Universidade Estadual de Campinas
Instituto de Computação

Ícaro Cavalcante Dourado

Bag of Textual Graphs: an accurate, efficient, and general-purpose graph-based text representation model

Sacola de Grafos Textuais: um modelo de representação de textos baseado em grafos, preciso, eficiente e de propósito geral

CAMPINAS
2016

Ícaro Cavalcante Dourado

Bag of Textual Graphs: an accurate, efficient, and general-purpose graph-based text representation model

Sacola de Grafos Textuais: um modelo de representação de textos baseado em grafos, preciso, eficiente e de propósito geral

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Ricardo da Silva Torres

Este exemplar corresponde à versão final da Dissertação defendida por Ícaro Cavalcante Dourado e orientada pelo Prof. Dr. Ricardo da Silva Torres.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

D748b

Dourado, Ícaro Cavalcante, 1985-
Bag of textual graphs : an accurate, efficient, and general-purpose graph-based text representation model / Ícaro Cavalcante Dourado. – Campinas, SP : [s.n.], 2016.

Orientador: Ricardo da Silva Torres.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Organização da informação. 2. Reconhecimento de padrões. 3. Representações dos grafos. 4. Sistemas de recuperação da informação. I. Torres, Ricardo da Silva,1977-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Sacola de grafos textuais : um modelo de representação de textos baseado em grafos, preciso, eficiente e de propósito geral
Palavras-chave em inglês:
Information organization
Pattern recognition
Representations of graphs
Information storage and retrieval systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ricardo da Silva Torres [Orientador]
Marcos André Gonçalves
André Santanchè
Data de defesa: 19-12-2016
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Ícaro Cavalcante Dourado

Bag of Textual Graphs: an accurate, efficient, and general-purpose graph-based text representation model

Sacola de Grafos Textuais: um modelo de representação de textos baseado em grafos, preciso, eficiente e de propósito geral

Campinas, 19 de dezembro de 2016

Acknowledgements

I give special thanks to my advisor Dr. Ricardo Torres for the interest and support during this research. I also thank him for his valuable words of encouragement and advice, sharing both academic and life experience to me and all his students.

I thank my family, in special my parents, Daniel and Maria Aparecida, for their dedication and unconditional support. I thank my wife, Ana Flávia, for standing by my side and for the companionship and patience.

I thank Dr. Renata Galante, from Federal University of Rio Grande do Sul, for her contributions. I also thank my colleagues from RECOD lab who helped with their experience for useful insights and revision.

I am also grateful to DGA/UNICAMP for its support during the development of this research.

This work was partially funded by CAPES, CNPq, FAPESP, and AMD.

Resumo

Modelos de representação de textos são o alicerce fundamental para as tarefas de Recuperação de Informação e Mineração de Textos. Apesar de diferentes modelos de representação de textos terem sido propostos, eles não são ao mesmo tempo eficientes, precisos e flexíveis para serem usados em aplicações variadas. Neste projeto, apresentamos a Sacola de Grafos Textuais (do inglês *Bag of Textual Graphs*), um modelo de representação de textos que satisfaz esses três requisitos, ao propor uma combinação de um modelo de representação baseado em grafos com um arcabouço genérico de síntese de grafos em representações vetoriais. Avaliamos nosso método em experimentos considerando quatro coleções textuais bem conhecidas: Reuters-21578, 20-newsgroups, 4-universidades e K-series. Os resultados experimentais demonstram que o nosso modelo é genérico o bastante para lidar com diferentes coleções, e é mais eficiente do que métodos atuais e largamente utilizados em tarefas de classificação e recuperação de textos, sem perda de precisão.

Abstract

Text representation models are the fundamental basis for Information Retrieval and Text Mining tasks. Despite different text models have been proposed, they are not at the same time efficient, accurate, and flexible to be used in several applications. Here we present Bag of Textual Graphs, a text representation model that addresses these three requirements, by combining a graph-representation model with an generic framework for graph-to-vector synthesis. We evaluate our method on experiments considering four well-known text collections: Reuters-21578, 20-newsgroups, 4-universities, and K-series. Experimental results demonstrate that our model is generic enough to handle different collections, and is more efficient than widely-used state-of-the-art methods in textual classification and retrieval tasks, without losing accuracy performance.

List of Figures

List of Tables

Acronyms

BoG	Bag of Graphs	18
BoTG	Bag of Textual Graphs	14
BoW	Bag of Words	13
CI	confidence interval	37
DCG	Discounted Cumulative Gain	37
DIG	Document Indexing Graph	15
EMD	Earth Mover's Distance	15
HAC	Hierarchical agglomerative clustering	15
IDCG	Ideal DCG	
KDTree	K-Dimensional Tree	42
kNN	k-Nearest Neighbors	16
LSH	Locality-Sensitive Hashing	42
MAP	Mean Average Precision	37
NB	Naive-Bayes	20
NDCG	Normalized Discounted Cumulative Gain	37

PCA Principal Component Analysis 19

SNB Semi-Naive Bayes 20

SVM Support Vector Machine 13

VSM vector space model 13

Contents

Chapter 1

Introduction

Text mining and information retrieval fields are of paramount importance to support important tasks, such as knowledge discovery and document indexing. Accurate text mining and information retrieval solutions rely on the use of text representation models that should combine both structural and semantic information. Different text representation models have been proposed in the literature. The classic model, called Bag of Words (BoW) [21], quantifies term occurrences within documents, assigning an importance measure to terms found in the document collection. Usually, the importance takes into account aspects such as the term frequency in the document, and its rarity in the collection. Documents are modeled as points in a hyper-space of n dimensions, called vector space model (VSM) [35], where n is the number of distinct terms in the collection. Although largely used due its simplicity and efficiency, the BoW model does not incorporate neither structural nor semantic information. Examples of useful information usually not encoded include term locality, document structural organization (e.g., sentences or paragraphs), and the order and proximity of terms. Typically, even markup information available in web collections is ignored in BoW-based representations.

Graphs are employed in many different applications, such as data modeling in graph databases, data representation of telecommunication networks, and even in the analysis of DNA sequences [12], because they are able to represent arbitrary structures and interrelationships among different elements of a model. Some graph-based text representation models have been proposed aiming to encode structural information from texts [19, 37, 44]. These representations are richer as they can be used to encode both structural and semantics information. In general, such proposals define a way of representing texts as graphs, along with a dissimilarity measure used to assess how close two graphs are. Although these techniques encode contextual information successfully, they strongly depend on the creation or adaptation of data mining or information retrieval algorithms to work on their proposed graph models. Several popular algorithms, such as Support Vector Machine (SVM) [11] and Random Forest [2] for example, can not be directly applied on those graph models. Furthermore, typically used graph-based matching algorithms are computationally costly.

Strategies for converting graphs to vector representations, also known as graph embedding techniques, have been proposed recently [39, 47] to address the shortcomings of graph representations. Their main benefits rely on their capacity of providing a resulting vector representation that can be used along with existing mining and retrieval techniques more easily and less costly. These techniques, however, are application dependent and have to be adjusted for each domain scenario.

In this work, we propose the Bag of Textual Graphs (BoTG), an accurate graph-based text representation model that combines term counting with also locality principles such as proximity and the order of terms within a text, in an efficient general-purpose vector space representation. The objective is to model textual documents as graphs and to project them in a vector space that has lower dimensionality and higher efficiency than the Bag-of-Words model, with comparable accuracy to the graph models but with more general applicability. BoTG is targeted to be used in a wide range of applications.

We validate the model by performing experiments in the context of classification and retrieval tasks, considering different textual collections. Experimental results demonstrate that the proposed model is more efficient and more effective than the traditional Bag-of-Words and graph-based representation models proposed in the literature. In summary, our main contributions are: the proposal of a text document representation model that is effective, efficient, and for general-purpose; and the extension of a recently proposed graph embedding technique proposed in [39] for text modeling in document classification and retrieval tasks.

This dissertation is organized as follows. Chapter 2 covers related work along with the main concepts handled in this work. Chapter 3 presents the BoTG model and describes its components and variants. Chapter 4 describes the experimental protocol used to validate the model in textual classification tasks. Chapter 5 describes the experiments aiming to validate BoTG in textual retrieval tasks. Then, in Chapter 6, we summarize and discuss our contributions and present some possible extensions to this work.

Chapter 2

Related Work

This chapter presents the related work and introduces background concepts useful for understanding our proposal.

2.1 Graph-based text representation

The approaches presented in this section typically model a text document (generally called sample) as a graph, in which a vertex represents a term from the text and an edge represent a relationship between terms.

Hammouda and Kamel [19] proposed a model for document indexing based on term's and sentence's presence in texts. Along with the indexing model, called Document Indexing Graph (DIG), they presented a similarity metric for DIG's that quantifies the occurrence of common subsequences between two graphs. They apply their indexing model for text clustering, using clustering algorithms that must rely only on pairs of distances between samples, such as Hierarchical agglomerative clustering (HAC), as the model does not support other measurements like centroids, means, or medians for the samples. These measures are, however, required by many clustering algorithms, as in K-Means or K-Medians, and this limitation restricts the applicability of the proposed model. As the model is based on common path discovery in graphs, it is very costly. As a positive aspect, it yields significantly better quantitative results compared to the use of Bag of Words for text representation.

Zhang and Chow [44] propose a multi-level representation model for web pages that segments the document in textual sections. This segmentation is done by the detection of text excerpts which are separated by specific tags, such as <p>,</br>,<li>, and </td>. To avoid representing a document with too small sections, subsequent blocks are merged such that each new block has at least 30 words. The proposed dissimilarity measure takes into account both the comparison as in Bag of Words with TF-IDF and a comparison by sections. The comparison of two documents by their sections is modeled as an optimization problem using the Earth Mover's Distance (EMD) [32]. The reasoning in this model is to encapsulate the notion of spatial term distribution, besides considering both global and local semantics. They validate the model in retrieval experiments over web collections. As in [19], this model yields a good representation accuracy in comparison to Bag of Words, but a high cost for text modeling and dissimilarity measurement.

Schenker et al. [37] proposed graph-based text representation models for web pages. These models have some minor differences and vary from two base models called *standard* and *relative-frequency*. The graph models induce distinct terms of a text as labels for vertices to the corresponding text graph, then create oriented edges for each pair of distinct consecutive terms in the same sentence, pointing the edge from the predecessor term to the next one. In the *standard* model, labels for the edges indicate the document section in which the linked terms occurs, and these sections correspond to title (TI), link (L), or text (TX). The *relative-frequency* model, on the other hand, associates weights with the vertices and edges, usually computed as a normalized term frequency and term-connection frequency, respectively: for each vertex v, its weight is given by ${}^{f(v)}/_{max(f(v))}$, where $f(v)$ is the number of occurrences of the label from v in the text; and each edge e is weighted as ${}^{f(e)}/_{max(f(e))}$, where $f(e)$ is the number of occurrences of the pair of terms linked by e.

Figures 2.1 and 2.2 illustrate, respectively, the *standard* and *relative-frequency* graphs from a hypothetical web document with title “YAHOO NEWS” with a link with “MORE NEWS HERE” and a text sentence “REUTERS NEWS SERVICE REPORTS. SERVICE REPORTS.” To calculate the dissimilarity between two graphs, Schenker et al. [37] suggest the use of one of the five discussed metrics in the research, which are well known in the literature and are based on previous extraction of maximum common sub-graph or minimum common supergraph [4, 6, 43, 14]. To validate the models, they perform experiments in web collections using their own customized versions of k-Nearest Neighbors (kNN) for classification and K-Means for clustering. For kNN, they precompute a dissimilarity matrix containing the distances between each sample to the others. For K-Means, which also requires the measurement of centroids, they propose a new function to induce centroids from their graph models. We use the *relative-frequency* model in our proposed framework.

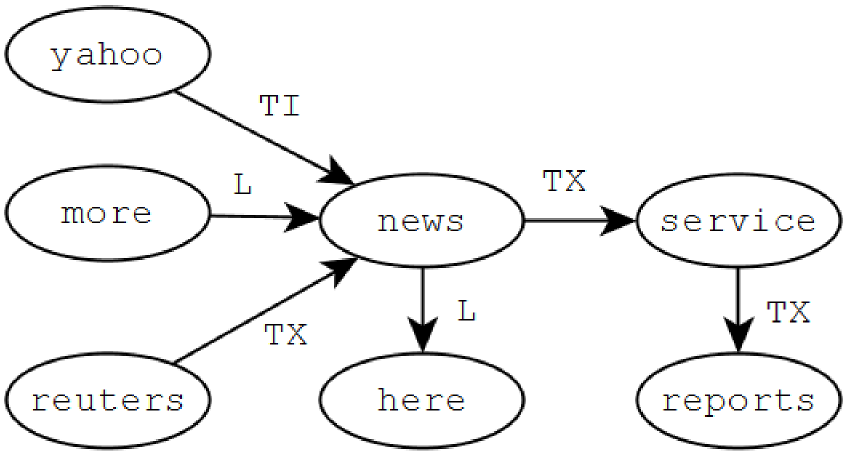


Figure 2.1: A text sample represented by Schenker's standard graph model.

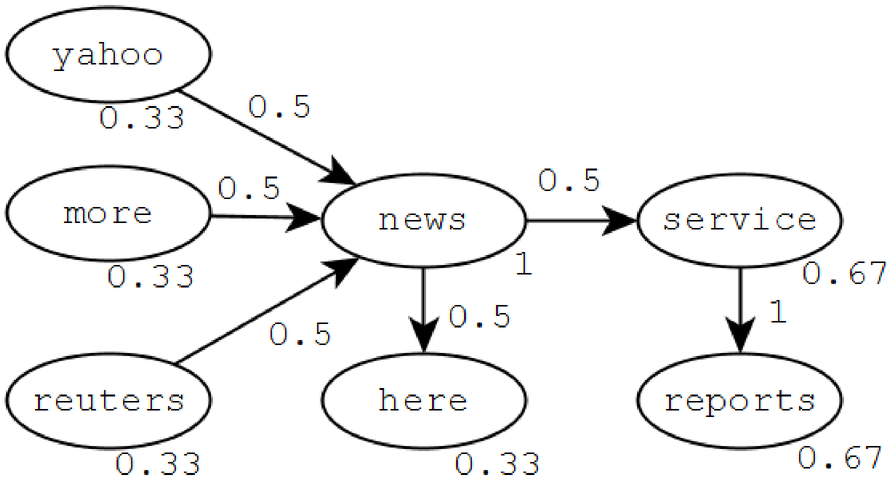


Figure 2.2: A text sample represented by Schenker's relative-frequency model.

2.2 Strategies for graph embedding in vector spaces

The increasingly use of graph databases expanded the use of search systems and mining techniques based on graphs. These approaches, however, deal with costly operations in graphs such as edit distance and extraction of maximum common sub-graph, which are NP-hard [17] operations. One common approach to handle this issue relies on the conversion of graph models into vector spaces (graph embedding), enabling graph-based representations to be used on larger collection and to benefit from existing mining and retrieval techniques.

The works of Riesen et al. [31] and Bunke and Riesen [5] propose a method for mapping graphs in multidimensional space, which heuristically selects x graphs from the collection as training prototypes, and then maps each graph into a x-dimensional vector. Each vector attribute corresponds to the distance between the graph to a prototype. Another approach is to use all the training graphs as prototypes and then apply feature selection algorithms in order to reduce dimensionality. A drawback relies on the need, for each graph sample during the vector projection, of computing a graph distance from it to all the x prototypes, which is still an expensive process when compared to the pure use of graphs in a mining or retrieval task. Gibert et al. [17], in turn, propose a function that maps graphs to vectors based on the statistics of the vertex attributes and edge attributes from the graph. This model, however, does not take advantage of structural information typically found in graphs, not being therefore very representative.

Zhu et al. [47] present a function to map a graph database to a multi-dimensional space, with the advantage of preserving distance and structure characteristics. The edit distance value between two graphs tends to be equal to the distance between their corresponding resulting vectors in multidimensional space, and it is also preserved in this space the distance that a new graph has to any graph from database. Although this presents a novel theoretical step in preserving distances among graphs by using their vectors, it tends to produce output vectors with high dimensionality and with only binary attributes.

Silva et al. [39] propose a generic framework that projects samples based on graphs – from any given application domain – into an induced vector space. Given a graph training set, the mechanism generates a vector space in which the dimensions correspond to words of a vocabulary, where the words (or attributes) are local patterns of training graphs. Local patterns are sub-graphs that are extracted from the graphs based on a previously given formal definition, so that they can be extracted from any input graph, and a dissimilarity function must be defined or provided. This leads to the possibility of designing a vector space, which is based on components from the input graphs, a mechanism called Bag of Graphs (BoG), expanding the concept of Bag of Words to graph domain. In addition to defining a vector space, the framework allows to project a new input sample onto this space. The framework is generic because it works for any input graphs in the sense that only a graph extractor – to induce a graph from an input sample – and a sub-graph model coupled to a dissimilarity function need to be defined. The vocabulary generation is made by clustering sub-graphs from training graphs, where the training graphs can be a subset from the collection, for instance. Each resulting centroid from this clustering defines a word in a vocabulary. For clustering, the authors suggest the use of MeanShift [10] adapted to work with a distance matrix as input. Figure 2.3 summarizes how BoG works. We use a BoG-based encoding approach in our proposed framework.

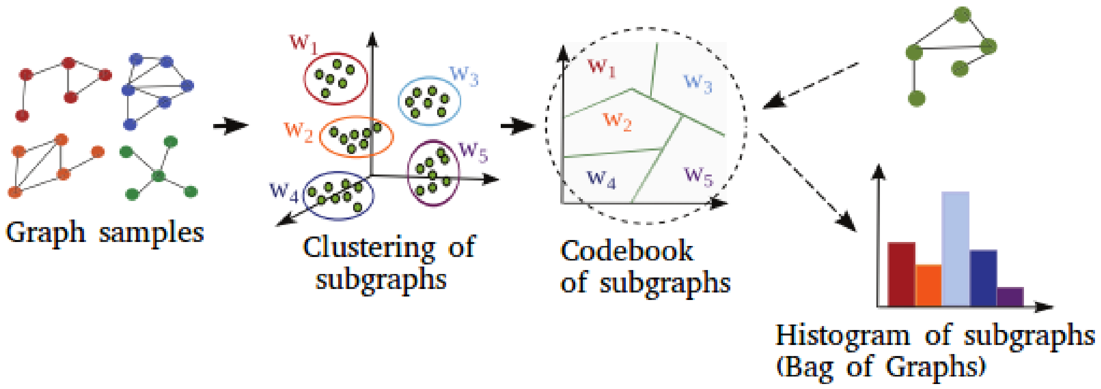


Figure 2.3: The BoG framework. Adapted from [39].

2.3 Vector representation of graph-based text representation models

Markov et al. [28] propose a hybrid text model representation, combining graph-based text representation with boolean vector representation. This model is applicable to web documents and restricted classification scenarios. The *standard* model from [37] is used as the basis and the method generates vectors for the samples – modeled as graphs – quantifying the occurrence of frequent sub-graphs. The frequent sub-graph detection initially identifies, for training, a limited amount of frequent sub-graphs by class, and then takes this sub-graph set as the attributes of a vector representation. For each test sample in the form of graph, the method creates a binary vector, which assigns 1 to each vector attribute (sub-graph) present in the graph, and 0, otherwise. The method is evaluated in text classification under the classifiers C4.5 and Naive-Bayes.

Chow et al. [8] propose a vector representation model based on graphs, applying it in the context of text retrieval. The proposal is based on the *relative-frequency* graph model from [37]. The proposed vector model is a combination of two vectors: first, a vector using the classical Bag of Words with TF weighting; and second, a vector whose attributes are each possible pair of neighbor terms. Each attribute value for pairs of terms, called *connection-term-frequency*, is given by the weight of the edge in the original graph that, in turn, is the frequency that the pair of terms appears in sequence in the text from that graph. Because the number of possible combinations of neighbor terms is high, the final dimensionality of the model is also very high. To overcome this issue, they suggest the use of Principal Component Analysis (PCA) [13] for the vector training set, and, at retrieval step, to apply the same PCA projection on the query samples.

Our model presents some advantages when compared to those approaches. Unlike [28], we adopt more flexible *assignment* and *pooling* procedures, thus not restricting the model to binary weighting schemes. Our model also is not limited to classification tasks due to its independence of the existence of labeled train samples. Therefore our method can be used, for instance, in other applications such as retrieval and clustering. Different from the model of [8], which takes into account both term occurrence and term neighborhood, our method can co-relate terms in a broader context proximity. As we will show later, this property makes our model more accurate and more flexible than methods based on strict term neighborhood definitions.

2.4 Co-occurrence-based text representation models

Despite the presence of graph-based models to represent textual documents, there is a vast literature of text models based on co-occurrence of terms. They vary on their definition of co-occurrence and some of them are specific to a certain goal.

N-grams are sequences of n adjacent words and can be used for bag representations, called bag of n-grams. In this approach, a document is represented by a vector whose attributes can be both singular words and n-grams up to a particular value of n. The reasoning of n-grams is to capture common expressions. Bag of n-grams has been shown to be more effective than BoW for n values up to 5 [40]. However, n-grams presents higher complexity than BoW and the gains are usually either absent or marginal [1].

Skip-grams generalizes n-grams so that the words do not need to be contiguous. Instead, it allows words to be skipped in gaps up to a certain size k [18]. 1-skip-bi-gram, for example, is a skip-gram model with maximum gap size k of 1 that considers all the sequences of 2 words with 0 to 1 words as a gap between the pair. This approach overcomes the data sparsity problem of n-grams, at a cost of producing even larger language models.

Figueiredo et al. [15] propose the notion of compound-features for text classification tasks, a relaxed skip-bi-gram model whose attributes are composed of words that co-occur without restrictions on order or distance between them within the document. Furthermore, this approach leads to a very high dimensionality – 2^t for a collection of t terms – and additional noise. In fact, the proposal presents some mandatory feature selection strategies to discard irrelevant features and reduce noise. The feature selection retains compound-features that are present in documents of a certain class but not in the others. This model presents good effectiveness and efficiency, but it is limited to the text classification scenario.

Although some works focus on language model generation, which can be used with different algorithms and for different tasks, other works propose a direct algorithm modification, such as into Naive-Bayes (NB). NB is a probabilistic classifier algorithm which assumes that each attribute is independent to one another, and applies bag principles as in BoW. Semi-Naive Bayes (SNB) relaxes the independence assumption and incorporates feature dependency [45], promoting more effective classification results in cases where NB performs badly, such as collections with class imbalance, feature sparseness, and strong relationships among attributes. SNB are not usually applied in large collections due to their high training costs [42].

Viegas et al. [42] propose a SNB model that employs a novel lazy feature selection strategy. That lazy strategy selects the most important features from the documents and produces a more compact language model than those obtained from other SNB-based initiatives, leading to gains in terms of effectiveness and efficiency. This proposal, however, is specific to text classification and was only compared to BoW-based approaches of kNN and SVM.

Chapter 3

Bag of Textual Graphs

This chapter introduces the proposed Bag of Textual Graphs (BoTG), a graph-based textual representation model. Section 3.1 provides an overview of the proposed method highlighting its main components. Sections 3.2, 3.3, and 3.4 present, respectively, how we extract graphs from textual documents, how we create a vocabulary based on graph representations, and how vector representations can be obtained based on the generated vocabulary.

3.1 Overview

In order to create a model that meets the goals of efficiency, accuracy, and flexibility, we propose a combination between a graph-based representation model [37] and a framework for graph-to-vector space projection called *Bag of Graphs* (BoG) [39]. The efficiency and flexibility of our method come from the resulting vector representation, whereas accuracy is achieved from the graph-based model applied beforehand. Our proposed representation model is expected to be used in a wide range of applications such as text mining or information retrieval tasks in general (e.g., text clustering, classification, or ranking). This dissertation focuses on textual document classification and retrieval problems.

Figure 3.1 shows the proposed pipeline employed to define vector-based representations from graphs. Two sequences of steps are presented, one for vocabulary creation and another for creating the representation given the created vocabulary. Given a set of samples (referred to here as *training set*), we initially extract their corresponding graphs (step *1a*). Later sub-graphs within those graphs are extracted based on predefined criteria (*2a*), and then are used to create a graph-based vocabulary used to define a vector space model (*3*). Samples are represented by vectors created based on a projection of their graphs onto the created vocabulary. *Assignment* and *Pooling* (*4a*) functions are used in the projection. Similar steps are employed for handling novel samples. Given a novel sample, its corresponding graph (*1b*) is extracted and key sub-graphs (*2b*) are defined. Later, these sub-graphs are mapped to the created vocabulary in order to define its vector representation based also on the use of specific *Assignment* and *Pooling* functions (*4b*).

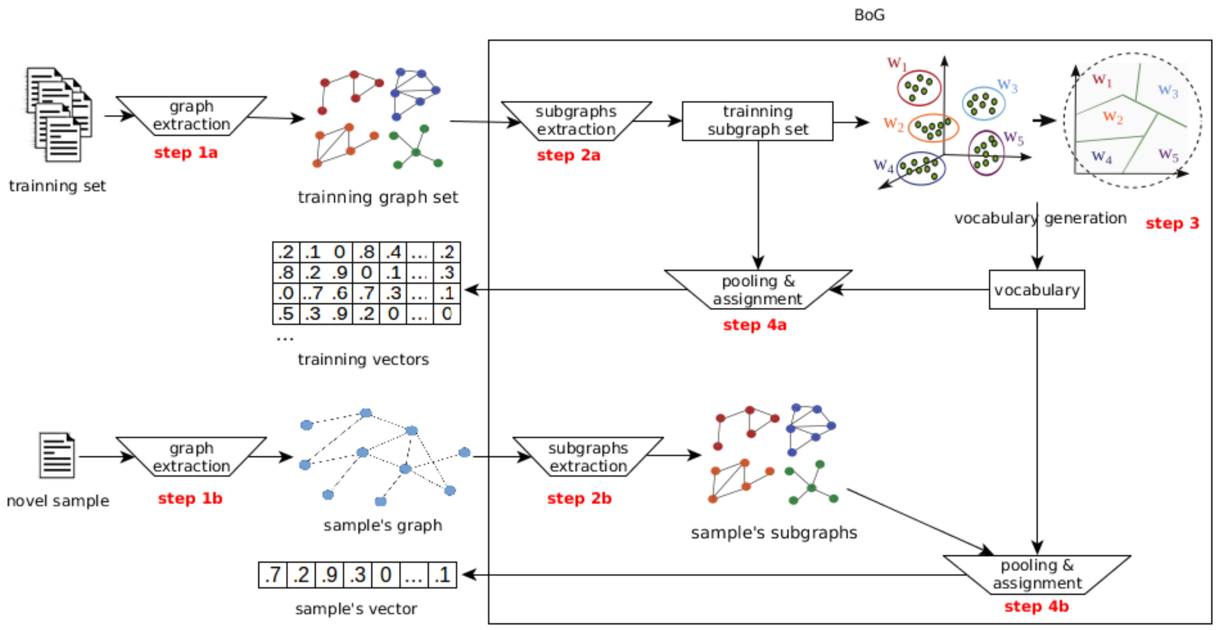


Figure 3.1: Proposed pipeline for vector representation of graphs extracted from textual documents.

3.2 Graph-based text modeling

The first step of our method relies on the representation of textual documents onto graphs. In the beginning, documents are pre-processed with the objective of detecting textual *segments* defined in terms of sections and sentences within the text. Next, these segments are decomposed into a sequence of terms. Later, this sequence is refined by removing stop words based on a publish list[1] and by using the Porter's stemming algorithm [30].

For the graph-based text modeling, we use a modified version of the relative-frequency model proposed in [37]. Let $G = (V, E)$ be a graph, where V is a set of vertices and E is a set of edges. A vertex $v \in V$ refers to a term of the document. Edges $e \in E$ are defined based on the co-occurrence of terms within a same textual segment. Our graph model uses TF-IDF weighting for vertices and edges in order to incorporate information for rarity importance as proposed in [34]. For a directed edge, TF corresponds to the number of occurrences of its ordered pair of terms in the related document, and DF is the number of documents in which that pair occurs.

Sub-graphs within G are defined as follows: a directed sub-graph $G_t = (V_t, E_t)$, with $V_t \subseteq V$ and $E_t \subseteq E$ is created for each term t. There is an edge $e(i, j) \in E_t$ linking terms t_i to t_j if t_j appears after term t_i within a sequence of terms of a segment. Edges may be defined based on the size of the segments considered. The larger the size, the more edges are defined. A large segment allows the expansion of the term context, which may lead to better accuracy results. In Figure 3.2, we show sub-graphs extracted from the example shown in Figure 2.2, based on segments of size 1.

[1]Stop list available at `http://code.google.com/p/stop-words/` (As of January 2017).

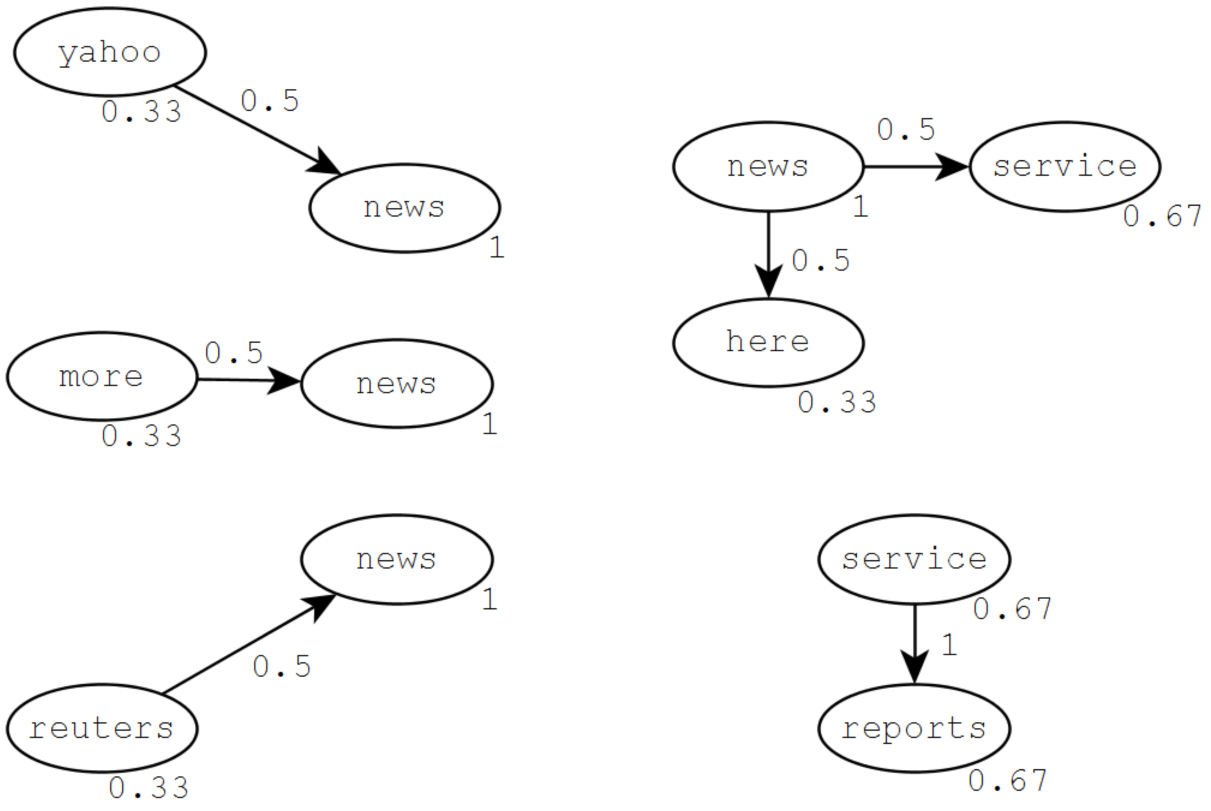


Figure 3.2: Sub-graphs from the weighted graph sample in Figure 2.2, based on segments of size 1.

3.3 Vocabulary Creation

The objective of this step is the creation of a graph-based codebook on which graphs of textual documents will be projected in order to define a vector representation. Here we present two possible approaches for codebook creation.

3.3.1 Vocabulary Creation based on subset selection

In this approach, the vocabulary creation consists of selecting a subset from the training subgraph set. This can be done in different ways, such as randomly selecting a maximum of subgraphs, or clustering of sub-graphs based on their dissimilarity. For large collections, clustering of sub-graphs may be a practical limit due the large number of subgraphs, but this issue can be addressed by prior subset sampling or approximate approaches for the clustering. Clustering is a more costly way but usually produces better codebooks [39]. In preliminary experiments, we noticed better results by clustering selection over random selection, so we decided to use clustering selection.

Different functions can be used for sub-graph dissimilarity measurement. There are functions based on the maximum common sub-graph, such as MCS [6] or WGU [43]. We present in Algorithm 1 the proposed dissimilarity function, which differs from those previously mentioned as it focuses on the contextual information defined in terms of the edges defined by the terms (and their neighbors) being compared. We apply the dissimilarity function in two cases: the subgraph clustering within vocabulary creation, and the *Assignment* phase (see steps *3* and *4*, in Figure 3.1). We that this proposed function is restricted to our sub-graph definition. For understanding Algorithm 1, the following definitions are considered:

- G_A and G_B are connected sub-graphs, weighted on nodes and edges, containing only a central term t_{G_A} (or t_{G_B}), and its neighbor nodes;
- $n\_weight(t, G)$ returns the weight of the node's term t in the graph G.
- $e\_weight(t_1, t_2, G)$ returns the weight of the edge linking the term t_1 to t_2 in G.
- $neighbors(t, G)$ returns the neighbor terms of a given term t in G.

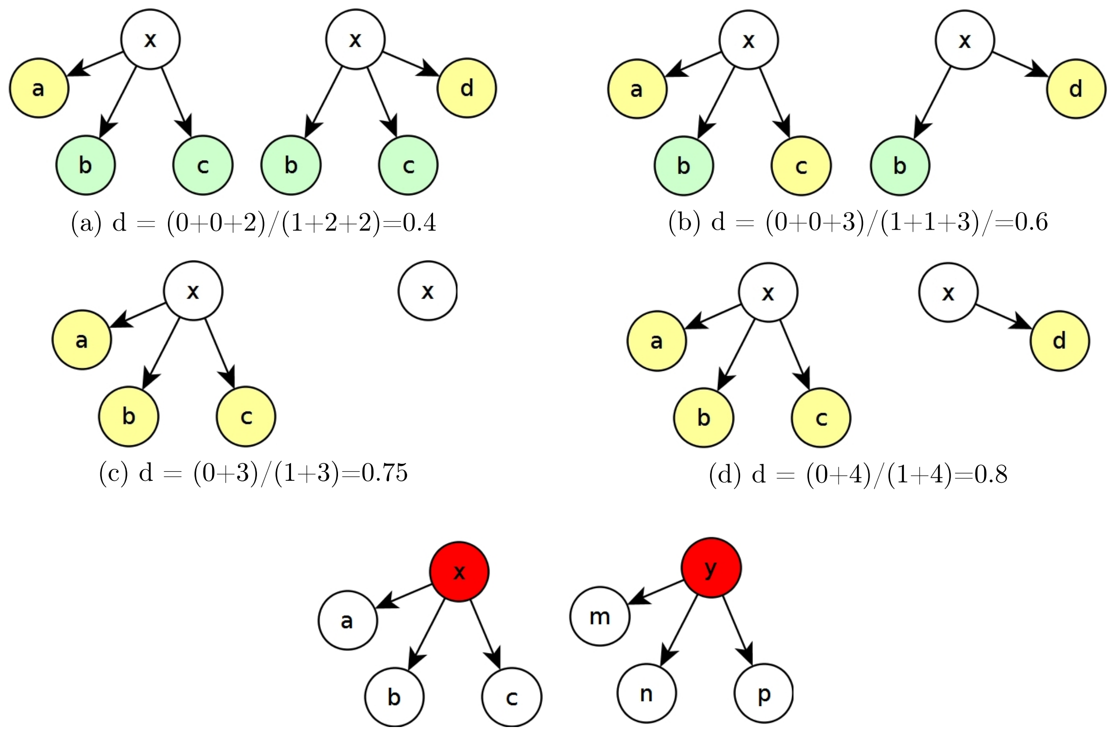
Given G_A and G_B, Algorithm 1 provides a dissimilarity in $[0, 1]$, with 0 for equivalent sub-graphs and 1 for sub-graphs with different central terms (Lines 1 to 2). *dist* accumulates the node weight difference between central nodes from G_A and G_B plus the edge weight differences from the sub-graphs. For edges connecting terms present in both sub-graphs, it adds the weight difference (Lines 7 to 9), and for all other edges it adds 1 (Lines 10 to 12), which is also the limit value for both edge and node weights in our prior graph model. *numComparisons* is used in the end to limit the dissimilarity in desired interval (Line 13).

Figure 3.3 provides examples related to the use of this function, considering weights equal to 1. Red vertices indicate different central terms between the subgraph pair, which gives dissimilarity 1 (example *e*). For remaining cases, the central term's weight difference (0 in the examples) is added to *dist*, and *numComparisons* then starts by 1. Green vertices indicate common term neighbors, and each common neighbor adds the edge's weight difference (0 in the examples) to *dist*, and 1 to *numComparisons*. Yellow vertices indicate different term neighbors, and each one adds 1 to *dist*, and 1 to *numComparisons*.

Algorithm 1 Proposed sub-graph dissimilarity function.

```
1: if $t_{G_A} \neq t_{G_B}$ then
2:     return 1
3: $neighborsGA \leftarrow neighbors(t_{G_A}, G_A)$
4: $neighborsGB \leftarrow neighbors(t_{G_B}, G_B)$
5: $dist \leftarrow |n\_weight(t_{G_A}, G_A) - n\_weight(t_{G_B}, G_B)|$
6: $numComparisons \leftarrow 1$
7: for all $t' \in neighborsGA \cap neighborsGB$ do
8:     $dist \leftarrow dist + |e\_weight(t_{G_A}, t', G_A) - e\_weight(t_{G_B}, t', G_B)|$
9:     $numComparisons \leftarrow numComparisons + 1$
10: for all $t' \in (neighborsGA \cup neighborsGB) - (neighborsGA \cap neighborsGB)$ do
11:     $dist \leftarrow dist + 1$
12:     $numComparisons \leftarrow numComparisons + 1$
13: return $^{dist}/_{numComparisons}$
```

Different algorithms can be used for graph clustering [36, 46]. In fact, the dimensionality for the vocabulary produced depends on the clustering method. In this work, we use MeanShift [10], adapted to work with a dissimilarity matrix as input. In MeanShift, the



(a) d = (0+0+2)/(1+2+2)=0.4

(b) d = (0+0+3)/(1+1+3)/=0.6

(c) d = (0+3)/(1+3)=0.75

(d) d = (0+4)/(1+4)=0.8

(e) d = 1

Figure 3.3: Examples of dissimilarity (d) measurements for pairs of subgraphs, based on Algorithm 1 and considering weights 1.

number of seeds and the bandwidth used affect the final number of clusters considered in the vocabulary. If desired, K-Medoids could be used to define a codebook with a specific dimensionality.

3.3.2 Vocabulary Creation based on attribute fusion

Our prior formulation for vocabulary creation to graph embedding tends to generate a vector space with less dimensions than Bag-of-Worlds' space (typically less than 50%), as we will show later in validation experiments. Although this provides efficiency to our method, sometimes its codebook quality is not enough because possible relevant terms from collection are left out in the representation model. To handle this issue, two approaches are used:

1. Induce the vocabulary generation to provide larger codebooks. This requires to adjust the clustering parameters accordingly, if possible;

2. Generate a more restricted codebook containing only subgraphs that have edges, and then create a hybrid vector model whose attributes are composed of both this codebook and the terms from BoW. This will project an input graph as a concatenated pair of vectors, each one computed differently but composing a unique representation.

This first approach may lead to a slower clustering selection. The idea behind the second approach is to encode an expanded vocabulary that contains both isolated terms, as in BoW, as well as some relevant contextualized terms. Contextualized terms are encoded to the vocabulary in the form of subgraphs. This is roughly analogue to use of n-grams [33], but with the advantage of a more concise dimensionality. This is also similar to the strategies that some of our counterparts [8, 28]: instead of creating a new vocabulary, we expand a preliminary one by means of some selection criteria. We will refer to this approach as BoTG$_{alt}$. Figure 3.4 illustrates how BoTG$_{alt}$ feature vector is computed.

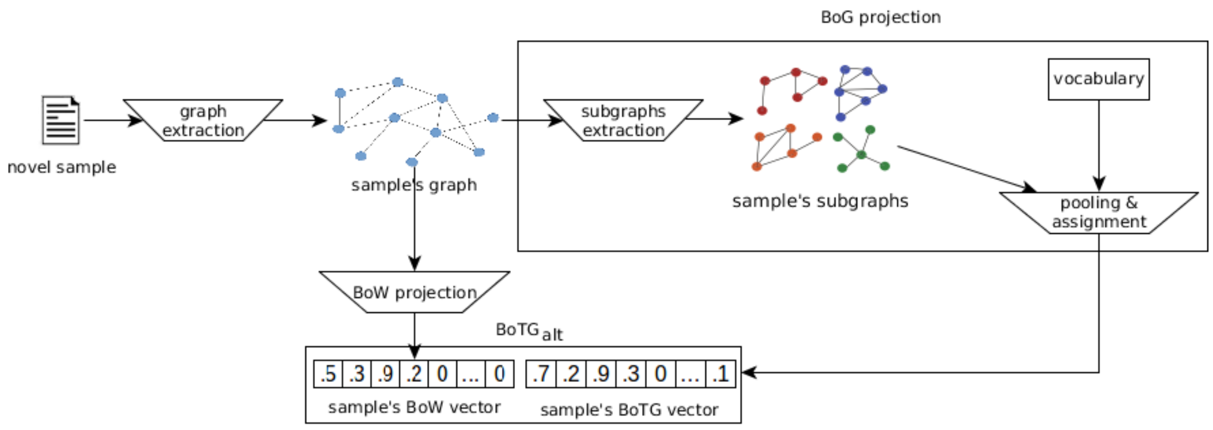


Figure 3.4: Vocabulary creation based on attribute fusion.

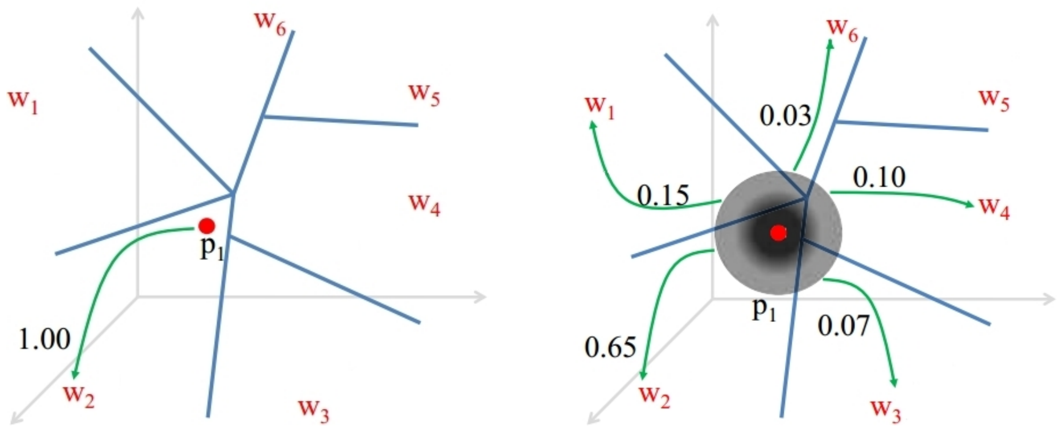
3.4 Vector Representation Creation: Assignment and Pooling Functions

The creation of vector representations relies on the projection of graphs of a textual document onto the created vocabulary. Two steps are used to guide this process: sub-graph assignment and pooling. We investigate in Section 4.3.3 the influence of different assignment and pooling functions over classification accuracies in our experiments.

An assignment function maps each sub-graph to a codebook cluster (also known as word). In a *hard* assignment, the sub-graph is associated with the closest centroid (word) in the space model. A *soft* assignment, in turn, adopts a kernel function to establish the degree of a sub-graph belonging to different clusters. Let S be a set of sub-graphs of a certain input graph. The hard assignment a_{ij} for the sub-graph $s_i \in S$ to the cluster (word) w_j is given by Equation 3.1, where N is the number of clusters and $D(s_i, w_k)$ computes the dissimilarity between s_i and w_k. Equation 3.2 defines a soft assignment that uses a Gaussian to smooth the dissimilarities [41], where $K(x) = {exp(-\frac{x^2}{2\sigma^2})}/{\sigma\sqrt{2\pi}}$, and σ allows the smoothness control. Figure 3.5 shows an example of *hard* and *soft* assignments.

$$a_{ij} = \begin{cases} 1, & \text{if } j = \underset{1 \le y \le N}{\text{argmin}}\, D(s_i, w_y) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$a_{ij} = \frac{K(D(s_i, w_j))}{sum_{k=1}^{N} K(D(s_i, w_k))} \tag{3.2}$$



(a) hard (b) soft

Figure 3.5: Example of (a) *hard* and (b) *soft* assignment for a point p_1 (red circle). Green arrows indicate the words assigned to p_1 and the corresponding assignment values. Adapted from [29].

Pooling functions summarize into a single output vector the assignments performed for all sample sub-graphs. For the *sum* pooling, each $v[j]$ associated with the word j from vector $v[1..N]$ is given by the sum of all relations between the input sample sub-graphs to j (Equation 3.3). *Average (avg)* pooling associates with $v[j]$ the percentage of associations to the word j, as shown in Equation 3.4. In the *max* pooling (Equation 3.5), $v[j]$ is the highest value associated with the word j. Table 3.1 illustrates an example of *sum*, *avg*, and *max* pooling for a graph with 8 subgraphs, considering a codebook of 10 words.

$$v_j = \sum_{i=1}^{N} a_{ij} \tag{3.3}$$

$$v_j = \frac{\sum_{i=1}^{N} a_{ij}}{|S|} \tag{3.4}$$

$$v_j = \max_{1 \leq i \leq S} a_{ij} \tag{3.5}$$

Table 3.1: Example of *sum*, *avg*, and *max* pooling, for a graph with 8 subgraphs, a vocabulary of 10 words and the use of soft assignment. Adapted from [29].

	Words									
Subgraphs	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
s_1	0.02	0.30	0.10	0.00	0.00	0.00	0.00	0.58	0.00	0.00
s_2	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.10
s_3	0.40	0.00	0.00	0.20	0.00	0.10	0.10	0.20	0.00	0.00
s_4	0.00	0.00	0.00	0.00	0.00	0.50	0.40	0.00	0.05	0.05
s_5	0.05	0.05	0.00	0.10	0.00	0.00	0.00	0.00	0.80	0.00
s_6	0.00	0.95	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
s_7	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.00	0.60
s_8	0.00	0.30	0.30	0.30	0.00	0.00	0.00	0.00	0.00	0.10
sum	0.47	1.60	0.45	0.60	0.90	0.60	0.70	0.98	0.85	0.85
avg	0.06	0.20	0.06	0.08	0.11	0.08	0.09	0.12	0.11	0.11
max	0.40	0.95	0.30	0.30	0.90	0.50	0.40	0.58	0.80	0.60

Chapter 4

Validation in Textual Classification Tasks

This chapter presents adopted experimental protocol aiming to validate the proposed BoTG model comprising the scenario of text classification. Performed experiments consider the analysis of the effectiveness and the efficiency of BoTG, as well as of baseline methods. The main goal here is to validate the hypothesis that BoTG aggregates the benefits from both baselines being considered, achieving comparable or even superior efficiency and effectiveness performances.

4.1 Datasets and Baselines

The experiments are performed on the following collections:

- Reuters-21578[1]: 21,578 news from Reuters newswire, categorized according to five different category sets: topics, places, people, orgs, and exchanges. We consider topic classification. A few samples are unlabeled or multi-labeled. This collection has very skewed distribution: documents per class vary from less than a dozen to almost four thousand.

- 4-universities[2]: 8,282 web pages distributed into 7 classes, collected from website departments of the universities Cornell, Texas, Washington, and Wisconsin.

- 20-newsgroups[3]: 20,000 messages from USENET forum, distributed in 20 classes (called newsgroups), with one thousand samples per class. The collection contains some multi-labeled samples.

- K-series[4]: 2,340 web news pages from Yahoo, distributed into 6 categories: business, health, politics, sports, technology, and entertainment. This is a skewed collection, with classes containing from 60 to more than one thousand documents. Some pages

[1]`http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection` (As of January 2017).

[2]`http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data` (As of January 2017).

[3]`http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups` (As of January 2017).

[4]`http://www-users.cs.umn.edu/~boley/ftp/PDDPdata` (As of January 2017).

Table 4.1: Statistics of collections after preprocessing and graph extraction.

Collection	# classes	# samples	# terms	Mean number of nodes per graph
Reuters-21578	65	8,655	19,370	41
4-universities	7	8,202	53,428	92
20-newsgroups	21	10,945	64,011	69
K-series	6	2,340	25,843	175

are distributed into sub-categories. In our experiments, we do not use sub-category information.

In the experiments regarding classification, we discard from the collections the samples that are unlabeled or multi-labeled, aiming simplicity but without loss of generality. The samples with empty body text are also discarded.

Initially, in an offline stage, all collection documents are represented as graphs, as explained in Section 3.2. The offline stage also comprises the vocabulary generation and classification training. We call as online stage the steps involved for an input graph, with starts by projecting it within the BoTG's vector space (see Section 3.4), and then – depending on the experimental scenario – either classifying the sample or using it as a query for retrieval.

We show in Table 4.1 statistics from collections after pre-processing steps and graph extraction. The collections vary in terms of their contents, the number of classes and samples and also the size of their samples, what reflects on the mean number of nodes per extracted graph. All collections presented more terms than samples, which is a common characteristic in text collections.

We conduct experiments that compare the proposed BoTG model with the traditional Bag of Words (BoW) and with the relative-frequency graph model [37]. The same text preprocessing procedures previously mentioned are used for BoTG and for the two baseline models. For the use of Bag of Words, we adopt TF-IDF weighing and rescale attribute values in the range $[0, 1]$. The relative-frequency model is used as baseline in the experiments following its original proposal.

4.2 Experimental Procedures

We adopt as evaluation criteria the comparison of both the effectiveness and efficiency performance of the methods in text document classification tasks.

We conduct a 10-fold cross-validation protocol to compare the proposed BoTG method with the baselines. We use a stratified cross-validation to impose folds with same size and same class distribution. At each step, one fold is used for testing and the others are used for creating the BoTG model and training the classifier. The test samples are later classified and we measure the fold classification effectiveness. The same process is performed for every fold and, at the end, we compute the mean effectiveness score and the standard deviation.

The effectiveness of evaluated methods is measured in terms of macro-F_1, which is derived from standard measures such as precision and recall. Due to the collections used in our experiments are skewed, F_1 is preferred over the use of accuracy. Let $C = \{c_1, c_2, \ldots, c_k\}$ be the classes of a certain collection. By initially computing the confusion matrix, we get the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) per class. Equation 4.1 presents the precision rate for class i, Equation 4.2 presents recall for class i, and Equation 4.3 indicates how F_1 is measured for class i. Macro-F_1 is defined as the average of the F_1 values for each class (Equation 4.4).

$$P(c_i) = \frac{TP_i}{TP_i + FP_i} \tag{4.1}$$

$$R(c_i) = \frac{TP_i}{TP_i + FN_i} \tag{4.2}$$

$$F_1(c_i) = \frac{2P(c_i)R(c_i)}{P(c_i) + R(c_i)} \tag{4.3}$$

$$macro - F_1 = \frac{\sum_{i=1}^{K} F_1(c_i)}{K} \tag{4.4}$$

The effectiveness comparison between models uses the mean macro-F_1 from the folds and we validate it with the Wilcoxon-signed rank test [38] paired on folds to analyze if there is statistical significance between results from different models, considering 95% confidence level. The efficiency evaluation is based on the mean time spent to classify a test sample.

We apply the same process using different classifiers to compare the models under different classification scenarios: kNN, Random Forest, and SVM. For both BoTG and BoW, we measure F_1 with the same classifiers, and for relative-frequency we adapt kNN to work with distance matrices which are precomputed with specific graph-based functions, such as MCS [6] or WGU [43]. The folds are created before the classification procedures and all methods use the same folds. We adopt kNN with three different configurations, one for each distance function used: cosine, Jaccard, and Euclidean. We use Random Forest with 100 stamps and $log_2(dimensionality - 1) + 1$ attributes. We use SVM with linear kernel and adjusted for multi-class classification using one-vs-all strategy. Given the presence of hyperparameters, we use grid search internally to the cross-validation in order to adjust the classifier model, in a procedure known as nested cross-validation: during the analysis on the k-th fold, the remaining $k - 1$ folds are used for training and evaluation, so that a internal cross-validation finds an optimal configuration, which is then used to train the model over the $k - 1$ folds, which is finally used to test the k-th fold. We vary k from kNN in $\{1, 3, 5, 7, 11\}$ and C from SVM in powers of 10 between 10^{-3} and 10^3.

Performed experiments also evaluate different parameter settings of the proposed representation, which are the number of seeds for MeanShift, varying between 10% and 100% in steps of 10%, and the segment size during subgraph extraction, which we vary in $\{1, 3, 5\}$.

4.3 Results and Analysis

This section presents and discusses experimental results related to the evaluation of the proposed graph representation in comparison to baselines with regard to their effectiveness and efficiency performances. We also discuss results related to the parameter settings evaluation.

4.3.1 Effectiveness Evaluation

Table 4.2 shows the results obtained for the collections, using Macro-F_1, comparing BoW and BoTG. Table 4.3 compares the relative-frequency results to BoTG comprising kNN – same classifier but with different distance functions – and SVM (overall best classifier for BoTG). We adopt the following symbols to denote the statistical comparisons of our method:

- Symbol ▲: it overcame the baseline;
- Symbol ●: it was tied to baseline but with better or equal mean;
- Symbol ○: it was tied to baseline but with lower mean;
- Symbol ▽: it was worse than the baseline.

Table 4.2: Comparisons between BoW and BoTG with Macro-F_1, over the collections.

	kNN-Euclidean		kNN-cosine		kNN-Jaccard		Random Forest		SVM	
Collection	BoW	BoTG	BoW	BoTG	BoW	BoTG	BoW	BoTG	BoW	BoTG
Reuters-21578	68.8 ± 2.8	71.7 ± 1.9 ▲	88.8 ± 2.6	88.3 ± 2.4 ○	88.0 ± 2.5	88.5 ± 2.4 ●	82.5 ± 3.0	85.6 ± 2.2 ▲	94.3 ± 1.1	93.7 ± 1.0 ○
4-universities	43.5 ± 3.6	49.9 ± 3.5 ▲	63.2 ± 3.1	66.0 ± 3.1 ▲	64.3 ± 3.0	65.8 ± 2.8 ▲	61.4 ± 2.1	63.6 ± 2.3 ▲	77.3 ± 1.9	76.0 ± 1.8 ○
20-newsgroups	57.1 ± 1.6	55.3 ± 1.9 ▽	85.5 ± 1.0	84.2 ± 1.2 ▽	84.9 ± 0.8	84.4 ± 0.9 ▽	78.2 ± 1.7	78.4 ± 1.1 ●	86.5 ± 1.3	89.0 ± 0.9 ▲
K-series	64.3 ± 2.8	73.6 ± 2.5 ▲	91.9 ± 2.4	93.3 ± 2.2 ●	92.6 ± 2.2	93.6 ± 1.8 ●	71.9 ± 7.5	77.5 ± 4.0 ▲	99.0 ± 0.6	99.5 ± 0.6 ●

Table 4.3: Comparisons between relative-frequency and BoTG with Macro-F_1, over the collections.

Collection	relative-frequency & kNN-MCS	BoTG & kNN	BoTG & SVM
Reuters-21578	88.6 ± 2.2	88.5 ± 2.4 (cosine) ○	**93.7** ± 1.0 ▲
4-universities	62.5 ± 1.7	66.0 ± 3.1 (cosine) ▲	**76.0** ± 1.8 ▲
20-newsgroups	86.8 ± 0.9	84.4 ± 0.9 (Jaccard) ▽	**89.0** ± 0.9 ▲
K-series	95.6 ± 1.7	93.6 ± 1.8 (Jaccard) ▽	**99.5** ± 0.6 ▲

In Reuters-21578, BoTG overcame BoW for the kNN-Euclidean and Random Forest, and tied statistically considering the kNN-cosine, kNN-Jaccard, and SVM classifiers. Compared to the relative-frequency model, no statistical differences were observed when the kNN classifier was used. However, BoTG overcame this baseline when SVM was used, one of several classifiers that can be applied to our model but not to the graph model directly. The results point out that our model overcomes the restricted use of its corresponding baseline graph model.

In 4-universities, BoTG overcame BoW for all classifiers except for SVM in which there was a statistical tie. BoTG also overcame relative-frequency for all classifiers considered in our study. In 20-newsgroups, BoTG performed better than BoW and relative-frequency by its overall best F_1, achieved by SVM. In this collection, relative-frequency overcame BoW. In K-series, BoTG performed better than BoW for kNN-Euclidean and Random Forest, and tied for the others. BoTG also overcame relative-frequency when the SVM classifier is used (see Table 4.3).

From Table 4.2, it can be noticed that BoTG was worse than BoW for 20-newsgroups in the three kNN variants, and tied with lower mean in Reuters-21578 for kNN-cosine and SVM. Our model, however, presented good results for most cases. We also investigated the use BoTG$_{alt}$ for these worst cases to check if a joint representation combining BoW and BoTG would yield better results. Tables 4.4 and 4.5 present the results of BoTG$_{alt}$ in these cases, respectively for Reuters-21578 and 20-newsgroups. For Reuters-21578, BoTG$_{alt}$ improved the previous results in at least 1%, achieving better mean than BoW for kNN-cosine and SVM but still presenting statistical ties. For 20-newsgroups, BoTG$_{alt}$ improved the previous results between 0.6% and 2.1%, achieving better mean than BoW for the kNN classifiers but still presenting statistical ties.

Table 4.4: Comparisons between BoTG$_{alt}$ and BoW with Macro-F_1, over the previous failure cases of BoTG in Reuters-21578.

Classifier	BoW	BoTG	BoTG$_{alt}$
kNN-Euclidean	68.8 ± 2.8	**71.7** ± 1.9 ▲	69.9 ± 2.9 ▲
kNN-cosine	88.8 ± 2.6	88.3 ± 2.4 ○	**89.3** ± 2.2 ●
kNN-Jaccard	88.0 ± 2.5	**88.5** ± 2.4 ●	**88.5** ± 2.4 ●
Random Forest	82.5 ± 3.0	**85.6** ± 2.2 ▲	84.7 ± 3.4 ▲
SVM	94.3 ± 1.1	93.7 ± 1.0 ○	**94.8** ± 0.9 ●

Table 4.5: Comparisons between BoTG$_{alt}$ and BoW with Macro-F_1, over the previous failure cases of BoTG in 20-newsgroups.

Classifier	BoW	BoTG	BoTG$_{alt}$
kNN-Euclidean	57.1 ± 1.6	55.3 ± 1.9 ▽	**57.4** ± 1.2 ●
kNN-cosine	85.5 ± 1.0	84.2 ± 1.2 ▽	**86.1** ± 0.7 ●
kNN-Jaccard	84.9 ± 0.8	84.4 ± 0.9 ▽	**85.0** ± 1.0 ●
Random Forest	78.2 ± 1.7	**78.4** ± 1.1 ●	78.2 ± 1.0 ●
SVM	86.5 ± 1.3	89.0 ± 0.9 ▲	**89.9** ± 1.2 ▲

4.3.2 Efficiency Evaluation

The vector samples produced by BoTG has dimensionality at least 50% less than in BoW for the most executions performed. This result enables a fast classification time with our method, as shown in Table 4.6. This table presents the mean time to classify one test sample during online phase, with kNN using k=1 and the Euclidean distance (for BoW and BoTG), and kNN with MCS distance for the relative-frequency model. The results refers to the average of 10 runs. Despite our method requires an extra overhead to project the test sample onto vector space, this does not compromise the benefit obtained from using less dimensions. Besides that, our method achieved classification times usually around 10% of the scores observed for the method based on relative-frequency. We confirmed statistical difference between BoTG's classification time compared to the baselines, using Wilcoxon-signed rank test paired on the 10 average results.

Figure 4.1 presents the results of BoTG and BoW regarding efficiency, as in Table 4.6, but also reflects effectiveness from the models. In this chart, the points correspondent to BoTG results are mainly in the superior left corner, which yields better effectiveness levels with lower computational cost. The accuracy and efficiency levels achieved by our model, in different collections and compared to BoW and graph-based models, shows that BoTG promotes a good balance between efficiency, effectiveness, and applicability.

Table 4.6: Classification time (in milliseconds) by sample over the collections, for kNN with the Euclidean and MCS distances.

	BoW		Relative-frequency	BoTG	
Collection	Dimensions	Time	Time	Dimensions	Time
Reuters-21578	7241	31.0 ± 0.3	435.0 ± 7.8	**3339**	**16.0** ± 0.1 ▲
4-universities	16150	82.0 ± 0.9	4104.7 ± 70.1	**6892**	**42.0** ± 0.2 ▲
20-newsgroups	29860	354.7 ± 5.4	1768.0 ± 39.5	**13900**	**187.3** ± 1.3 ▲
K-series	14358	32.0 ± 0.8	2088.0 ± 29.0	**7199**	**30.7** ± 0.5 ▲

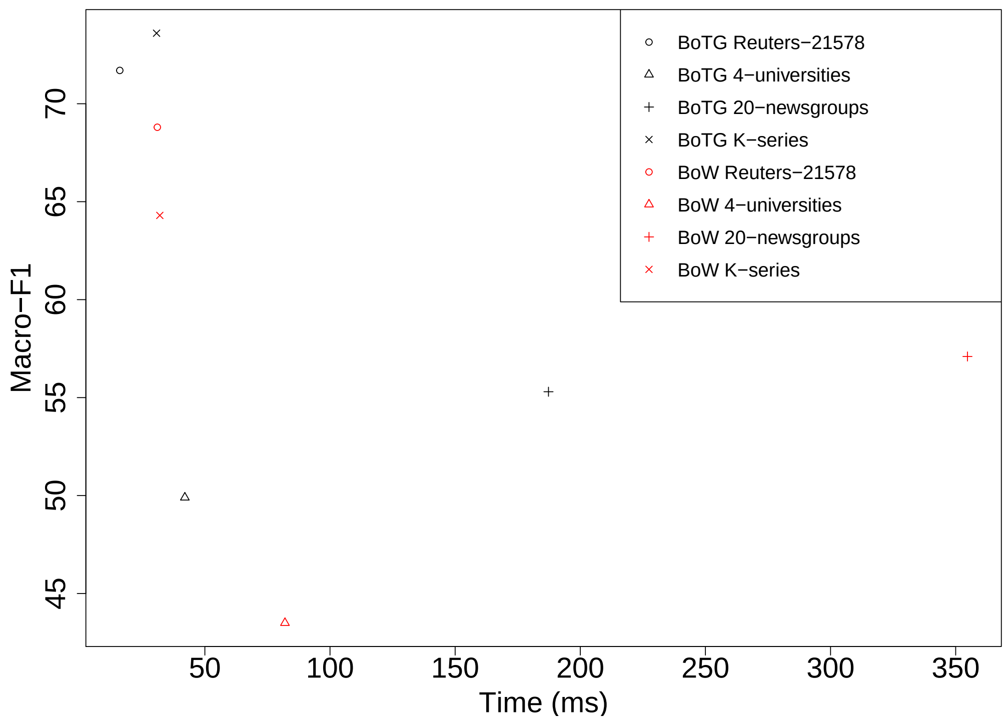


Figure 4.1: BoTG and BoW models compared by effectiveness and efficiency, for four text collections. Each collection is identified by a marker, and each model is identified by a color.

Although the BoTG presents an overall good performance during its use, the main effort in our model regards the offline stage, i.e. the vocabulary creation. This, however, is performed only once, per collection. In our experiments, this step had required an average of up to 12 hours per collection, in a regular computer with Intel Core i5-3317U CPU @ 1.70GHz and 8 gigabytes of memory.

4.3.3 Parameter Evaluation

Comprising the parameters of the model, *sum* pooling and *hard* assignment performed better in most cases than their alternative functions discussed earlier, as shown in Table 4.7. This table presents the F_1 values achieved for BoTG over the collections and using different assignment and pooling functions. The use of TF-IDF weighing in the graph extraction step was better than the use of normalized TF, which is actually an improvement of our graph formulation compared to the original proposal. In kNN, small values of k performed better, usually with $k = 1$ and $k = 3$. Jaccard and cosine distance functions performed better for kNN than Euclidean distance, and this is in accordance with results in tasks involving text collections or multidimensional data [24].

In the vocabulary creation during the offline step, the use of larger segment sizes on subgraph extraction, such as 3 and 5, as well as the use of more seeds to MeanShift improved the results in general up to a certain degree, because this tends to increase the resulting dimensionality for the vector space model. Higher dimensionalities increase the generality of the model, conducting to more representative features. It is important to mention that not all sub-graphs from the training set need to be used for the vocabulary creation. In fact, it is possible to select, for example, only sub-graphs whose its central term weight is higher than a certain threshold.

Table 4.7: Impact of different assignment and pooling functions.

	Hard Assignment			Soft Assignment		
Collection	Sum	Avg	Max	Sum	Avg	Max
Reuters-21578	**93.7** $\pm$ 1.0	91.4 $\pm$ 1.7	93.6 $\pm$ 0.9	93.5 $\pm$ 0.9	91.4 $\pm$ 1.5	93.5 $\pm$ 0.9
4-universities	**76.0** $\pm$ 1.8	63.5 $\pm$ 1.5	**76.0** $\pm$ 1.8	75.8 $\pm$ 1.8	63.9 $\pm$ 1.6	75.8 $\pm$ 1.8
20-newsgroups	88.2 $\pm$ 1.2	88.9 $\pm$ 1.0	88.1 $\pm$ 1.1	88.1 $\pm$ 1.1	**89.0** $\pm$ 0.9	88.1 $\pm$ 1.1
K-series	**99.5** $\pm$ 0.6	98.1 $\pm$ 0.9	**99.5** $\pm$ 0.6	**99.5** $\pm$ 0.6	98.2 $\pm$ 0.8	**99.5** $\pm$ 0.6

Chapter 5

Validation in Textual Retrieval Tasks

This chapter presents the evaluation of the proposed model in text retrieval tasks. In text retrieval scenario, the main goal is, for a text given as a query, to return a ranked list containing the most relevant collection documents, preferably in the first positions.

5.1 Datasets and Baselines

We adopt here the same collections of our previous validation for text classification, as we have detailed in Section 4.1.

In real-world text retrieval tasks, efficiency is usually a crucial factor due to the need of real-time low latency between a user's query and the presentation of the retrieved list. In fact, search engines perform many sorts of optimizations, such as indexing, caching, and approximations [3] to return as fast as possible relevant results without actually composing the ranked lists from the whole dataset in brute-force manner, for example. Because of this practical efficiency restriction, we limit our retrieval experiments to the comparison between BoTG and BoW, leaving graph-matching-based out in this case because they can not be applied directly in most real-world scenarios. We will not consider aforementioned optimizations in the protocol, although they can be used for BoTG if desired.

5.2 Experimental Procedures

In our proposed experimental protocol, which uses labeled collections, we use text documents as queries and we consider a retrieved document as relevant to the query if it belongs to the same class of the query document, i.e., relevancy in the experiments are either 1 for relevant or 0 for irrelevant. In practice, relevancy can be measured according to a limited numerical range, but our condition here does not affect the applicability of our model.

In order to produce a ranked list for an input text, we measure its distance against the collection documents, and then return a sorted list varying from the closest documents to the farthest. This approach relies on which text representation model and distance function are being applied, but the general procedure is the same while evaluating different representation models regardless these two conditions.

For the compared use of BoTG and BoW in retrieval tasks, we adopt one different vector-based distance function at a time: cosine, Jaccard, and Euclidean. The comparison by the use of the same distance function guarantees that only representations models are actually being compared.

For a given distance function, we compare the results from the two models using Normalized Discounted Cumulative Gain (NDCG) [25] for different ranked list sizes. NDCG@k denotes the measurement obtained for size k. NDCG is a well-known metric for ranking quality evaluation and was chosen here because it presents some advantages over other commonly used metrics such as Precision or Mean Average Precision (MAP). NDCG rewards relevant documents in the top ranked results more heavily than those ranked lower and allows graded degrees of relevance, not only binary relevance.

NDCG is a normalized version of Discounted Cumulative Gain (DCG) (Equation 5.1), to avoid that different ranked list sizes affect the comparisons (Equation 5.2). DCG is measured for the query q analyzing its ranked list up to the position k. In Equation 5.1, $rel(q, i)$ measures the relevance of the i-th element for q. In Equation 5.2, *Ideal DCG* (IDCG) is the maximum possible DCG for query q comprising all possible results to it, which is theoretically obtained if we sort its ranked list placing the most relevant results first.

$$DCG(q,k) = \sum_{i=1}^{k} \frac{2^{rel(q,i)} - 1}{log_2(i+1)} \tag{5.1}$$

$$NDCG(q,k) = \frac{DCG(q,k)}{IDCG(q)} \tag{5.2}$$

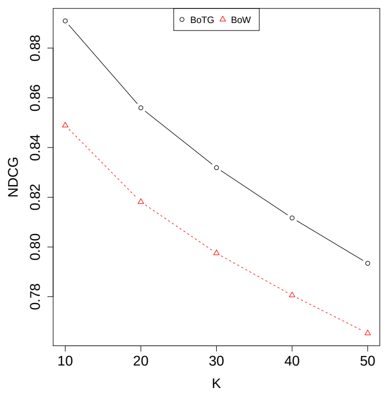
NDCG is measured for each query, so we compare the mean NDCG of each representation model. We also perform statistical tests using Wilcoxon-signed rank test paired by query. However, because p-value tends to zero for a large amount of paired samples [9], we also analyze confidence interval (CI): given D the difference between two population means, CI is the interval such that there is statistical difference when D is out of CI, considering a confidence level of 95%. We say that there is statistical difference when these two conditions occurs.

For the codebook construction in BoTG for these experiments, we trained the model taking the collection as training and using the parameter values that performed better in classification experiments.

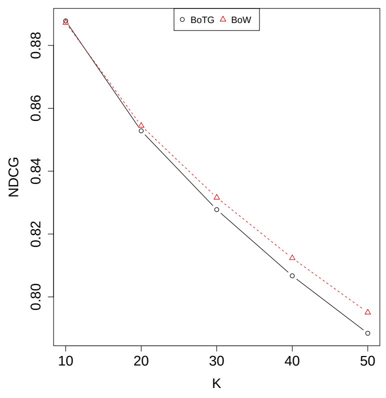
We also performed experiments with BoTG$_{alt}$ for retrieval tasks, as we did in the classification experiments. The results, however, were not better than those observed for BoTG. We believe that higher dimensionalities produced by BoTG$_{alt}$ affected the effectiveness of the method negatively due to the curse of dimensionality [26]. We plan to investigate dimensionality reduction approaches and their effects for BoTG$_{alt}$ in future work.

5.3 Results and Analysis

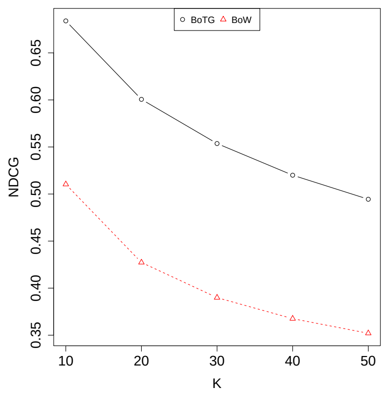
We computed NDCG for ranked lists of size k varying in 10, 20, 30, 40, and 50 to analyze how the results vary comparatively for BoTG and BoW as more documents are retrieved. Figures 5.1, 5.2, 5.3 and 5.4 present the mean NDCG results for the two models over different list sizes, for cosine, Jaccard, and Euclidean distances, respectively, for Reuters-21578, 4-universities, 20-newsgroups, and K-series. The results naturally decrease as more results are retrieved because the first positions tend to get the most relevant documents. In Reuters-21578, BoTG outperformed BoW for cosine and Jaccard distances. In 4-universities and K-series, BoTG outperformed BoW for the three distances. In 20-newsgroups, BoTG outperformed BoW for Euclidean distance but was worse for the other two distance measures.



(a) NDCG for cosine.

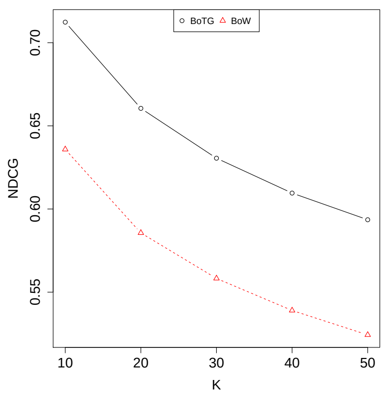


(b) NDCG for Jaccard.

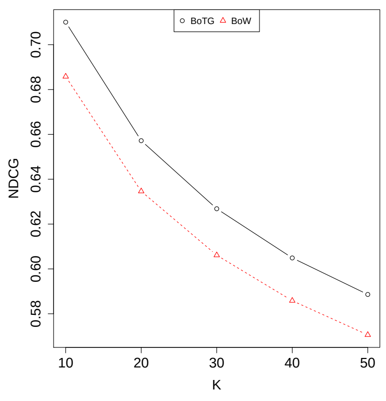


(c) NDCG for Euclidean.

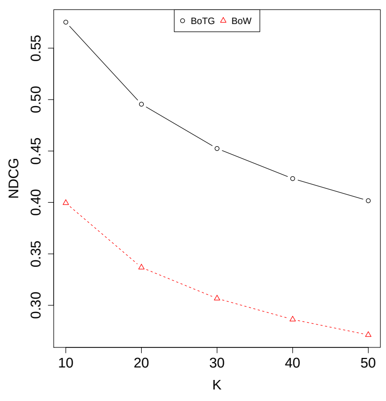
Figure 5.1: Results for text retrieval in Reuters-21578.



(a) NDCG for cosine.



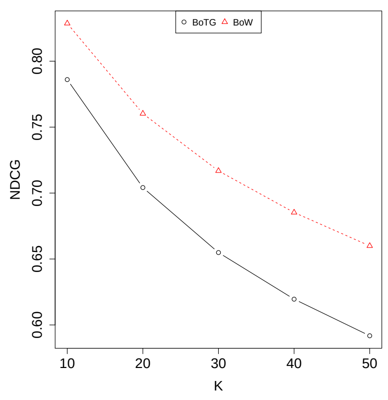
(b) NDCG for Jaccard.



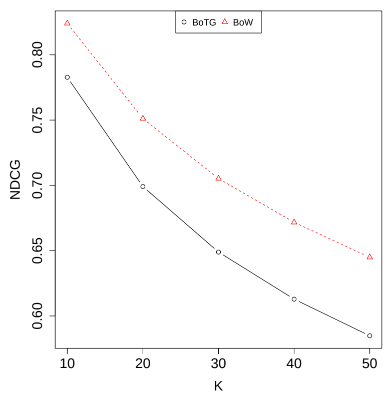
(c) NDCG for Euclidean.

Figure 5.2: Results for text retrieval in 4-universities.

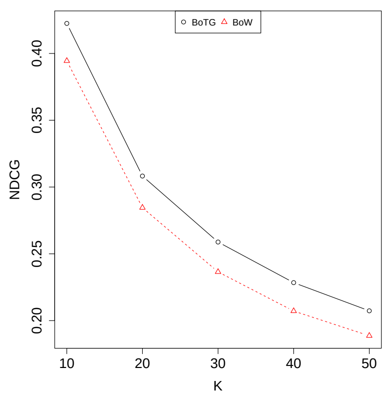
Although the comparison of NDCG curves from the models gives us a good overview, this is insufficient for detailed analysis, such as to check for statistical ties. Furthermore, due to retrieval concerns more about a relatively small number of retrieved documents, we generally fix a rank size limit, like 5, 10 or 20, to compare retrieval methods. Table 5.1



(a) NDCG for cosine.

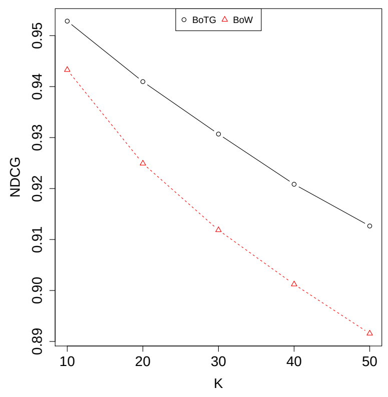


(b) NDCG for Jaccard.

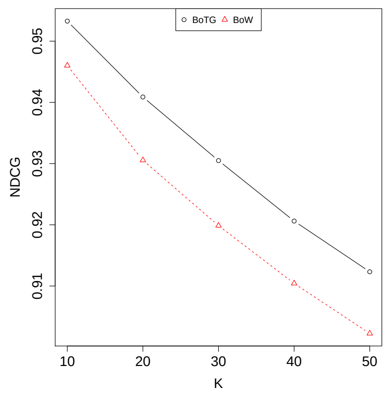


(c) NDCG for Euclidean.

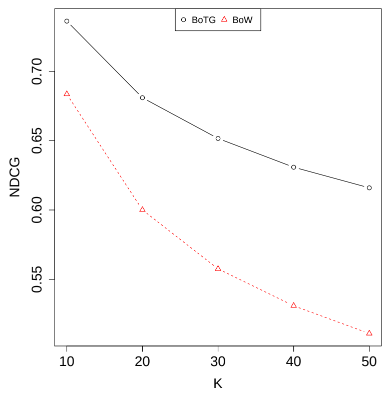
Figure 5.3: Results for text retrieval in 20-newsgroups.



(a) NDCG for cosine.



(b) NDCG for Jaccard.



(c) NDCG for Euclidean.

Figure 5.4: Results for text retrieval in K-series.

summarizes NDCG@10 and the presence or absence of statistical difference considering Wilcoxon-signed rank paired by query and confidence intervals.

Table 5.1: Comparisons between BoW and BoTG with NDCG@10, over the collections.

	cosine		Jaccard		Euclidean	
Collection	BoW	BoTG	BoW	BoTG	BoW	BoTG
Reuters-21578	84.89	89.09 ▲	88.72	88.78 ●	51.03	68.40 ▲
4-universities	63.59	71.24 ▲	68.58	71.01 ▲	39.95	57.53 ▲
20-newsgroups	82.87	78.60 ▽	82.40	78.28 ▽	39.44	42.25 ▲
K-series	94.33	95.29 ▲	94.60	95.33 ▲	68.36	73.62 ▲

In both retrieval and classification experiments, BoTG results seemed particularly worse than the baselines for the 20-newsgroups collection, as in some other minor cases. This collection has a larger proportion of less formal text than the others, based on news or documents from universities. This imposes lack of punctuation, typos etc, which compromises some premises from our model and other graph-based representation models. This reflects in a weaker BoTG vocabulary. We noticed that the best results had codebook attributes – subgraphs – with higher proportion of edges on them, which means more informative attributes. We also noticed that higher averages of DF values from the edges (pairs of terms) within the codebook promoted better results. On the other hand, none of these two aspects are directly aimed in our model. We plan to address these findings in future research.

Chapter 6

Conclusions

This research has introduced the Bag of Textual Graphs, a novel approach for text representation, which encodes textual document graph representations into vectors. The proposed approach formulates a graph-based representation model combined with a framework to project graphs into a vector space. Our objective is to combine the effectiveness of graph models to encode local contextual information with the efficiency of vector representations.

Performed experiments considering four datasets and three classifiers demonstrate that the proposed method is effective for different scenarios, which was demonstrated in document classification and retrieval tasks, yielding comparable or superior performance than baselines. The proposed method is also efficient as it is not dependent on expensive graph-matching procedures on its use during online stage.

In all datasets considered, our model showed promising results in both effectiveness and efficiency aspects. We also presented an alternative formulation for the codebook generation, as an extension of BoW, which consistently overcame this model for the few cases in which BoTG performed worse than BoW, at a cost of producing larger dimensionalities for the vector space. We emphasize that this hybrid approach remains more efficient than graph representation models.

There are relevant future work that can be addressed:

- Investigate other graph representation models as the initial component for BoTG, such as hierarchical representations or more discriminative models in general, which considers HTML markups or the presence of common sub-sentences between texts, for example [44];

- Explore the use of the proposed method in other applications such as multimodal representation of multimedia data, text clustering and text summarization. Our text representation model can be combined to image or video descriptors, composing multimodal representation models [7];

- Analyze different approaches for creating graph vocabularies, such as the use of other clustering algorithms like K-Medoids, or the use of graph clustering techniques directly [36, 46];

- Evaluate BoTG effectiveness for larger collections, such as Reuters Corpus Volume 1 (RCV1) [27] or MEDLINE [23], given that the vocabulary creation needs to process in an offline stage, over thousands of training samples. These scenarios may require us to restrict the codebook selection to a random subset, or to use approximate clustering techniques [22];

- Evaluate efficiency for BoTG in retrieval tasks, including the use of indexing and other optimization techniques, such as Locality-Sensitive Hashing (LSH) [20] or K-Dimensional Tree (KDTree) [16], to be used together with our method. This is important due to the requirement of low response time in these tasks.

Bibliography

[1] Ron Bekkerman and James Allan. Using bigrams in text categorization. Technical report, Technical Report IR-408, Center of Intelligent Information Retrieval, UMass Amherst, 2004.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689 – 694, 1997.

[5] H. Bunke and K. Riesen. Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*, 44(9):1928–1940, 2011.

[6] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3):255–259, 1998.

[7] Rodrigo Tripodi Calumby, Ricardo da Silva Torres, and Marcos André Gonçalves. Multimodal retrieval with relevance feedback based on genetic programming. *Multimedia tools and applications*, 69(3):991–1019, 2014.

[8] Tommy W S Chow, Haijun Zhang, and M. K M Rahman. A new document representation using term frequency and vectorized graph connectionists with application to document retrieval. *Expert Systems with Applications*, 36(10):12023–12035, 2009.

[9] J Cohen. Statistical power analysis for the behavioral sciences, 1988. ISSN 01621459.

[10] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (5):603–619, 2002.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] Narsingh Deo. *Graph theory with applications to engineering and computer science*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, N.J., 1974. ISBN 0-13-363473-6.

[13] G.H. Dunteman. *Principal components analysis*. Number 69. Sage, 1989.

[14] Mirtha-Lina Fernández and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6):753–758, 2001.

[15] F. Figueiredo, L. Rocha, T. Couto, T. Salles, M. A. Gonçalves, and W. Meira. Word co-occurrence features for text classification. *Information Systems*, 36(5):843–858, 2011. ISSN 03064379. doi: 10.1016/j.is.2011.02.002.

[16] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[17] J. Gibert, E. Valveny, and H. Bunke. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, 45(9):3072–3083, 2012.

[18] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4, 2006.

[19] K.M. Hammouda and M.S. Kamel. Efficient phrase-based document indexing for Web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1279–1296, 2004.

[20] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

[21] Zellig S. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[22] Mohamed Hefeeda, Fei Gao, and Wael Abd-Almageed. Distributed approximate spectral clustering for large-scale datasets. In *Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, pages 223–234, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0805-2.

[23] William Hersh, Chris Buckley, TJ Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

[24] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, pages 49–56, Christchurch, New Zealand, 2008.

[25] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[26] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer, 2011.

[27] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5 (Apr):361–397, 2004.

[28] A. Markov, M. Last, and A. Kandel. The hybrid representation model for web document classification. *International Journal of Intelligent Systems*, 23(6):654–679, 2008.

[29] Otávio A. B. Penatti. *Image and Video Representations based on Visual Dictionaries*. PhD thesis, University of Campinas, 2012.

[30] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[31] K. Riesen, M. Neuhaus, and H. Bunke. Graph embedding in vector spaces by means of prototype selection. In *Graph-Based Representations in Pattern Recognition*, pages 383–393. Springer-Verlag, 2007.

[32] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[33] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[34] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[35] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[36] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[37] Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel. *Graph-Theoretic Techniques for Web Content Mining*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2005.

[38] Sidney Siegel. *Nonparametric statistics for the behavioral sciences*. McGraw-hill, 1956.

[39] Fernanda B. Silva, Salvatore Tabbone, and Ricardo da S. Torres. Bog: A new approach for graph matching. In *Proceedings of the 2014 22nd International Conference on Pattern Recognition*, ICPR '14, pages 82–87. IEEE, 2014.

[40] Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. The use of bigrams to enhance text categorization. *Information processing & management*, 38(4):529–546, 2002.

[41] Jan C Van Gemert, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1271–1283, 2010.

[42] F. Viegas, M. A. Gonçalves, W. Martins, and L. Rocha. Parallel lazy semi-naive bayes strategies for effective and efficient document classification. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1071–1080. ACM, 2015.

[43] W.D. Wallis, P. Shoubridge, M. Kraetz, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6):701–704, 2001.

[44] Haijun Zhang and Tommy W.S. Chow. A multi-level matching method with hybrid similarity for document retrieval. *Expert Systems with Applications*, 39(3):2710–2719, 2012.

[45] Fei Zheng and Geoffrey I Webb. A comparative study of semi-naive bayes methods in classification learning. In *Proceedings of the fourth Australasian data mining conference (AusDM05)*, pages 141–156, 2005.

[46] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.

[47] Yuanyuan Zhu, Jeffrey Xu Yu, and Lu Qin. Leveraging graph dimensions in online graph search. *Proceedings of the VLDB Endowment*, 8(1):85–96, 2014.