Bag of Textual Graphs (BoTG): A General Graph-Based Text Representation Model

Ícaro Cavalcante Dourado
UNICAMP–IC, Campinas, Brazil. E-mail: icaro.dourado@ic.unicamp.br

Renata Galante
UFRGS–INF, Porto Alegre, Brazil. E-mail: galante@inf.ufrgs.br

Marcos André Gonçalves
UFMG–DCC, Belo Horizonte, Brazil. E-mail: mgoncalv@dcc.ufmg.br

Ricardo da Silva Torres
UNICAMP–IC, Campinas, Brazil. E-mail: rtorres@ic.unicamp.br

Text representation models are the fundamental basis for information retrieval and text mining tasks. Although different text models have been proposed, they typically target specific task aspects in isolation, such as time efficiency, accuracy, or applicability for different scenarios. Here we present Bag of Textual Graphs (BoTG), a general text representation model that addresses these three requirements at the same time. The proposed textual representation is based on a graph-based scheme that encodes term proximity and term ordering, and represents text documents into an efficient vector space that addresses all these aspects as well as provides discriminative textual patterns. Extensive experiments are conducted in two experimental scenarios—classification and retrieval—considering multiple well-known text collections. We also compare our model against several methods from the literature. Experimental results demonstrate that our model is generic enough to handle different tasks and collections. It is also more efficient than the widely used state-of-the-art methods in textual classification and retrieval tasks, with a competitive effectiveness, sometimes with gains by large margins.

Introduction

Accurate text mining and information retrieval (IR) 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), assigns an importance measure to terms found in the document collection, using the term frequency and its rarity in the collection. Documents are modeled as points in a hyperspace of n dimensions, called Vector Space Model (VSM), where n is the number of distinct terms in the collection. Although largely used due to its simplicity and efficiency, the BoW model does not incorporate either structural nor semantic information. Examples of useful information usually not encoded include term locality, document structural organization (for example, 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 applications, because they can represent arbitrary structures and interrelationships among different elements of a model. Some graph-based text representation models have been proposed and they are richer, as they encode both structural and semantic information (Blanco & Lioma, 2012; Hammouda & Kamel, 2004; Rousseau & Vazirgiannis, 2013; Schenker, Bunke, Last, & Kandel, 2005; Zhang & Chow, 2012). 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 IR algorithms to work on their proposed graph models. Several popular algorithms, such as support vector machine (SVM) and Random Forest, cannot be

Received November 20, 2017; revised August 29, 2018; accepted October 22, 2018

© 2019 ASIS&T • Published online Month 00, 2018 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/asi.24167

directly applied on those graph models. Furthermore, typically used graph-based matching algorithms are computationally costly.

Strategies for converting graphs to vector representations, sometimes called graph embedding techniques, have been proposed recently (Silva, Werneck, Goldenstein, Tabbone, & Torres, 2018; Zhu, Yu, & Qin, 2014) 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: they depend on how the graphs are modeled, and some functions must be explicitly defined.

Inspired by graph-based text representations and graph embedding strategies, we devise in this work an integrative generic framework for representing textual documents, regardless of the target application. In this sense, our contributions focus on the feature engineering component of textual mining solutions. As we shall see, our model allows for the use of different graph-based textual document representations that encode order, proximity, and other types of relationships among terms.

The objective is to take advantage of graphs to effectively encode local relations among terms and of vectors to support the creation of efficient applications (for example, retrieval and classification) associated with textual documents. A second contribution of this article relies on the instantiation of this generic framework for classification and retrieval applications. Finally, we introduce a new dissimilarity function, which is used to compute graph codebooks and to create bag-based representations relying on the occurrence of "graph-words" of the vocabulary.

We validate the model by performing experiments in the context of classification and retrieval tasks, considering multiple textual collections. Experimental results demonstrate that the model is more efficient and more effective than the traditional BoW, some graph-based representation models, and other baselines from the literature. In summary, our main contribution is the proposal of a general-purpose text document representation based on a graph-to-vector framework, that: (i) allows encoding document local structural information; (ii) is at the same time effective and efficient; and (iii) can be easily tailored to multiple usage scenarios.

Related Work

Some solutions in the literature directly exploit **graph-based text representations** Hammouda and Kamel (2004), for instance, proposed a document indexing model, called Document Indexing Graph (DIG) along with a similarity metric that quantifies the occurrence of common subsequences between two graphs. As the model is based on common path discovery in graphs, it is very costly. Zhang and Chow (2012) proposed a multilevel representation model for web pages that segments documents into textual sections. The comparison by sections is modeled as an optimization problem using the Earth Mover's Distance (EMD) (Rubner, Tomasi, & Guibas, 2000). As in Hammouda and Kamel (2004), this model yields a good accuracy in comparison to BoW, but at a high cost.

Schenker et al. (2005) proposed graph-based text representation models for web pages called *standard* and *relative-frequency*. The models induce distinct terms of a text as labels for vertices to the corresponding text graph, then creates oriented edges for each pair of distinct consecutive terms in the same sentence, pointing the edge from the predecessor term to the next one. We use the relative-frequency model in our proposed framework.

Graph-based models have been also proposed for document retrieval (Blanco & Lioma, 2012; Rousseau & Vazirgiannis, 2013), varying their graph definition and distance functions. Rousseau and Vazirgiannis (2013) addressed the textual document retrieval problem by proposing a scoring function, called TW-IDF (term weight - inverse document frequency), based on subgraph matching and TF-IDF (term frequency - inverse document frequency). Their method was shown to be effective, but it was not validated in text classification tasks. The focus was on the distance computation, instead of feature generation, as in our case. This possibly limits its applicability. For classification problems, it could be combined with distance-based classifiers, such as k-Nearest Neighbor (kNN). One of our baselines, referred to relative-frequency, represents this possible implementation.

Graph embeddings in vector spaces, on the other hand, rely on the conversion of graph models into vector spaces (graph embedding), enabling graph-based representations to be used on large collections and to benefit from existing mining and retrieval techniques. Riesen, Neuhaus, and Bunke (2007) and Bunke and Riesen (2011) proposed a method for mapping graphs into multidimensional space, which heuristically selects *x* graphs as training prototypes, and then maps each graph into an *x*-dimensional vector. The drawback is the need of computing, for each graph sample during the vector projection, a graph distance from it to all the prototypes. Gibert, Valveny, and Bunke (2012), in turn, proposed a function that maps graphs to vectors based on statistics of vertex attributes and edge attributes from the graph. This model, however, does not take advantage of structural information typically found in graphs, a key property exploited by our approach. It is worth mentioning that graph embedding constitutes one of the components of our solution.

Zhu et al. (2014) presented a function to map a graph database to a multidimensional space, with the advantage of preserving distance and structural characteristics. The edit distance value between two graphs tends to be equal to the distance between their corresponding resulting vectors in multidimensional space. Although this presents a theoretical advance in preserving distances, it produces vectors with high dimensionality and restricted to binary attribute weighting.

Silva et al. (2018) proposed a generic framework (BoG) for projecting samples based on graphs into an induced vector space. Given a graph training set, the goal is to generate a vector space in which the dimensions correspond to words of a vocabulary. Those words (attributes) correspond to local patterns of training graphs. This solution is generic because it works for any input graph, as long as a graph extractor—to induce a graph from an input sample—and a subgraph model coupled to a dissimilarity function are externally provided to the method. We introduce a BoG-based encoding approach in our proposed framework, extending it with an efficient, accurate, and flexible graph-based representation model.

Another line of solutions involves **vector representation of graph-based text models**. Markov, Last, and Kandel (2008) proposed a hybrid text representation model, combining graph-based representation with a Boolean vector representation. This model is applicable to web documents and restricted to classification scenarios. Chow, Zhang, and Rahman (2009) proposed a vector representation model based on the relative-frequency graph model (Schenker et al., 2005). This model is a combination of two vectors: one exploiting the use of the classical BoW with TF weighting and a second, whose attributes encode each possible pair of neighbor terms. Given that 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).

Our model presents advantages when compared with these approaches. Unlike Markov et al. (2008), we adopt more flexible *assignment* and *pooling* procedures, thus not restricting the model to binary weighting schemes. Our model is also not limited to classification tasks due to its independence of the existence of labeled train samples. Also, different from Chow et al. (2009), which encodes term occurrence and term neighborhood, our method can co-relate terms in a broader proximity context. This property makes it more accurate and more flexible than methods based on strict term neighborhood definitions.

There is also a family of models based on **co-occurrence of terms**, which vary mostly in their definition of co-occurrence. *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* up to 5 (Tan, Wang, & Lee, 2002). However, *n*-grams present higher complexity than BoW and the gains are usually either absent or marginal (Bekkerman & Allan, 2004). Skip-grams generalize *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* (Guthrie, Allison, Liu, Guthrie, & Wilks, 2006). This approach overcomes the data sparsity problem of *n*-grams, at a cost of producing even larger language models.

Figueiredo et al. (2011) proposed the notion of compound-features (*cfeatures*) for text classification tasks, a relaxed skip-bigram model whose attributes are composed of words that co-occur without restrictions on order or distance within the document. They model documents based on a vocabulary that combines BoW's attributes and feature-selected *cfeatures*, both using binary weights. This approach leads to a very high dimensionality—2^t for a collection of t terms—and relies on feature selection strategies to discard irrelevant features and reduce noise. The key idea is, for each class, to select compound-features that are discriminative to that class but not to the others. The model was compared with BoW and binary-weighted 2-grams and showed relevant effectiveness gains. The method validation, however, was limited to the text classification scenario and has efficiency issues. Anyway, we use the c-feature models as a baseline.

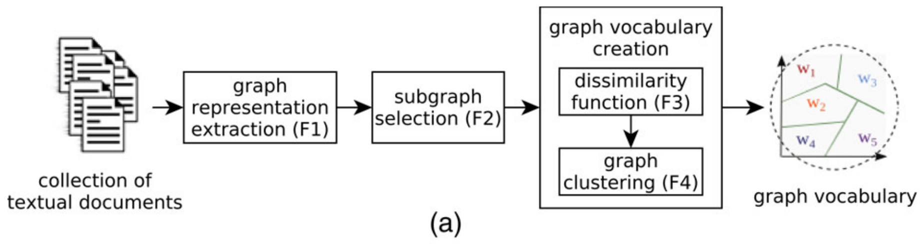
Finally, there are some **task-specific learning models**. For instance, Niepert, Ahmed, and Kutzkov (2016) present a Convolutional Neural Network (CNN) architecture for graph-based classification that works for arbitrary graph data sets. This approach, however, is restricted to the classification scenario, has not been validated in the text domain, and holds a costly training step. By contrast, our solution acts as a feature engineering for general-purpose and efficient text-mining-based applications. CNNs have also been applied to fit models for words, so that word disambiguation can now be done reliably, assuming a large enough training corpus (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). From these line of works emerged the so-called *word embeddings*, a vector representation of words that encodes semantics and relations between isolated terms. Based on word embeddings, some works emerged as for sentence classification (Kim, 2014) and text distance computation (Kusner, Sun, Kolkin, & Weinberger, 2015). These works have not yet been targeted for general applicability or efficiency.

Bag of Textual Graphs

This section introduces Bag of Textual Graphs (BoTG), a graph-based representation model for textual document feature engineering. First, we present an overview of the method highlighting its main components. Then we instantiate this model for classification and retrieval tasks. We show how to induce graphs from textual documents to create a vocabulary based on graph representations. We also present how vector representations can be obtained based on the generated vocabulary.

Conceptual View

The proposed text representation relies on the integration of a graph-based representation model with a graph-to-vector encoding scheme. 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 representation model is


collection of textual documents
graph representation extraction (F1)
subgraph selection (F2)
graph vocabulary creation
dissimilarity function (F3)
graph clustering (F4)
graph vocabulary
(a)

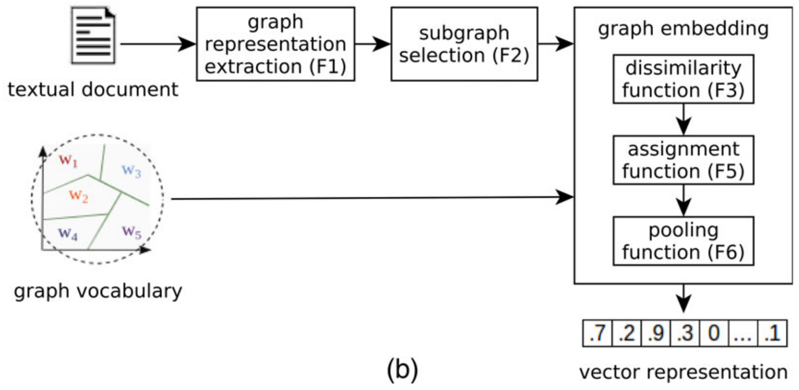


FIG. 1. Bag of Textual Graphs framework. (**a**) Pipeline for the graph vocabulary creation. (**b**) Pipeline for vector embedding based on the created vocabulary.

expected to be used in a wide range of applications such as text mining or IR tasks in general (for example, text clustering, classification, or ranking). This article focuses on text classification and retrieval problems.

Figure 1 shows the proposed pipeline employed to define vector-based representations from graphs. Two sequences of steps are presented, one for vocabulary creation (Figure 1a) and another for creating the representation given the created vocabulary (Figure 1b). Given a set of samples (collection documents), we initially extract their corresponding graphs (module *F1*). Later, subgraphs within those graphs are extracted based on predefined criteria (*F2*), and those are used to create a graph-based vocabulary to define a vector space model. The graph vocabulary creation relies on the use of a graph dissimilarity function (*F3*) and on a graph clustering method (*F4*). New textual documents are represented by vectors created based on a projection of their graphs onto the created vocabulary. Assignment (*F5*) and Pooling (*F6*) functions are used in the projection. Similar steps are employed for handling novel samples. Given a novel sample, its corresponding graph (*F1*) is extracted and key subgraphs (*F2*) are defined. Later, these subgraphs are mapped to the created vocabulary in order to define its vector representation based also on the use of specific Assignment and Pooling functions (*F5* and *F6*).

As can be observed, instantiating the BoTG model for different textual mining applications relies on the proper definition of six functions: graph representation extraction (*F1*), subgraph selection (*F2*), graph dissimilarity functions (*F3*), graph clustering (*F4*), assignment function (*F5*), and pooling function (*F6*). Those functions can be formally defined as follows:

1. Graph-representation extraction (***F1***): Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, **F1** is the function $\mathcal{P}(DO) \rightarrow (\mathcal{V} \cup \mathcal{E})$, which associates a textual document element of *d* with a vertex of $\mathcal{V}$ or an edge of $\mathcal{E}$. The **power digital object**, denoted $\mathcal{P}(d)$, is the set of all possible elements (for example, terms) of a given textual document *d*.
2. Subgraph selection (*F2*): **F2** is the function $\mathcal{P}(G) \rightarrow \{0,,1\}$, which determines if a subgraph of *G* is of interest, that is, verifies if a subgraph satisfies a property *P*.
3. Graph dissimilarity function (***F3***): F3 is implemented through the use of a graph descriptor. Let $\mathcal{G}$ be a set of graphs and $\mathcal{T}$ be a set defined as the union of the domains of vertex and edge attributes, a **graph descriptor** is a tuple (ϵ, δ) where $\epsilon : \mathcal{G} \rightarrow \mathcal{T}$ is a function that associates a graph with an element of $\mathcal{T}$ and $\delta : \mathcal{T} X \mathcal{T} \rightarrow \mathrm{IR}$ is a function that computes the similarity between two graphs. Both ϵ and δ are composite functions implemented based on vertex and edge descriptors.
4. Graph clustering (***F4***): Let $\mathcal{G}$ be a set of graphs, *F4* computes a partition on $\mathcal{G}$. The resulting sets are the clusters. A codebook, or dictionary, $\mathfrak{C} = \{w_1, w_2, \ldots, w_{|\mathfrak{C}|}\}$ is a set of words representing each group defined by a clustering.
5. Assignment function (***F5***): Let $\mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{G}|}\}$ be a set of graphs, and $\mathfrak{C} = \{w_1, w_2, \ldots, w_{|\mathfrak{C}|}\}$ be a codebook. F5 is a function that defines an activation value for each pair (g_i, w_j), where $g_i \in \mathcal{G}$ and $w_j \in \mathfrak{C}$.
6. Pooling function (***F6***): Let $\mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{G}|}\}$ be a set of graphs, $\mathfrak{C} = \{w_1, w_2, \ldots, w_{|\mathfrak{C}|}\}$ be a codebook, and f_{assign} an assignment function. A **coding** is $C = \{c_1, c_2, \ldots, c_{|\mathcal{G}|}\}$, where c_i is a vector that $c_i[j] = f_{assign}(g_i, w_j)$, where $g_i \in \mathcal{G}$ and $w_j \in \mathfrak{C}$, $1 \leq i \leq |\mathcal{G}|$, $1 \leq j \leq |\mathfrak{C}|$. Given a coding *C*, F6**:** $C \rightarrow \mathrm{IR}^N$ is a function that summarizes all word assignments, defined in a coding *C*, into a numerical vector.

Instantiating the BoTG Model

In this section, we instantiate the BoTG model, considering different implementation choices. Notice, however, that our generic model is not restricted to these choices. The instantiated model combines a graph-based representation that encodes term proximity and ordering with a vector embedding scheme to support textual document classification and retrieval.

Graph-based text modeling. The first step of our method relies on the representation of textual documents onto graphs. In the beginning, documents are preprocessed 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 published list[1] and by using Porter's stemming algorithm.

For the graph-based text modeling, we use a modified version of the relative-frequency model proposed previously (Schenker et al., 2005). In relative-frequency, a procedure called normalized TF is adopted: a value in [0, 1] is assigned for each vertex by dividing its vertex's TF value by the maximum vertex's TF in the graph; a similar procedure is performed for the edges, comprising the frequency occurrence of its term pair. Figure 2 illustrates the relative-frequency graph from a hypothetical web document with the title "YAHOO NEWS" with a link with "MORE NEWS HERE" and a text sentence "REUTERS NEWS SERVICE REPORTS. SERVICE REPORTS."

More formally, let $D = (T, S)$ be a document with term set T and sentences S, where T refers to all unique terms t from the document after stop word removal and stemming, and each $s \in S$ is a sequence of terms from T. Let $G = (V, E)$ be a directed weighted graph, where V is a set of vertices and E is a set of edges. Every vertex $v \in V$ refers to a term $t \in T$, such that v_t represents the vertex corresponding to term t and $|V| = |T|$. An edge is defined based on the co-occurrence of ordered terms within a same textual segment, so that it exists $e(x, y) \in E$, directed, that goes from v_x to v_y if term x precedes term y in any $s \in S$. We prefer our graphs to be directed in order to capture term ordering, which is semantically important.

Our graph model uses TF-IDF weighting for vertices and edges in order to incorporate information for frequency and rarity importance. 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. For a vertex v_x, TF and DF use the same definition for a term t_x as usual.

Subgraphs within G are defined as follows: a directed subgraph $G_t = (V_t, E_t)$, with $V_t \subseteq V$ and $E_t \subseteq E$ is created for each vertex v_t, where t is called the central term of G_t. There is an edge $e(x, t) \in E_t$ linking v_x to v_t if term x appears before term t within a sequence of terms of a segment, and there is an edge $e(t, y) \in E_t$ if t appears before y on same condition. Edges may be defined based on the size of the segments considered. The larger the size, the more edges are connected, leading to terms linked by a broader proximity. A large segment allows the expansion of the term context, which may lead to better accuracy results. In Figure 3, we show subgraphs extracted from the example shown in Figure 2, based on segments of size 1, that is, paths in the graph with size 1. For example, there is an edge linking the vertex related to term "yahoo" to the vertex related to term "news" in the graph represented in Figure 2. The subgraph depicted in the top left position of Figure 3 will be obtained given the restriction of path of size 1. The same is true for the other subgraphs in the figure. The main idea of this subgraph definition is to identify local features from the text to serve as feature candidates for document modeling. This representation provides more information than an n-gram, because it encodes not only proximity relations among terms, but also their relative order.

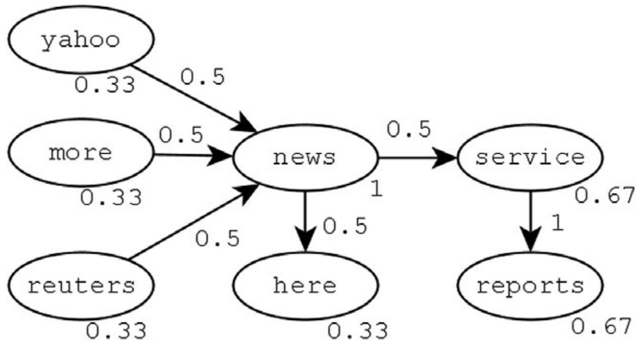


FIG. 2. A text sample represented by Schenker's models. This representation encodes into a graph a textual document with title "YAHOO NEWS" with a link with "MORE NEWS HERE" and a text sentence "REUTERS NEWS SERVICE REPORTS. SERVICE REPORTS."

Vocabulary creation. The objective of this step is the creation of a graph-based codebook onto which graphs of

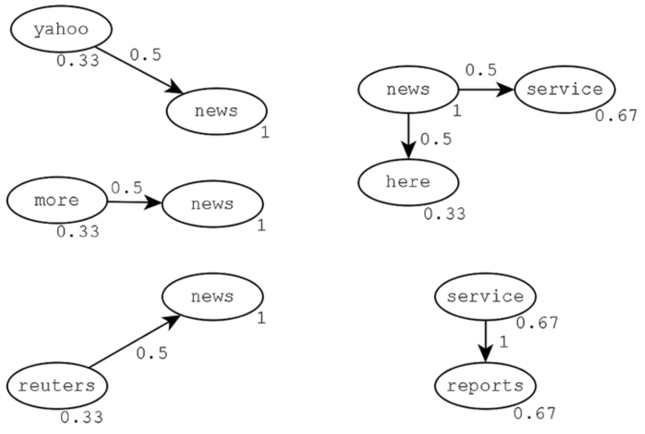


FIG. 3. Extracted subgraphs from the weighted graph depicted in Figure 2, based on segments of size 1, that is, paths of size 1.

[1] Stop list available at http://code.google.com/p/stop-words/ (as of August 2018).

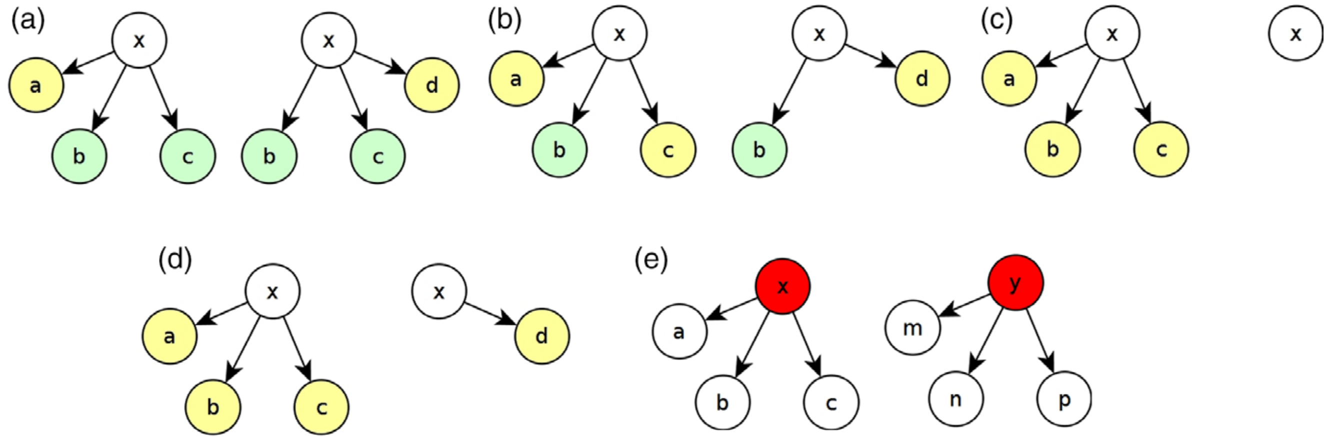


FIG. 4. Examples of dissimilarity (d) measurements for pairs of subgraphs, based on Algorithm 1. (**a**) $d=\frac{(0+0+2)}{(1+2+2)}=0.4$; (**b**) $d=\frac{(0+0+3)}{(1+1+3)}=0.6$; (**c**) $d=\frac{(0+3)}{(1+3)}=0.75$; (**d**) $d=\frac{(0+4)}{(1+4)}=0.8$; (**e**) $d = 1$.

textual documents are projected in order to define a vector representation. We present two possible approaches for codebook creation.

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 subgraphs based on their dissimilarity. For large collections, clustering of subgraphs may be a practical limit due to the large number of them, but this issue can be addressed by prior subset sampling or approximate approaches for clustering. Clustering is a more costly way, but usually produces better codebooks (Silva et al., 2018). In preliminary experiments, we noticed better results by clustering over random selection.

The clustering of subgraphs aims at selecting discriminative subgraphs from the training, from which a vocabulary set can be built. It automatically selects co-occurrence subgraphs that are either frequent or well representative of many others. Different from some other feature selection techniques, such as in Figueiredo et al. (2011), no document labeling is used during this step, which allows our method to be used in unsupervised tasks, such as document retrieval or clustering. Furthermore, our method is better than just applying simple feature selection or dimensionality reduction over BoW, because our vector representation is a concise set of attributes that are more discriminative than only terms or *n*-grams.

Different functions can be used for subgraph dissimilarity measurement. There are functions based on the maximum common subgraph, such as MCS (Bunke & Shearer, 1998) or WGU (Wallis, Shoubridge, Kraetz, & Ray, 2001). 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 moments: the subgraph clustering for vocabulary creation, and the graph embedding (see *F3* in Figure 1a,b). We emphasize that this proposed function is restricted to our subgraph definition. For understanding Algorithm 1, the following definitions are considered: G_A and G_B are connected subgraphs, 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 subgraphs and 1 for subgraphs 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 subgraphs. For an edge connecting terms present in both subgraphs, it adds the weight difference (Lines 7–9), and for all other edges it adds 1 (Lines 10–12), 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).

Algorithm 1: Proposed subgraph 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 $dist/numComparisons$.
```

Figure 4 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 the 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.*

Different algorithms can be used for graph clustering (Schaeffer, 2007; Zhou, Cheng, & Yu, 2009). In fact, the dimensionality for the vocabulary produced depends on the clustering method. In this work, we use MeanShift (Comaniciu & Meer, 2002), using a dissimilarity matrix as input. In MeanShift, the number of seeds and the bandwidth affect the final number of clusters. If desired, K-Medoids could be used to define a codebook with a specific dimensionality.

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: subgraph assignment and pooling.

An assignment function maps each subgraph to a codebook cluster (also known as word). In *hard* assignment, the subgraph 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 subgraph belonging to different clusters. Let S be a set of subgraphs of a certain input graph. The hard assignment a_{ij} for the subgraph $s_i \in S$ to the cluster (word) w_j is given by Equation 1, where N is the number of clusters and $D(s_i, w_k)$ computes the dissimilarity between s_i and w_k. Equation 2 defines a soft assignment that uses a Gaussian to smooth the dissimilarities (Van Gemert, Veenman, Smeulders, & Geusebroek, 2010), where $K(x) = \frac{exp\left(-\frac{x^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}}$, and σ allows the smoothness control.

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

$$a_{ij} = \frac{K\left(D\left(s_i, w_j\right)\right)}{sum_{k=1}^{N} K(D(s_i, w_k))} \tag{2}$$

Pooling functions summarize into a single output vector the assignments performed for all sample subgraphs. For *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 subgraphs to word j (Equation 3). *Average (avg)* pooling associates with $v[j]$ the percentage of associations to the word j (Equation 4). In *max* pooling (Equation 5), $v[j]$ is the highest value associated with the word j.

TABLE 1. Statistics of collections after preprocessing.

Collection	# classes	# samples	# terms	Mean number of nodes per graph
Ohsumed	23	18,302	31,951	59
Reuters-21578	65	8,655	19,370	41
20-newsgroups	21	10,945	64,011	69
K-series	6	2,340	25,843	175
4-universities	7	8,202	53,428	92

$$v_j = \sum_{i=1}^{|S|} a_{ij} \tag{3}$$

$$v_j = \frac{\sum_{i=1}^{|S|} a_{ij}}{|S|} \tag{4}$$

$$v_j = \max_{1 \le i \le S} a_{ij} \tag{5}$$

Validation in Textual Classification Tasks

The experiments were performed with the following standard collections for text classification: Ohsumed[2]; Reuters-21578[3]; 20-newsgroups[4]; K-series[5]; and 4-universities.[6] In the experiments regarding classification, we discard from the collections the samples that are unlabeled or multilabeled, as in Figueiredo et al. (2011), aiming at simplicity but without a 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 already. The offline stage also comprises the vocabulary generation and classification training. We call as the online stage the steps involved for an input graph, with starts by projecting it within the BoTG's vector space, and then—depending on the experimental scenario—either classifying the sample or using it as a query for retrieval.

We show in Table 1 statistics from collections after preprocessing steps and graph extraction. The collections vary in terms of their content, the number of classes and samples, and also the size of their samples, which reflects the mean number of nodes per extracted graph. All collections presented more terms than samples, which is a common characteristic in text collections.

[2] http://disi.unitn.it/moschitti/corpora.htm (As of August 2018).

[3] http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection (As of August 2018).

[4] http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups (As of August 2018).

[5] http://www-users.cs.umn.edu/boley/ftp/PDDPdata (As of August 2018).

[6] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data (As of August 2018).

We conduct experiments that compare the proposed BoTG model with the following baselines: BoW, the graph-based relative-frequency model (Schenker et al., 2005), 2-grams, and *cfeatures* (Figueiredo et al., 2011). They represent traditional and recently proposed text representations, used in several applications for encoding term co-occurrences.

The same text preprocessing procedures previously mentioned are used for BoTG and for its baselines. For BoW and 2-grams, we adopt TF-IDF weighting and rescale attribute values in the range [0, 1]. The relative-frequency model is used in the experiments following its original formulation. For the *cfeatures* model, we adopt the parameter values recommended by the authors: all terms are candidates for the compound features (vocabulary size = V), and minimum support of 2. For the *cfeatures* parameter of the *dominance* threshold, we adopted the best reported values for the collections it was used in. For the remaining collections we tested the suggested values of 50%, 60%, 70%, 80%, 90%, and 100% and report the best results.

Experimental Procedures

We use a 10-fold cross-validation experimental protocol to compare the proposed BoTG method with the baselines. We use a stratified cross-validation to impose folds with the same size and the 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 each 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 mean macro-F_1 of the test folds, which is derived from the standard measures precision and recall. We validate the effectiveness comparisons in two ways, performing statistical analysis per collection, and for multiple collections. For a per collection analysis, we use the Wilcoxon signed-rank test paired on folds to analyze whether there is statistical significance between results from our model and the one being compared, considering a 95% confidence level. As for a general statistical comparison, comprising two representation models but over all collections, we adopt Friedman's test, also considering a 95% confidence level. The efficiency evaluation is based on the mean time spent to classify a test sample.

We compared the models in this classification scenario by the using the SVM classifier for all models, except for the relative-frequency model in which we had to use kNN along with the MCS distance function (Bunke & Shearer, 1998). The choice of SVM followed a preliminary observation in which SVM had an overall superior accuracy—for all the vector-based models and text collections—when compared with other classifiers, such as Random Forest and kNN (with either cosine similarity or Jaccard distance functions). For the relative-frequency model, we use MCS (Bunke & Shearer, 1998) instead of WGU (Wallis et al., 2001) distance function, as recommended by the authors (Schenker et al., 2005).

We use SVM with linear kernel and adjusted for multi-class classification using a one-vs.-all strategy. Given the presence of hyperparameters, we use grid search internally with cross-validation in the training set in order to adjust the classifier's parameters, a procedure known as nested cross-validation (Varma & Simon, 2006). We vary C from SVM in powers of 10 between 10^{-3} and 10^{3}.

The performed experiments also evaluate different parameter settings of the proposed representation, including: (i) the number of seeds for MeanShift, varying between 10% and 100% in steps of 10%; (ii) the segment size during subgraph extraction, which we vary in $\{1, 3, 5\}$; (iii) the assignment function, varying in $\{$*Hard*, *Soft*$\}$; and (iv) the pooling function, varying in $\{$*Sum*, *Avg*, *Max*$\}$. We tune these parameters automatically with a grid search during the nested cross-validation, but we also investigate their influence.

Effectiveness evaluation. Table 2 shows the results obtained for the collections, using Macro-F_1, for BoTG and the baselines BoW, relative-frequency, 2-grams, and *cfeatures*, along with the statistical analysis among the BoTG results and those of the baselines. The symbols in the last column refer to each baseline in the same order they appear in in the table from left to right. The last line indicates how BoTG performed compared with each baseline considering the Friedman's test over all collections. We adopt the following symbols to denote the statistical comparisons of our method: symbol ▲ means that BoTG overcame the baseline; symbol • indicates that there is no significance difference from the results of the baseline, but BoTG yields better or equal mean; symbol ◦ means that there is no significance difference, and the observed mean

TABLE 2. Macro-F_1 classification results and statistical analysis for BoTG and baselines, over the collections.

Collection	BoW	relative-frequency	2-grams	*cfeatures*	BoTG	Wilcoxon tests
Ohsumed	64.9 ± 2.3	59.3 ± 1.6	64.0 ± 2.4	64.9 ± 2.4	67.6 ± 1.6	▲▲▲▲
Reuters-21578	94.3 ± 1.2	88.6 ± 2.2	93.7 ± 1.2	94.3 ± 1.2	93.9 ± 1.1	○▲●○
20-newsgroups	86.5 ± 1.4	86.8 ± 0.9	87.5 ± 1.3	84.0 ± 1.3	89.1 ± 0.8	▲▲▲▲
K-series	98.9 ± 0.6	95.6 ± 1.7	99.1 ± 0.6	99.1 ± 1.0	99.7 ± 0.4	▲▲●●
4-universities	77.3 ± 2.0	62.5 ± 1.7	79.1 ± 2.6	78.0 ± 2.2	78.2 ± 2.6	●▲○●
Friedman tests	▲	▲	▲	▲		

TABLE 3. Dimensionality, and classification time (in milliseconds) by sample, for BoTG and baselines, over the collections.

	BoW		Relative-frequency	2-grams		*cfeatures*		BoTG	
Collection	Dimensions	Time	Time	Dimensions	Time	Dimensions	Time	Dimensions	Time
Ohsumed	18003	287.1 ± 2.0	1665.2 ± 31.0	193376	2968.8 ± 46.2	451956	6829.2 ± 50.6	**4880**	**75.3** ± 0.8 ▲
Reuters-21578	7241	31.0 ± 0.3	435.0 ± 7.8	38640	154.4 ± 1.1	253516	1025.6 ± 24.1	**2076**	**10.0** ± 0.1 ▲
20-newsgroups	29860	354.7 ± 5.4	1768.0 ± 39.5	163154	1878.2 ± 30.5	2286553	27310.8 ± 216.7	**13900**	**187.3** ± 1.3 ▲
K-series	14358	34.0 ± 0.8	2088.0 ± 29.0	87729	184.3 ± 1.2	2687085	9459.2 ± 174.6	**7199**	**30.7** ± 0.5 ▲
4-universities	16150	82.0 ± 0.9	4104.7 ± 70.1	83923	406.5 ± 6.9	753827	4572.8 ± 46.6	**6892**	**42.0** ± 0.2 ▲

of BoTG is worse; and symbol △ indicates that BoTG yielded statistically worse results than the baseline.

For Ohsumed and 20-newsgroups, BoTG overcame all baselines, achieving from 2 to 5 p.p. of increase in Macro-F_1 results. Statistical significance was observed for all cases. For K-series, BoTG overcame relative-frequency baseline, and tied statistically with higher Macro-F_1 when compared with BoW, 2-grams, and *cfeatures.* For Reuters-21578, BoTG overcame relative-frequency baseline, and tied statistically with higher Macro-F_1 when compared with 2-grams. No statistical difference was observed when compared with BoW and *cfeatures.* For 4-universities, BoTG overcame relative-frequency, and tied statistically with BoW, 2-grams, and *cfeatures.*

BoTG was superior in all collections. The statistical analysis comparing our model with the baselines show it was superior against the four models. Besides the fact that relative-frequency makes part of our model, its restriction to kNN makes it less competitive than BoTG. We also observed that *cfeatures* was either better or worse than 2-grams in two of five collections. At first this seems a contradiction to the *cfeatures* author's conclusions, but our experiments use 2-grams with TF-IDF instead of binary weighting, which provides a more competitive 2-grams baseline.

In summary, BoTG yielded better or comparable results to all baselines, considering all collections. A hard data set for our method was 4-universities. For this data set, we identified that the subgraphs within the resulting vocabulary contained fewer edges when compared with the graphs considered for the other data sets (for example, for K-series 70.1% of the vocabulary contain subgraphs with edges, while for the 4-universities, only 50.5%). We believe that the adoption of strategies for using dense graphs in the construction of the vocabulary may improve the results for this data set. Besides, for supervised tasks, such as for classification, the vocabulary generation could take into account subgraph frequency, in both inter- and intraclass, for example, in order to maximize information gain. We leave these two investigations for future work.

Efficiency evaluation. The vector samples produced by BoTG has dimensionality at least 50% less than BoW for most executions performed. Compared with 2-grams and *cfeatures*, dimensionality was at least 90% and 98% smaller, respectively. This result enables a fast classification time with our method. We analyzed its performance as well as for the baselines, assuming account raw texts as input for each model, so that both model's overheads for representation and the classification procedure itself are considered for the cost computation. Although our method needs to project the test sample as a graph onto vector space, we show that this does not compromise the benefit obtained from having fewer dimensions.

Table 3 presents the mean time to classify one test sample during the online phase, with SVM for BoTG and the other vector-based models, and kNN with MCS distance for the relative-frequency model. The results refer to the average of 10 runs. It also counts the costs for text representation, although it is almost negligible. Our method achieved classification times of only around 10% of the time of the relative-frequency, 5% of the 2-grams model, and 1% of the *cfeatures*; that is, speedups of 10, 20, and 100, respectively. We confirmed the statistical difference between BoTG's classification time compared with the baselines using the Wilcoxon signed-rank test paired with

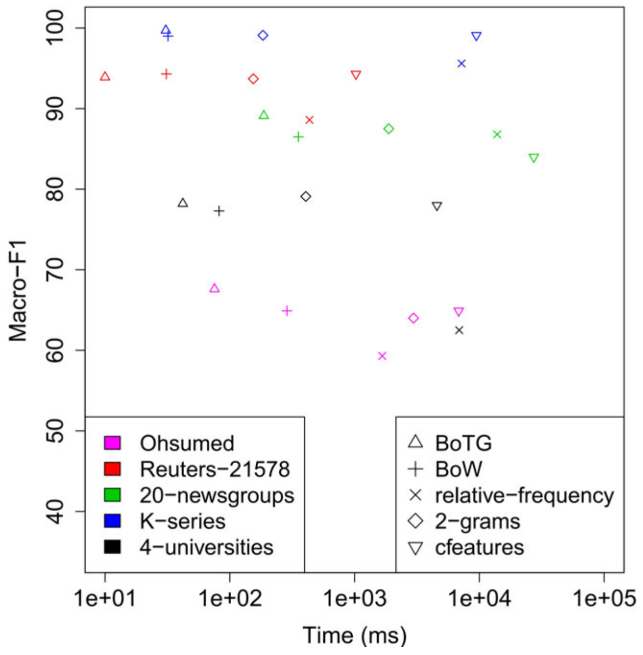


FIG. 5. BoTG compared with the baseline models in terms of effectiveness and efficiency, for four text collections. Each collection is identified by a marker, and each model is identified by a color. (**a**) Ohsumed. (**b**) Reuters-21578. (**c**) 20-newsgroups. (**d**) K-series. (**e**) 4-universities.

TABLE 4. Impact of assignment and pooling functions.

	Hard assignment			Soft assignment		
Collection	Sum	Avg	Max	Sum	Avg	Max
Ohsumed	**67.6** ± 1.6	**67.6** ± 1.6	61.4 ± 1.1	67.4 ± 1.3	67.2 ± 1.0	60.7 ± 0.9
Reuters-21578	**93.9** ± 1.1	91.4 ± 1.7	93.6 ± 0.9	93.5 ± 0.9	91.4 ± 1.5	93.5 ± 0.9
20-newsgroups	88.2 ± 1.2	88.9 ± 1.0	88.1 ± 1.1	**89.1** ± 0.8	89.0 ± 0.9	88.1 ± 1.1
K-series	**99.7** ± 0.4	98.1 ± 0.9	**99.7** ± 0.4	**99.7** ± 0.4	98.2 ± 0.8	**99.7** ± 0.4
4-universities	**78.2** ± 2.6	63.5 ± 1.5	76.0 ± 1.8	75.8 ± 1.8	63.9 ± 1.6	75.8 ± 1.8

the 10 average results. We emphasize that the BoTG efficiency benefit is not only about absolute cost values presented in Table 3, which happened to be small, but it is intrinsic to the model.

Figure 5 presents the results regarding efficiency, as in Table 3, but also reflects effectiveness from the models. In this chart, the points correspondent to BoTG results are mainly in the superior left corner, which means better effectiveness levels with lower computational cost. The accuracy and efficiency levels achieved by our model, in different collections, compared with four other baselines, shows that BoTG promotes a good balance between efficiency and effectiveness.

BoTG presents a good overall performance during its use, but its main cost regards the offline stage, that is, the vocabulary creation. This, however, is performed only once per collection. In our experiments, this step had taken at most 12 hours per collection, on an Intel Core i5-3317 U CPU @ 1.70GHz with 8GB of RAM.

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. 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 weighting in the graph extraction step was better than the use of normalized TF, which is actually an improvement of our graph formulation compared with the original proposal.

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 dimensionality increases the generality of the model, leading to more representative features. It is important to mention that not all subgraphs from the training set need to be used for the vocabulary creation. In fact, it is possible to select, for example, only subgraphs whose central term weight (for example, frequency) is higher than a certain threshold.

Impact of different subgraph selection approaches. Table 5 presents the classification effectiveness results of the proposed method in terms of the F_1 score, considering three different strategies for subgraph selection. It also presents statistical comparisons between the clustering approach to the others, using Wilcoxon signed-rank at 95% confidence level paired on folds. The methods Random and Clustering were already discussed. For Random, we vary the codebook sizes in 10,000, 20,000, 30,000, 40,000, and 50,000. The approach referred to as *Selection*, inspired by feature selection strategies, produces a codebook with the top-X best ranked subgraphs with respect to TF-IDF of their central terms, where X is the codebook size. This approach aims to emphasize the most relevant subgraphs according to their occurrence. As in the Random method, in Selection we also vary the codebook sizes in 10,000, 20,000, 30,000, 40,000, and 50,000.

As we can observe, the best results were obtained for the Clustering method, which was used in the remaining experiments. The investigation of other selection procedures, based, for example, on information gain and X^2 statistics (Forman, 2003), is left for future work. In fact, one nice aspect of our solution relies on its flexibility to support different selection methods, which might be more appropriate for different target applications.

Impact of different dissimilarity functions. Table 6 shows the classification results, in terms of the F_1 score, for different dissimilarity functions. We also present similar statistical tests comparing our method to the others. As before, we use the Wilcoxon signed-rank at 95% confidence level paired on folds. The proposed dissimilarity function (referred to as Algorithm 1) was superior for three

TABLE 5. Impact of subgraph selection strategies for vocabulary creation, in classification tasks. Results refer to the F_1 measure.

Collection	Random	Selection	Clustering
Ohsumed	60.0 ± 1.7	**68.5** ± 1.8	67.6 ± 1.6 ▲ ▽
Reuters-21578	93.4 ± 1.2	**94.2** ± 1.0	93.9 ± 1.1 ● ○
20-newsgroups	86.2 ± 1.2	88.2 ± 0.7	**89.1** ± 0.8 ▲ ▲
K-series	99.5 ± 0.6	98.8 ± 0.6	**99.7** ± 0.4 ● ▲
4-universities	75.0 ± 2.0	73.2 ± 1.2	**78.2** ± 2.6 ▲ ▲

TABLE 6. Impact of different dissimilarity functions.

Collection	MCS	WGU	Algorithm 1	Statistical tests
Ohsumed	59.6 ± 1.9	60.0 ± 1.7	67.6 ± 1.6	▲ ▲
Reuters-21578	93.5 ± 1.9	93.5 ± 1.8	93.9 ± 1.1	● ●
20-newsgroups	83.2 ± 1.3	83.7 ± 1.4	89.1 ± 0.8	▲ ▲
K-series	99.5 ± 0.6	99.5 ± 0.6	99.7 ± 0.4	● ●
4-universities	74.2 ± 2.1	74.7 ± 2.5	78.2 ± 2.6	▲ ▲

out of five collections, and tied statistically for the other two, considering MCS (Bunke & Shearer, 1998) and WGU (Wallis et al., 2001) as alternative dissimilarity functions. In Ohsumed, 20-newsgroups, and 4-universities, the gains observed were around 7, 7, and 4 percentage points, respectively.

Because local text structures are modeled as graphs, it is possible to compare them using different dissimilarity functions. However, our formulation is centered in the context of one term and its relationships to others, which are close (for example, same sentence). This dissimilarity function weights local relations among close terms in an effective way. It is also efficient, as it does not depend on costly graph matching procedures.

Also, note that our generic framework can be implemented using different dissimilarity functions, which might be more effective or more efficient for a particular application.

Validation in Textual Retrieval Tasks

We adopted the same collections of our previous validation for text classification, as we have detailed before. These collections have been used in retrieval experiments in some of our related works as well, not only for classification (Huang et al., 2016; Xie, Deng, & Xing, 2015). The fact they have labels is also useful in retrieval—although not used for the task itself—for further analysis and calculation of the quality metrics.

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 optimizations, such as indexing, caching, and approximations (Brin & Page, 2012) to return as fast as possible relevant results without actually composing the ranked lists from the whole data set in a brute-force manner. Because of this practical efficiency restriction, in our retrieval experiments, we do not consider the relative-frequency model, because graph-matching-based models cannot be applied directly in most of real-world scenarios. We will not consider the aforementioned optimizations in the protocol, although they can be used for BoTG, if desired.

For the baselines, *cfeatures* will not be considered due to its restriction to classification tasks. We then selected BoW and 2-grams as baselines to BoTG in this retrieval scenario. We also included WMD (Kusner et al., 2015), a recent work for textual document comparison based on word embedding, as an additional baseline in retrieval tasks.

Experimental Procedures

In our experimental protocol, we used text documents as queries and considered a retrieved document as relevant to the query if it belongs to the same class of the query document, since we are validating in labeled collections, that is, relevant labels in the experiments are either 1 for relevant or 0 for irrelevant. In practice, relevance can be measured according to a numerical range, but our condition here does not affect the applicability of our model. This protocol concerns document retrieval, also referred to as ad-hoc retrieval, which is a common procedure for evaluating document representation models (Huang et al., 2016; Xie et al., 2015), although it could be similarly used for shorter queries, such as for sentences or based on document titles, without loss of generality.

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.

We compared the results of two models using Normalized Discounted Cumulative Gain (NDCG) 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 emphatically than those ranked lower and allows graded degrees of relevance, not only binary relevance.

NDCG is measured for each query, so we compare the mean NDCG of each representation model. We also performed statistical tests using theWilcoxon signed-rank test paired by query. However, because the *p*-value tends to zero for a large amount of paired samples (Cohen, 1988), we also analyzed 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 occur.

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 best in classification experiments. We report the BoTG scores for the distances cosine, Jaccard, and Euclidean, referred to as $BoTG_C$, $BoTG_J$, and $BoTG_E$, respectively.

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 its baselines as more documents are retrieved. Figure 6 presents the mean NDCG results for the models, respectively for Ohsumed, 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. BoTG variations clearly outperformed BoW, 2-grams, and WMD in the five collections. One exception refers to the performance of $BoTG_E$ in the 20-newsgroups data set.

Although the comparison of NDCG curves from the models gives us a good overview, this is insufficient for a

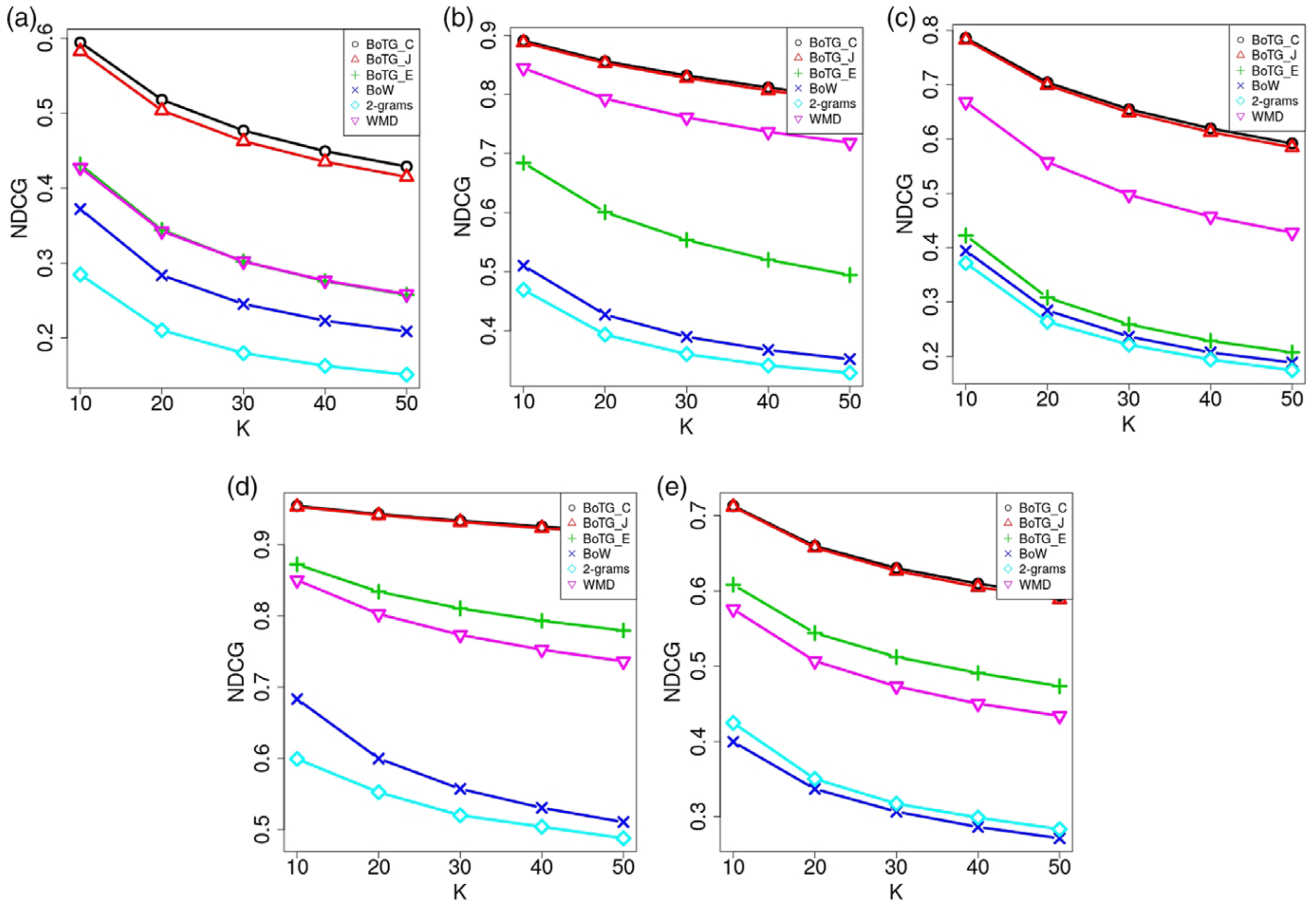


FIG. 6. Results for text retrieval with BoTG variations ($BoTG_C$, $BoTG_J$, and $BoTG_E$), BoW, 2-grams, and WMD, over the collections.

TABLE 7. Scores for retrieval tasks, with NDCG@10, over the collections.

Collection	BoW	2-grams	WMD	BoTG$_E$	BoTG$_J$	BoTG$_C$
Ohsumed	37.20	28.45	42.74	43.13 ▲ ▲ ●	58.27 ▲ ▲ ▲	59.45 ▲ ▲ ▲
Reuters-21578	51.03	46.94	84.45	68.40 ▲ ▲ ▽	88.78 ▲ ▲ ▲	89.09 ▲ ▲ ▲
20-newsgroups	39.44	37.17	66.83	42.25 ▲ ▲ ▽	78.28 ▲ ▲ ▲	78.60 ▲ ▲ ▲
K-series	68.36	59.95	85.05	87.26 ▲ ▲ ▲	95.34 ▲ ▲ ▲	95.46 ▲ ▲ ▲
4-universities	39.95	42.46	57.57	60.81 ▲ ▲ ▲	71.16 ▲ ▲ ▲	71.29 ▲ ▲ ▲

detailed analysis, such as to check 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 7 summarizes NDCG@10 and the presence or absence of a statistical difference considering Wilcoxon signed-rank paired by query and CIs. Each cell in the last three columns contains the comparisons between all variations of BoTG with BoW, 2-grams, and WMD. Our methods were superior to the baselines WMD, 2-grams, and BoW for all collections. While WMD was the strongest baseline, $BoTG_C$ and $BoTG_J$ surpass it by more than 10p.p. in most cases. WMD is able to encode texts with term weighting, using the embeddings instead of the terms, but BoTG takes into account proximity, ordering, co-occurrence, and common local term patterns, all encoded into a single representation, thus being more discriminative. Moreover, although not explored here, our embeddings are potentially more amenable to interpretation, as they correspond to occurrences of subgraphs with potentially some (semantic) meaning.

Conclusion

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 presents a graph-based representation model along with a procedure to project graphs into a vector space. It combines the effectiveness of graph models in encoding local contextual information with the efficiency of vector representations. Experiments performed considering scenarios of classification and retrieval tasks, five data sets, and several baseline models demonstrate that the proposed method is applicable for different situations, yielding comparable or superior performance than baselines in terms of both effectiveness and efficiency. The proposed method is efficient, as it is not dependent on expensive graph-matching procedures during the online stage, and also produces compact yet representative vocabularies. For future work, we plan to 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 subsentences between texts (Zhang & Chow, 2012). We also want to explore the use of the proposed method in other applications such as clustering and summarization. Another venue for investigation is the potential better interpretation of our representation to understand, explain, and improve results. Finally, we also intend to incorporate temporal information into our models and study how graph vocabularies and BoTG instances evolve over time.

References

Bekkerman, R. & Allan, J. (2004). Using bigrams in text categorization (Tech. Rep.). Technical Report IR-408, Center of Intelligent Information Retrieval, UMass Amherst, MA.

Blanco, R., & Lioma, C. (2012). Graph-based term weighting for information retrieval. Information Retrieval, 15(1), 54–92.

Brin, S., & Page, L. (2012). Reprint of: The anatomy of a large-scale hypertextual web search engine. Computer Networks, 56(18), 3825–3833.

Bunke, H., & Riesen, K. (2011). Improving vector space embedding of graphs through feature selection algorithms. Pattern Recognition, 44(9), 1928–1940.

Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters, 19(3), 255–259.

Chow, T.W., Zhang, H., & Rahman, M.K.M. (2009). A new document representation using term frequency and vectorized graph connectionists with application to document retrieval. Expert Systems with Applications, 36(10), 12023–12035.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (Vol. 2nd). Hillsdale, New Jersey: Erlbaum Associates.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. TPAMI, 24(5), 603–619.

Figueiredo, F., Rocha, L., Couto, T., Salles, T., Gonçalves, M.A., & Meira, W. (2011). Word co-occurrence features for text classification. Information Systems, 36(5), 843–858.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. Machine Learning Research, 3, 1289–1305.

Gibert, J., Valveny, E., & Bunke, H. (2012). Graph embedding in vector spaces by node attribute statistics. Pattern Recognition, 45(9), 3072–3083.

Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006). A closer look at skip-gram modelling. In Proc. lrec'06 (pp. 1–4). Genoa, Italy: European Language Resources Association (ELRA).

Hammouda, K.M., & Kamel, M.S. (2004). Efficient phrase-based document indexing for web document clustering. TKDE, 16(10), 1279–1296.

Huang, G., Guo, C., Kusner, M., Sun, Y., Sha, F., & Weinberger, K. (2016). Supervised word mover's distance. In Advances in neural information processing systems (pp. 4862–4870). Barcelona, Spain: Curran Associates Inc.

Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv, 1408, 5882.

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015). From word embeddings to document distances. In Proceedings of the 32Nd International Conference on Machine Learning (Vol. 37, pp. 957–966). Lille, France: JMLR.org.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems (Vol. 2, pp. 3111–3119). Lake Tahoe, Nevada: Curran Associates Inc.

Niepert, M., Ahmed, M., & Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In Proceedings of the 33rd International Conference on International Conference on Machine Learning (Vol. 48, pp. 2014–2023). New York, NY, USA: JMLR.org.

Riesen, K., Neuhaus, M., & Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. In Graph-based representations in pattern recognition (pp. 383–393). Berlin: Springer. Alicante, Spain.

Rousseau, F. & Vazirgiannis, M. (2013). Graph-of-word and tw-idf: New approach to ad hoc ir. In Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (pp. 59–68). San Francisco, California, USA: ACM.

Rubner, Y., Tomasi, C., & Guibas, L. (2000). The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 40(2), 99–121.

Schaeffer, S.E. (2007). Graph clustering. Computer Science Review, 1(1), 27–64.

Schenker, A., Bunke, H., Last, M., & Kandel, A. (2005). Graph-theoretic techniques for web content mining. River Edge, NJ: World Scientific Publishing.

Silva, F.B., Werneck, R.O., Goldenstein, S., Tabbone, S., & Torres, R.S. (2018). Graph-based bag-of-words for classification. Pattern Recognition, 74(Supplement C), 266–285.

Tan, C.M., Wang, Y.F., & Lee, C.D. (2002). The use of bigrams to enhance text categorization. IP&M, 38(4), 529–546.

Van Gemert, J.C., Veenman, C.J., Smeulders, A.W., & Geusebroek, J.M. (2010). Visual word ambiguity. TPAMI, 32(7), 1271–1283.

Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics, 7(1), 91.

Wallis, W.D., Shoubridge, P., Kraetz, M., & Ray, D. (2001). Graph distances using graph union. Pattern Recognition Letters, 22(6), 701–704.

Xie, P., Deng, Y., & Xing, E. (2015). Diversifying restricted boltzmann machine for document modeling. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1315–1324). Sydney, NSW, Australia: ACM.

Zhang, H., & Chow, T.W. (2012). A multi-level matching method with hybrid similarity for document retrieval. Expert Systems with Applications, 39(3), 2710–2719.

Zhou, Y., Cheng, H., & Yu, J.X. (2009). Graph clustering based on structural/attribute similarities. Proc. VLDB'09, 2(1), 718–729.

Zhu, Y., Yu, J.X., & Qin, L. (2014). Leveraging graph dimensions in online graph search. Proc. VLDB'14, 8(1), 85–96.