3.1.1, but replacing the Word2Vec [5] with GloVe [7] word embeddings to check the consistence of the test errors through different embeddings. The GloVe embeddings [7] employed were made available by the paper authors at their webpage [4].

[4]https://nlp.stanford.edu/projects/glove/

First, it is important to notice that the 20NEWS dataset is absent from the table. The reason is that, in this specific case, the dataset made available by Kusner only contains the embeddings of each word and their respective frequencies in each document, which makes problematic to test alternative embeddings.

The remaining datasets present a marginal increase in the test errors with respect to the test errors previously obtained. There are two possible explanations for this behavior: First, as we do not have the original documents, the words used in this experiment are a subset of the ones used by the Word2Vec. Second, the quality of the word embeddings varies in accordance with the number of words in its training dataset [5]. The GloVe embeddings used were trained in a dataset containing 6 billion words while the Word2Vec embeddings were trained over 100 billion words. Still, these test errors are close to the original values, implying that all methods should be compatible with different word embeddings.

Table 4.3: The behavior of test error for different distances/datasets, replacing the Word2Vec with GloVe embeddings.

DATASET	BEST	COSINE	WMD	RWMD	REL-WMD	REL-RWMD	MF-WMD
AMAZON	7.14	1.81	1.00	1.01	1.07	1.04	1.12
BBCSPORT	4.36	1.10	1.10	1.00	1.10	1.04	1.15
CLASSIC	3.20	1.98	1.01	1.00	1.04	1.09	1.04
OHSUMED	42.11	1.10	1.00	1.00	1.01	1.01	1.01
RECIPE	43.07	1.07	1.01	1.01	1.00	1.01	1.00
REUTERS	4.29	2.09	1.00	1.05	1.01	1.11	1.03
TWITTER	28.52	1.12	1.01	1.01	1.00	1.00	1.01
AVERAGE	—	1.47	1.02	1.01	1.03	1.04	1.05

4.3.1.3 Sensitivity to the number of related words

Table 4.4 presents the effect of the $r(w)$ parameter over the MF-WMD method, where $r = x$ denotes MF-WMD with $r(w)$ set to x for all words w. The Rel-WMD and Rel-RWMD are omitted because they exhibit similar behavior while the WMD and RWMD are kept for comparison. Furthermore, we only tested $r = \{1, 2, 4, 8, 16, 32, 64, 128\}$ due to the considerable running time required to run each configuration on some datasets.

For the MF-WMD, it becomes clear that the test error gets higher for small values of r, which is somehow expected because small r progressively captures less semantic. It is interesting to mention that even $r = 1$ yields results significantly better than COSINE. Also interesting is the fact that the

error stabilizes around $r = 32$. As small values of r yield to faster executions, there is no motivation to use values of r much larger than 32.

Table 4.4: The behavior of test error of the MF-WMD over different r(w) values.

Dataset	BEST	WMD	RWMD	r=1	r=2	r=4	r=8	r=16	r=32	r=64	r=128
20news	24.03	1.00	1.03	1.05	1.06	1.04	1.03	1.00	1.00	1.01	1.00
amazon	6.91	1.04	1.00	1.43	1.37	1.33	1.24	1.20	1.16	1.11	1.09
bbcsport	4.27	1.26	1.21	1.06	1.15	1.00	1.13	1.00	1.02	1.19	1.26
classic	2.81	1.08	1.00	1.31	1.30	1.26	1.22	1.17	1.11	1.12	1.09
ohsumed	41.43	1.03	1.05	1.02	1.03	1.02	1.01	1.00	1.00	1.01	1.01
recipe	42.97	1.08	1.08	1.00	1.01	1.00	1.01	1.01	1.00	1.00	1.01
reuters	3.84	1.00	1.06	1.37	1.23	1.13	1.12	1.10	1.04	1.06	1.05
twitter	28.76	1.01	1.01	1.11	1.07	1.04	1.02	1.01	1.00	1.00	1.02
average	—	1.06	1.05	1.17	1.15	1.10	1.10	1.06	1.04	1.06	1.06

4.3.2 Computational Performance and Memory Requirements

In this subsection, we present the behavior of the computation performance obtained by the distances under consideration over the 8 datasets.

Because our approaches cache all the k closest words to each word in the vocabulary, it is fair to do the same for the competing distances. In practice, we implement a cache version only for the RWMD because its complexity is dominated by the computation of the distances between the embeddings.

If we chose to cache the distances between all words, it would be required an $O(n^2)$ space in main memory, which is not feasible in general. Thus, we opted for a more reasonable approach that only cache the essential distances for the document being classified at each time. This way, we only cache the distances between the words in the vocabulary and the p words in the current document, reducing the space from $O(n^2)$ to $O(np)$, where $n \gg p$. However, it becomes necessary to recalculate the cache after each new document received.

For better readability of all tables, all running times are relative to that of the WMD since it is the baseline of our comparisons.

4.3.2.1 By dataset

Table 4.5 presents the running times of the experiments on Subsection 4.3.1.1.

Of course, as we can see, the COSINE has the greatest speed up factor with respect to the WMD, gaining up to 3 to 4 orders of magnitude which

is expected since the former is linear while the latter is super cubic. The RWMD obtains similar gains to the ones formerly reported by Kusner et al. [9], however, this gain considerably increases when we apply a cache strategy in it what demonstrates the burdensome of the computation of the distances matrix over the faster approaches. It is interesting to note that the cached version of RWMD is also faster than the Rel-WMD, Rel-RWMD, and MF-WMD. This occurs because all our methods contain a preprocessing step for identifying the pairs of words between D and D' in the `RELATED` set before computing the distance between documents. This step should be negligible for large documents, however, all datasets tested are mostly composed of small documents as displayed in the Table 4.1.

Table 4.5: The speed up factor w.r.t to WMD of different distances/datasets.

Dataset	COSINE	RWMD	RWMD (CACHE)	REL-WMD	REL-RWMD	MF-WMD
20news	2433.2	7.1	120.5	21.9	80.8	43.9
amazon	916.3	5.0	38.6	10.5	35.1	22.0
bbcsport	3972.2	7.2	37.7	10.7	35.9	32.6
classic	878.5	5.0	44.8	8.8	33.3	20.7
ohsumed	1306.7	6.8	57.1	9.9	42.1	27.5
recipe	1006.8	5.0	77.1	5.5	34.1	19.3
reuters	674.1	3.9	38.7	6.2	35.1	18.0
twitter	97.8	3.2	16.5	3.6	12.1	7.7
average	1410.7	5.4	53.9	9.6	38.6	24.0

4.3.2.2 Sensitivity to the number of related words

Analogously to the previous subsection, Table 4.6 presents the running times of the experiments on Subsection 4.3.1.3.

As expected, the running time of the MF-WMD is naturally proportional to value of r. Smaller r are as fast as the RWMD (cache) while larger r slowly begins to approach the running time of the WMD, however there is no motivation to use larger values because, as already stated earlier, the test error stabilizes around $r = 32$.

All running times of the distances displayed are the sum of their preprocessing and classification times. The WMD, RWMD, and RWMD (cache) only have the latter because the computation of the distance matrices is an integral part of the classification since it has to be done for each new document to be classified. However, the MF-WMD is composed of both since it only computes them once and then stores the results. This implies that if a collection is small

and its vocabulary is large, the preprocessing time can represent a significative part of the total running time, covering the gains on the classification time. For example, more than half of the running time of the MF-WMD with $r(w)$ up to 16 on the BBCSPORT - the dataset with the largest documents - is spent on the preprocessing step. Thus, if that collection was larger, the performance gain with small r would have been higher.

Figure 4.6 displays the percentages of the preprocessing time within the total running times of the distinct versions of the MF-WMD run in this subsection.

Table 4.6: The speed up factor w.r.t to WMD of the MF-WMD over different r(w) values.

Dataset	RWMD	RWMD (CACHE)	r=1	r=2	r=4	r=8	r=16	r=32	r=64	r=128
20news	7.1	120.5	126.7	109.4	94.2	77.4	64.9	43.9	36.8	29.9
amazon	5.0	38.6	46.9	42.3	37.1	33.5	27.7	22.0	17.5	14.8
bbcsport	7.2	37.7	60.0	55.2	49.8	48.2	42.0	32.6	24.4	18.2
classic	5.0	44.8	56.8	47.5	41.4	34.3	27.3	20.7	16.7	14.5
ohsumed	6.8	57.1	79.0	67.6	58.8	49.7	38.2	27.5	20.4	17.2
recipe	5.0	77.1	55.9	50.2	41.9	35.3	26.5	19.3	14.2	11.2
reuters	3.9	38.7	43.5	38.4	33.9	28.6	22.4	18.0	15.0	12.4
twitter	3.2	16.5	17.3	14.5	11.4	10.1	8.8	7.7	6.6	6.1
average	5.4	53.9	60.8	53.1	46.1	39.6	32.2	24.0	19.0	15.5

Table 4.7: The percentage of preprocessing time within the running time for the MF-WMD.

Dataset	r=1	r=2	r=4	r=8	r=16	r=32	r=64	r=128
20news	3%	3%	2%	2%	2%	1%	1%	1%
amazon	36%	33%	30%	26%	22%	17%	15%	12%
bbcsport	69%	67%	64%	59%	49%	41%	29%	24%
classic	26%	23%	20%	16%	13%	10%	8%	7%
ohsumed	10%	8%	7%	6%	4%	3%	3%	2%
recipe	4%	4%	3%	3%	2%	1%	1%	1%
reuters	17%	15%	12%	12%	8%	7%	6%	5%
twitter	25%	24%	19%	18%	14%	14%	11%	11%
average	24%	22%	20%	18%	14%	12%	9%	8%

4.3.2.3 By the size of the documents

So far, we have described some limitations on the analysis while comparing the running times of the distances under consideration, being the main

ones: the documents are small which yields a small number of edges in the WMD and its variants; The preprocessing time usually represents a considerable time of the running time. These limitations end up hindering part of the comparisons between the distances because we do not know how they behave under different document sizes and while disregarding the preprocessing time at the same time.

With this in mind, we designed an auxiliary experiment to deeply understand how the running times of the distances themselves behave under the different documents sizes, expressed in this subsection by the number of edges generated between the documents during the computation of the distances. For that, we sampled 200 documents from 20NEWS and divided them into two groups, each of them of size 100. The number of distinct words in the documents of each group varies from 15 to 500. For the 10.000 pairs obtained by picking one document from each group, we measured the running time to compute the distances between them.

Figure 4.1 shows the behaviour of the running time for the different distances as a function of the size of the transportation graph, measured by the number of edges. The axis y correspond to speed up factors with respect to the running time of WMD, without caching. In this graph, WMD (cache) and RWMD (cache) are, respectively, the versions of WMD and RWMD that use an $O(n^2)$ space cache to store the distances between the word embeddings of the vocabulary. In this case, $n = 22439$. We do not use the same cache strategy as before because there is no specific order of the document pairs being computed this time.

Our first observation is that WMD does not benefit much from caching, which is not surprising since its bottleneck is the computation of the optimal solution of the transportation problem rather than computing the distances between embeddings. The maximum observed speed up factors for WMD (cache) were less than 3. On the other hand, RWMD provides a more considerable speed up, achieving factors around 20-30 for pairs of large documents. In contrast to WMD, its bottleneck is the computation of the distance matrix for each pair of documents. In fact, when we add a cache to RWMD, it achieves a more significant speed up, obtaining factors close to 500 for large documents.

The MF-WMD also yields significant performance gains. As expected, the smaller the value of r the larger the speed up. For $r = 32$, the maximum speed up factors were around 1000 while for $r = 4$ and $r = 1$ the maximum values were close to 2500 and 3300, respectively. It is interesting to observe that for pairs of smalls documents, RWMD (cache) is faster than MF-WMD. This happens because, in this case, r is close to the number of distinct words

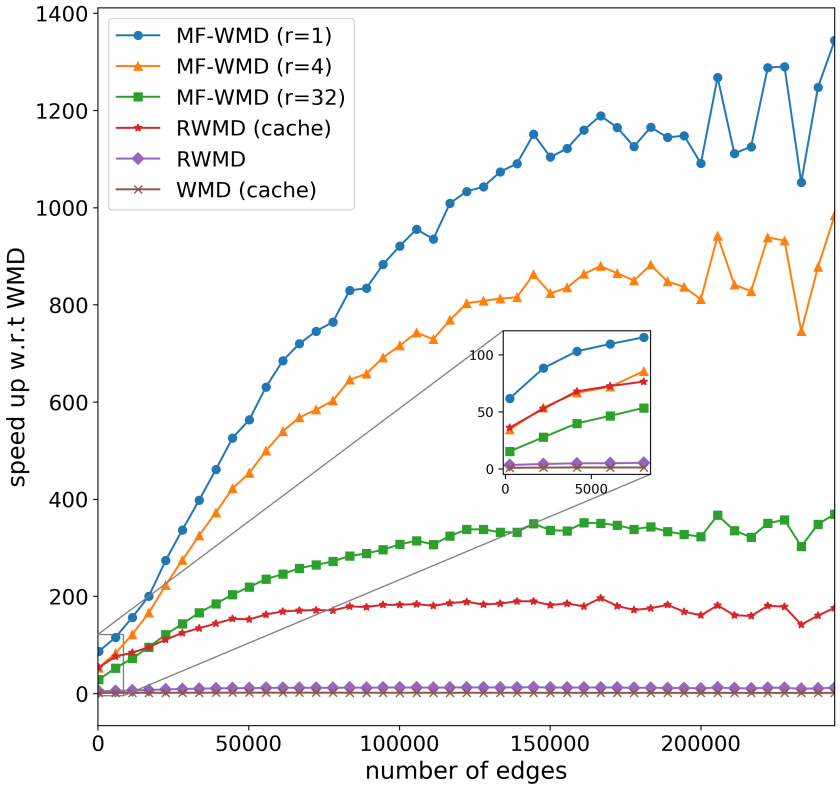

Figure 4.1: Execution time as a function of the product of the number of distinct words in the pair of documents.

in the documents so that the graphs are not sparse and, as a consequence, MF-WMD loses its key advantage. The COSINE distance, not included in the graphs, is by far the fastest one, achieving speed up factors close 200.000.

Finally, with respect to memory consumption, MF-WMD is much more efficient since it requires $O(rn)$ space rather than $O(n^2)$ space. As an example, for a vocabulary of 100.000 tokens, MF-WMD with $r = 32$, uses approximately 12Mb while RWMD (cache) requires around 20GB.

4.3.3 Additional experiment

We have seen in Subsection 4.3.2.3 that MF-WMD – similarly Rel-WMD and Rel-WMD – performs better than its competitors as the size of the documents becomes larger. However, so far we have analyzed their behaviour over datasets mostly composed of small documents. For example, among the datasets tested, the one with largest documents contains 100 tokens per document in average which incurs 10.000 edges when computing distances and falls under the lower end of the spectrum in Figure 4.1.

Thus, to complement our experiments, we created an additional dataset

with larger documents to analyze the behavior of a dataset with documents in the higher end of the spectrum in Figure 4.1 as well. We extracted all video game-related pages for six popular video game genres (action, RPG, puzzle, platform, racing, shooter) from Wikipedia[5]. Pages presented in more than one category were removed. Then we applied the same preprocessing done to the other datasets: lowercase and remove punctuation, accents, numbers, and stopwords. We also removed tokens that do not have a word embedding in the Word2Vec pre-trained word embeddings made available by Google. This resulted in 5200 pages, ranging from 300 to 1500 pages per category and having 275 tokens per page on average. The vocabulary is composed of 40.000 tokens.

Although being our base of comparison, the WMD is absent from all the following tables, imposing some adjustments in them. This occurred because, after running for five days, the experiment had not ended and needed to be canceled. For the sake of comparison, for our previously most-time-consuming dataset (20NEWS), the WMD had taken four days to finish the classification task. Nonetheless, our prior experiments should have displayed enough evidence to support the assumption that the WMD would remain competitive in terms of test error and inferior in terms of running time.

4.3.3.1 Test Error

Table 4.8 presents the test error obtained by the distances under consideration over the WIKIPEDIA dataset.

As before, the COSINE yields the worst test error since it does not capture the semantic relations between different words. Between the remaining distances, they all yield a similar test error which suggests again that just a few pairs of distance are relevant. However, it is interesting that a smaller r provides better results than a larger one. Possibly, the higher number of tokens per document must be increasing the number of edges, bringing pairs of words that are not so related and thus having larger distances. Consequently, this must be smoothing the distance $c_{rel}(D, D')$ between documents D and D'.

Table 4.8: The behavior of test error for different distances on the WIKIPEDIA dataset.

Dataset	BEST	COSINE	RWMD	REL-WMD			REL-RWMD			MF-WMD		
				r=1	r=4	r=32	r=1	r=4	r=32	r=1	r=4	r=32
wikipedia	22.43	1.24	1.00	1.01	1.00	1.05	1.00	1.02	1.04	1.01	1.01	1.05

[5]https://en.m.wikipedia.org/wiki/Category:Video_game_genres

4.3.3.2 Computational Performance

Table 4.9 presents the running times obtained by the distances under consideration over the WIKIPEDIA dataset. In this table, we replace the WMD with the RWMD as our base of comparison since we could not finish the WMD.

Even with respect to the RWMD, the COSINE still achieves a speed-up of three orders of magnitude. However, this improvement over the RWMD is probably due to the use of larger documents, since it should make more evident the discrepancy between the linear and quadratic complexity of the respective methods.

When comparing solely the Rel-WMD, Rel-RWMD and MF-WMD, we can detect that Rel-WMD is up to one order of magnitude slower then the others, without any advantage in terms of test error. Between the Rel-RWMD and MF-WMD, the former continue consistently two times faster than the latter. One possible reason for this constant factor between them comes from the fact that the Rel-RWMD is straightforward algorithm to implement while the MF-WMD needs to resolve a Maximum Flow algorithm which implies copying all nodes and edges to the Library being used.

Additionally, we can confirm the behaviour displayed in Figure 4.1. The WIKIPEDIA contains $275 \times 275 = 75.000$ edges on average, which means that MF-WMD with $r = 1, 2$ and 32 should be approximately $5, 4$ and 1.5 times faster than RWMD (cache). With respect to these values of $r(w)$, the MF-WMD performs a little slower than expected, obtaining a performance improvement of $4, 3$ and 1.5 times respectively. Nonetheless, it is still evident that the difference between those methods increased in this dataset, since RWMD (cache) was faster on average than MF-WMD with $r \geq 2$ on the datasets reported in Subsection 4.3.2.2.

Table 4.9: The speed up factor w.r.t to RWMD of different distances on the WIKIPEDIA dataset.

Dataset	COSINE	RWMD (CACHE)	REL-WMD			REL-RWMD			MF-WMD		
			r=1	r=4	r=32	r=1	r=4	r=32	r=1	r=4	r=32
wikipedia	2315.2	11.2	10.2	7.1	3.3	72.5	62.1	31.3	40.5	32.2	16.6

5
Final Remarks

In this work, we provided strong evidence that it is enough to focus on the nearest words of each word w in the vocabulary during the computation of WMD. This approach allows the replacement of the Transportation Problem with a Max Flow Problem as well as a dramatic reduction is space consumption, while still achieving error rates as good as the WMD. As our insight of reducing all distances to only two can be seen as too aggressive, we also proposed variants of the WMD and RWMD grounded on these same assumptions and analyzed that they achieve similar gains.

In any case, we believe that our approach described in this work could be similarly applied to other algorithms/applications of the field that relies on the computation of distances between embeddings.

Regarding future works, recall that we use $r(w)$ to the denote the number of related word to each word w. Although we set $r(w)$ uniformly in our experiments, we understand that from a semantic perspective this number should vary since some words are related to many more words than others. Thus, one potential line of investigation is on how to choose these values according to the characteristics of the word. One possible way relies on the behaviour displayed in Figure 3.1 where we can see that most words converge to the same value around 1.25. Intuitively, this "point" must varies depending on the word w, as well as the number of words before it, which can be considered as the ones related to w. So the idea is to find a way to automatically identify this point for each word for selecting and set the $r(w)$ in accordance to it.

Another interesting direction for future work is the application of the approach proposed here in settings similar to Huang et al.[16], where accuracies better than those achieved by WMD were reported via its supervised version.

Bibliography

[1] SHANNON, C. E.. **A mathematical theory of communication**. Bell System Technical Journal, 27(July & October):379–423 & 623–656, 1948.

[2] DUMAIS, S. T.; FURNAS, G. W.; LANDAUER, T. K.; DEERWESTER, S. ; HARSHMAN, R.. **Using latent semantic analysis to improve access to textual information**. In: PROCEEDINGS OF THE SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, p. 281–285. Acm, 1988.

[3] BLEI, D. M.; NG, A. Y. ; JORDAN, M. I.. **Latent dirichlet allocation**. Journal of machine Learning research, 3(Jan):993–1022, 2003.

[4] HOTHO, A.; STAAB, S. ; STUMME, G.. **Ontologies improve text document clustering**. In: DATA MINING, 2003. ICDM 2003. THIRD IEEE INTERNATIONAL CONFERENCE ON, p. 541–544. IEEE, 2003.

[5] MIKOLOV, T.; CHEN, K.; CORRADO, G. ; DEAN, J.. **Efficient estimation of word representations in vector space**. arXiv preprint arXiv:1301.3781, 2013.

[6] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S. ; DEAN, J.. **Distributed representations of words and phrases and their compositionality**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 3111–3119, 2013.

[7] PENNINGTON, J.; SOCHER, R. ; MANNING, C.. **Glove: Global vectors for word representation**. In: PROCEEDINGS OF THE 2014 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 1532–1543, 2014.

[8] LE, Q.; MIKOLOV, T.. **Distributed representations of sentences and documents**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 1188–1196, 2014.

[9] KUSNER, M.; SUN, Y.; KOLKIN, N. ; WEINBERGER, K.. **From word embeddings to document distances**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 957–966, 2015.

[10] DAS, R.; ZAHEER, M. ; DYER, C.. **Gaussian lda for topic models with word embeddings**. 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), volumen 1, p. 795–804, 2015.

[11] LI, C.; WANG, H.; ZHANG, Z.; SUN, A. ; MA, Z.. **Topic modeling for short texts with auxiliary word embeddings**. In: PROCEEDINGS OF THE 39TH INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, p. 165–174. ACM, 2016.

[12] RUBNER, Y.; TOMASI, C. ; GUIBAS, L. J.. **A metric for distributions with applications to image databases**. In: COMPUTER VISION, 1998. SIXTH INTERNATIONAL CONFERENCE ON, p. 59–66. IEEE, 1998.

[13] ATASU, K.; PARNELL, T.; DÜNNER, C.; SIFALAKIS, M.; POZIDIS, H.; VASILEIADIS, V.; VLACHOS, M.; BERROSPI, C. ; LABBI, A.. **Linear-complexity relaxed word mover's distance with gpu acceleration**. In: BIG DATA (BIG DATA), 2017 IEEE INTERNATIONAL CONFERENCE ON, p. 889–896. IEEE, 2017.

[14] PELE, O.; WERMAN, M.. **Fast and robust earth mover's distances**. In: COMPUTER VISION, 2009 IEEE 12TH INTERNATIONAL CONFERENCE ON, p. 460–467. IEEE, 2009.

[15] CUTURI, M.. **Sinkhorn distances: Lightspeed computation of optimal transport**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 2292–2300, 2013.

[16] HUANG, G.; GUO, C.; KUSNER, M. J.; SUN, Y.; SHA, F. ; WEINBERGER, K. Q.. **Supervised word mover's distance**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 4862–4870, 2016.

[17] BENGIO, Y.; DUCHARME, R.; VINCENT, P. ; JAUVIN, C.. **A neural probabilistic language model**. Journal of machine learning research, 3(Feb):1137–1155, 2003.

[18] MIKOLOV, T.; KOPECKY, J.; BURGET, L.; GLEMBEK, O. ; OTHERS. **Neural network based language models for highly inflective languages**. In: 2009 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, p. 4725–4728. IEEE, 2009.

[19] GOLDBERG, A. V.; TARJAN, R. E.. **A new approach to the maximum-flow problem**. Journal of the ACM (JACM), 35(4):921–940, 1988.

[20] ORLIN, J. B.. **Max flows in o (nm) time, or better**. In: PROCEEDINGS OF THE FORTY-FIFTH ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, p. 765–774. ACM, 2013.

[21] GOLDBERG, A. V.. **An efficient implementation of a scaling minimum-cost flow algorithm**. J. Algorithms, 22(1):1–29, 1997.

[22] CACHOPO, A. M. D. J. C.; OTHERS. **Improving methods for single-label text categorization**. Instituto Superior Técnico, Portugal, 2007.

A
Experimental results

Tables from A.1 to A.3 display all the results obtained during this work for the distances under consideration on different datasets. For the sake of future reference, test errors and running times are present in their brute values, differing from the approach done in Chapter 4.

Table A.1: The behavior of the brute test error for different distances/datasets.

Distance		20news	amazon	bbcsport	classic	ohsumed	recipe	reuters	twitter	wikipedia
COSINE		30.45	12.90	4.82	6.34	45.74	45.71	8.95	31.97	27.75
WMD		24.09	7.21	5.36	3.04	42.85	46.56	3.84	29.14	—
RWMD		24.64	6.91	5.18	2.81	43.33	46.47	4.07	29.06	22.48
RWMD (cache)		24.64	6.91	5.18	2.81	43.33	46.47	4.07	29.06	22.48
REL-WMD	r=1	25.31	9.80	4.36	3.67	42.21	43.20	5.16	31.87	22.71
	r=2	25.48	9.48	4.73	3.71	42.81	43.52	4.75	30.77	22.68
	r=4	25.16	8.99	4.73	3.54	42.13	43.19	4.39	29.91	22.52
	r=8	24.80	8.66	4.91	3.45	41.55	43.16	4.43	29.33	23.04
	r=16	24.17	8.33	4.73	3.21	41.02	43.33	4.11	29.12	23.20
	r=32	24.21	7.91	4.55	3.12	40.58	43.10	4.07	28.97	23.65
	r=64	23.89	7.57	4.91	3.14	41.32	42.94	4.20	28.61	23.82
	r=128	24.27	7.38	5.27	3.05	41.84	43.26	3.97	28.99	23.77
REL-RWMD	r=1	25.39	9.73	4.36	3.68	42.50	43.34	4.93	31.44	22.43
	r=2	25.16	9.17	4.55	3.54	42.93	43.23	4.66	30.73	22.94
	r=4	25.23	9.11	4.27	3.42	42.15	42.87	4.25	29.46	22.92
	r=8	24.67	8.49	5.36	3.32	41.55	42.79	4.48	29.38	23.34
	r=16	24.84	8.22	4.36	3.17	40.81	42.76	3.88	29.10	23.53
	r=32	24.25	7.69	4.91	3.11	41.51	43.14	4.11	28.86	23.42
	r=64	24.79	7.50	5.36	3.03	41.47	43.11	4.29	28.41	23.31
	r=128	24.66	7.11	5.64	2.94	41.74	43.19	4.25	28.67	23.29
MF-WMD	r=1	25.23	9.87	4.55	3.67	42.38	43.17	5.25	31.85	22.66
	r=2	25.47	9.45	4.91	3.65	42.56	43.39	4.71	30.84	22.85
	r=4	25.06	9.16	4.27	3.53	42.38	42.97	4.34	29.91	22.65
	r=8	24.79	8.56	4.82	3.42	41.72	43.23	4.29	29.23	22.81
	r=16	24.09	8.29	4.27	3.29	41.57	43.30	4.20	29.16	23.21
	r=32	24.03	8.04	4.36	3.11	41.43	43.07	3.97	28.78	23.54
	r=64	24.22	7.65	5.09	3.15	41.84	42.99	4.07	28.76	23.79
	r=128	24.13	7.50	5.36	3.05	41.80	43.48	4.02	29.31	23.54

Table A.2: The behavior of the brute test error for different distances/datasets, replacing the Word2Vec with GloVe embeddings.

Distance		amazon	bbcsport	classic	ohsumed	recipe	reuters	twitter
COSINE		12.93	4.82	6.35	46.19	45.89	8.95	31.97
WMD		7.14	4.82	3.22	42.11	43.30	4.29	28.82
RWMD		7.24	4.36	3.20	42.21	43.60	4.52	28.93
REL-WMD	r=1	9.87	3.91	3.83	42.62	43.69	5.16	31.31
	r=2	9.44	3.91	3.65	42.73	43.60	4.80	30.47
	r=4	8.92	4.36	3.54	42.40	43.83	4.48	29.61
	r=8	8.46	4.55	3.37	42.13	43.22	4.48	29.16
	r=16	8.22	4.73	3.26	42.11	43.05	4.29	28.88
	r=32	7.63	4.82	3.32	42.58	43.20	4.34	28.61
	r=64	7.26	4.82	3.29	42.27	43.10	4.29	28.80
	r=128	7.06	4.91	3.25	41.98	42.64	4.34	29.10
REL-RWMD	r=1	9.48	4.09	3.78	43.04	43.55	5.16	30.94
	r=2	9.05	4.18	3.55	43.10	44.04	4.61	30.04
	r=4	8.64	4.09	3.45	42.67	43.80	4.66	29.57
	r=8	8.18	4.73	3.36	41.94	43.87	4.52	29.25
	r=16	7.75	4.73	3.33	42.42	43.86	4.71	28.86
	r=32	7.42	4.55	3.51	42.42	43.71	4.75	28.52
	r=64	7.43	4.36	3.57	42.91	43.37	4.80	28.65
	r=128	6.95	4.18	3.38	42.54	43.17	4.84	28.97
MF-WMD	r=1	9.86	3.82	3.91	42.46	43.78	5.12	31.29
	r=2	9.41	4.00	3.61	42.91	43.59	4.84	30.21
	r=4	8.96	4.45	3.50	42.40	43.59	4.48	29.76
	r=8	8.52	4.55	3.36	41.88	42.97	4.57	29.33
	r=16	8.15	4.82	3.35	42.60	42.81	4.39	28.76
	r=32	8.01	5.00	3.32	42.32	43.07	4.43	28.73
	r=64	7.48	5.64	3.23	42.32	42.99	4.25	29.12
	r=128	7.41	5.09	3.22	42.32	42.91	4.48	29.38

Table A.3: The running time (in seconds) of different distances/datasets.

Dataset		20news	amazon	bbcsport	classic	ohsumed	recipe	reuters	twitter	wikipedia
COSINE		153.97	23.01	0.32	14.53	46.00	6.29	16.10	2.03	22.23
WMD		374646.69	21081.31	1260.27	12767.47	60107.19	6336.81	10853.37	199.02	—
RWMD		52737.47	4190.00	174.38	2575.09	8841.52	1256.96	2753.71	62.29	51462.71
RWMD (cache)		3110.20	546.35	33.44	285.13	1053.54	82.20	280.21	12.07	4598.68
REL-WMD	r=1	6426.41	734.25	44.76	424.96	1950.64	337.80	585.02	16.00	5045.66
	r=2	7543.17	929.42	50.69	572.67	2599.24	417.74	692.69	22.16	5881.95
	r=4	8969.47	1095.26	58.72	736.72	3407.60	520.73	791.19	29.89	7201.94
	r=8	10683.73	1329.30	68.60	864.07	4038.10	652.28	1105.33	36.88	9283.79
	r=16	13016.65	1651.58	87.49	1113.56	4780.51	846.20	1305.66	47.82	11792.25
	r=32	17117.32	2006.84	117.24	1443.01	6082.22	1150.73	1743.94	56.02	15580.05
	r=64	22873.38	2447.80	157.82	1854.93	7911.98	1508.15	2173.03	74.45	21563.10
	r=128	30397.09	2926.64	217.35	2279.05	9551.16	2033.85	2693.17	94.19	31013.59
REL-RWMD	r=1	1707.80	289.30	21.90	158.27	465.71	51.49	121.56	11.09	709.88
	r=2	1688.83	301.86	21.50	170.10	477.99	58.81	133.70	11.55	751.54
	r=4	1998.20	332.54	23.16	185.18	536.24	67.91	152.54	12.94	828.13
	r=8	2282.90	381.30	24.70	219.09	664.58	85.80	180.21	15.06	952.21
	r=16	3410.23	466.39	28.37	286.16	948.76	123.77	233.21	15.62	1195.25
	r=32	4635.98	600.92	35.15	383.65	1427.01	186.01	309.31	16.40	1643.17
	r=64	6743.20	743.13	45.69	485.84	1941.58	240.45	388.82	18.30	2496.70
	r=128	8765.04	878.83	61.82	562.69	2133.39	263.93	457.18	18.89	4007.57
MF-WMD	r=1	2956.37	449.41	21.01	224.86	761.29	113.36	249.22	11.53	1270.50
	r=2	3425.33	498.38	22.84	268.57	889.04	126.32	282.39	13.70	1395.93
	r=4	3977.14	568.55	25.30	308.03	1022.53	151.11	320.14	17.40	1600.38
	r=8	4840.14	630.02	26.15	372.60	1208.77	179.75	379.93	19.72	1849.95
	r=16	5772.18	761.50	30.02	467.64	1574.38	239.28	483.49	22.70	2302.65
	r=32	8537.08	959.93	38.66	617.28	2186.42	327.55	604.12	25.79	3103.65
	r=64	10185.05	1204.08	51.55	763.82	2952.10	445.55	722.53	29.95	4434.74
	r=128	12548.92	1422.41	69.24	882.31	3490.87	563.86	876.83	32.52	7105.88