Shortest-Path Graph Kernels for Document Similarity

Giannis Nikolentzos
École Polytechnique and AUEB
nikolentzos@aueb.gr

Polykarpos Meladianos
École Polytechnique and AUEB
pmeladianos@aueb.gr

François Rousseau
École Polytechnique
rousseau@lix.polytechnique.fr

Michalis Vazirgiannis
École Polytechnique and AUEB
mvazirg@aueb.gr

Yannis Stavrakas
IMIS / RC ATHENA
yannis@imis.athena-innovation.gr

Abstract

In this paper, we present a novel document similarity measure based on the definition of a graph kernel between pairs of documents. The proposed measure takes into account both the terms contained in the documents and the relationships between them. By representing each document as a graph-of-words, we are able to model these relationships and then determine how similar two documents are by using a modified shortest-path graph kernel. We evaluate our approach on two tasks and compare it against several baseline approaches using various performance metrics such as DET curves and macro-average F1-score. Experimental results on a range of datasets showed that our proposed approach outperforms traditional techniques and is capable of measuring more accurately the similarity between two documents.

1 Introduction

In recent years, we have witnessed a tremendous growth in the volume of textual documents available on the Web. With this rapid increase in the number of available content, new opportunities for knowledge extraction have arisen. Many text mining tasks such as information retrieval, text categorization and document clustering involve the direct comparison of two documents. It is thus crucial to be able to determine accurately how similar two documents are by defining a document similarity measure.

Generally speaking, a similarity measure is a real-valued function that quantifies the common information shared by two objects (in our case documents). Determining the similarity between two documents is not a trivial task. Whether two documents are similar or different is not always clear and may vary from application to application.

Similarity measures that make use of the vector-space model (Salton et al., 1975) treat words in a document as if they were independent of one another, which is not realistic. In fact, words relate to one another to form meaningful phrases and to develop ideas. It is known that the human brain utilizes these relations between words to facilitate understanding (Altmann and Steedman, 1988). In general, we assume that two terms are related if they co-occur together in a small context, typically a phrase or a window of specific size, which resulted in n-gram features in many text mining tasks (an n-gram is a sequence of n terms in this paper). But n-grams correspond to sequences of words and thus fail to capture word inversion and subset matching (e. g., "article about news" vs. "news article"). To take into account these statistical relations, we propose to represent each document as a graph-of-words instead. And then, in order to measure the similarity between two documents, we capitalize on recent advances in graph kernels. Kernels can be thought of as measures of similarity between pairs of objects (Schölkopf and Smola, 2002). A graph kernel is a kernel function that measures the similarity between pairs of graphs.

Our aim in this paper is neither to define a similarity measure for only a certain category of documents based on background knowledge and features specific to that field nor to improve similarity estimation by using external knowledge. In-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1891–1901
Copenhagen, Denmark, September 7–11, 2017.

stead, we propose to define a similarity measure that does not incorporate any background or external knowledge. Hence it is, without changes, applicable to all types of textual documents even if they come from different areas. The method takes as input a pair of documents and automatically computes how similar they are to each other based solely on their content.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work and elaborates our contribution. Section 3 provides a detailed description of our proposed graph-of-words kernel. Section 4 evaluates the proposed approach on a wide range of tasks. Finally, Section 5 summarizes the work and presents potential future work.

2 Related Work

In this section, we review the related work published in the areas of *document similarity*, *graph kernels*, *kernel-based text categorization* and *graph-based text categorization*.

2.1 Document Similarity

There has been a variety of similarity measures defined to assess how close two objects are to each other, including documents. Let $< d_1, d_2 >$ be a pair of documents and D_1 (resp. D_2) the set of terms in d_1 (resp. d_2). Common similarity measures discussed by Manning (1999) are defined as follows:

$$Matching(d_1, d_2) = |D_1 \cap D_2|$$

$$Dice(d_1, d_2) = 2\frac{|D_1 \cap D_2|}{|D_1| + |D_2|}$$

$$Jaccard(d_1, d_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}$$

$$Overlap(d_1, d_2) = \frac{|D_1 \cap D_2|}{min(|D_1|, |D_2|)}$$

$$Cosine(d_1, d_2) = \frac{|D_1 \cap D_2|}{\sqrt{|D_1| \times |D_2|}}$$

The terms might be processed unigrams as well as processed n-grams present in the text. The set of operations described above are equivalent to vector operations when representing d_1 and d_2 as binary vectors.

2.2 Graph Kernels

Graph kernels are instances of the R-convolution kernels introduced by Haussler (1999). Convolution kernels have been proposed as a principled way of designing kernels on structured objects, such as sequences, trees and graphs. Graph kernels compute the similarity between pairs of graphs, based on common substructures they share. A wide variety of substructures has been proposed, such as random walks (Gärtner et al., 2003; Vishwanathan et al., 2010), shortest paths (Borgwardt and Kriegel, 2005), subtrees (Ramon and Gärtner, 2003), cycles (Horváth et al., 2004), and graphlets (Shervashidze et al., 2009).

2.3 Kernel-based Text Categorization

In recent years, there has been a great deal of work in using kernel methods, such as SVMs for text classification (Joachims, 1998; Dumais et al., 1998). Such work concentrates on building specialized kernels aimed at measuring similarity between documents. We outline some of these approaches below.

The works closest to ours are the ones reported by Lodhi et al. (2002) and by Cancedda et al. (2003). Lodhi et al. propose the use of string kernels as an alternative to the vector-space model. The feature space is generated by any ordered subsequence of characters found in the text not necessarily contiguously. Each subsequence consists of a specific number of characters and is weighted by an exponentially decaying factor of its full length in the text. Due to the enormous amount of computation needed to compute this feature vector, the authors present a dynamic programming technique, which allows the efficient calculation of the kernel values. Our work differs from theirs in that we use graph kernels instead of sequence kernels, and we concentrate on the word level instead of the character level. Cancedda et al. modified their string kernel to work with sequences of words rather than characters. Two sequences of words are considered similar if they have many common words in a given order. The similarity between two documents is assessed by the number of matching word sequences. Non-contiguous occurrences are penalized according to the number of gaps they contain. The proposed kernel is more appealing as it is more computationally efficient and it takes advantage of the standard linguistic preprocessing techniques. This approach differs in fundamental respects from our work since we represented documents as graphs-of-words in order to model word co-occurrence rather than se-

quences of words and we used a graph kernel instead of a sequence kernel to measure the similarity between pairs of documents. Other text categorization works use kernels that measure the semantic similarity between concepts extracted from the text (Bleik et al., 2013; Wang and Domeniconi, 2008).

2.4 Graph-based Text Categorization

Our work is also related to methods that represent documents as graphs and perform graph mining tasks to achieve improved classification performance. These methods either extract frequent subgraphs which are then used to produce feature vectors for the documents (Jiang et al., 2010; Rousseau et al., 2015) or they determine term weights to be used in the vector-space model based on centrality criteria or random walks (Hassan et al., 2007; Malliaros and Skianis, 2015).

3 A Graph Kernel for Document Similarity

In this section, we first discuss the essential definitions from graph theory. We then present our graph-of-words model for representing textual documents. And finally, we define our custom Shortest-Path Graph Kernel (SPGK) capable of measuring the similarity between pairs of documents.

3.1 Graph Concepts

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph consisting of a set $\mathcal{V}$ of vertices and a set $\mathcal{E}$ of edges between them. In this paper, we will denote by n the number of vertices and by m the number of edges.

A labeled graph is a graph with labels on vertices and/or edges. Given a set of labels $\mathcal{L}$, $\ell : V \rightarrow \mathcal{L}$ is a function that assigns labels to the and/or edges of the graph. In our case, we deal with fully-labeled graphs as labels are assigned both to vertices and to edges.

A graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A}$. The $(i, j)^{th}$ entry of $\mathbf{A}$ is 1 if the edge (v_i, v_j) between vertices v_i and v_j exists, and 0 otherwise.

A walk in a graph $\mathcal{G}$ is a sequence of vertices $v_1, v_2, \ldots, v_{k+1}$ where $v_i \in \mathcal{V}$ and $(v_i, v_{i+1}) \in \mathcal{E}$ for $1 \leq i \leq k$. The length of the walk is equal to the number of edges in the sequence, i. e. k in the above case. A walk in which $v_i \neq v_j \Leftrightarrow i \neq j$ is called a path. In other words, a path is a walk without repetition of nodes.

3.2 Graph-of-words

We chose to represent each textual document as a statistical graph-of-words, following earlier approaches in keyword extraction (Ohsawa et al., 1998; Mihalcea and Tarau, 2004) and more recent ones in ad hoc IR (Blanco and Lioma, 2012; Rousseau and Vazirgiannis, 2013) and in summarization (Meladianos et al., 2015).

The construction of each graph is preceded by a preprocessing phase where standard text processing tasks such as tokenization, stopword, punctuation and special character removal, and stemming are performed. The processed document is then transformed into an unweighted, undirected graph whose vertices represent unique terms and whose edges represent co-occurrences between the connected terms within a fixed-size window (hence the statistical denomination). The graph-of-words representation of text provides enhanced modeling capabilities compared to the bag-of-words representation. Besides the terms (vertices), it also models the relationships between them (edges). All the words present in a document have some relationships with one another, modulo a window size outside of which the relationship is not taken into consideration, and graphs are able to capture these dependencies. The extended modeling capabilities, however, come with an increase in complexity.

An example of a document represented as an unweighted undirected graph is given in Figure 1. The source text comes from Shakespeare's play "Hamlet": "to be or not to be: that is the question". For illustration purposes, only the colon is removed and no other text processing tasks are performed. The size of the window is set to 2, i. e. it captures bigram relationships. Hence, each word (vertex) is connected with an edge with its previous and its next word, if any.

3.3 Shortest-Path Graph-of-words Kernel (SPGK)

Our proposed approach measures the similarity between two textual documents by representing them as graphs-of-words, transforming these graphs into other graphs, and using graph kernels to calculate the similarity of the new graphs. Specifically, we capitalize on the shortest-path graph kernel (Borgwardt and Kriegel, 2005) and

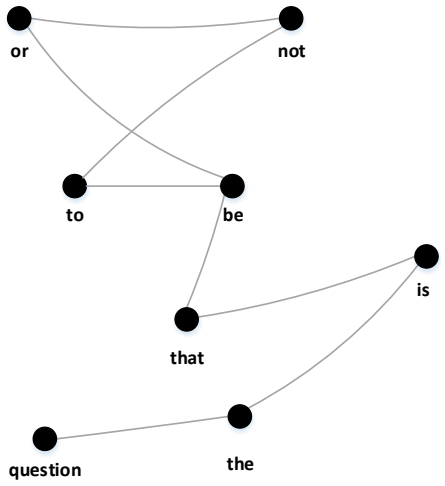

Figure 1: Example of the graph representation of a textual document.

we modify it to compare the graph representations of pairs of documents.

The first step of our proposed approach is to transform the graph-of-words representation of each document $\mathcal{G}$ into another graph $\mathcal{C}$ whose vertices are connected with an edge only if the shortest distance between them is not greater than a variable d. The emerging graph contains the same set of vertices as the graph-of-words from which it was generated. However, there exist edges only between vertices that are connected by a path of length at most d. Every node in $\mathcal{C}$ is labeled by the term that it represents, while every edge between two vertices is labeled by the shortest distance between these vertices given that it is no greater than d. Specifically, the label of an edge e that links two vertices whose shortest path is p is set equal to $label(e) = 1/p$. For $d = 1$, the emerging network is equivalent in a structural sense to its corresponding graph-of-words. For greater values of d, it is very likely that the number of edges of the graph will have increased compared to its predecessor.

The commonly-used unigram bag-of-words representation assumes that words in a document are independent of one another. Although similarity measures based on this assumption have shown to work well in practice in many fields, it is not rational to completely ignore word order and word dependence. Hence, the distance between two terms in a document determines their relationship. This led us to explore alternative document similarity metrics that take into account the co-occurrence of words in the documents. More specifically, we assume that two terms are related given that they appear together inside a window. The underlying assumption is that each word present in a document has some relationship with the other words that are close to it. We set the size of the window over the processed text equal to 2. Therefore, in our graph-of-words representation of a document, each term is linked with its preceding and its following term with an edge. In our transformed graphs, terms are not only connected with terms that are next to them, but also with terms that are close to their neighbors ($d = 2$), with lower label values, and close to neighbors of their neighbors ($d = 3$), with even lower label values. Parameter d determines how far from the initial terms we allow the paths to go. Our intuition is that given an initial term, terms that are close to terms that are close to the initial term or beyond, may have also some relation with the initial term, and the strength of this relation decreases as the shortest path length increases. Therefore, although the proposed kernel does not incorporate any knowledge of the language being used, it does capture some statistical information and is thus capable of outperforming metrics based on the unigram and even n-gram vector-space model.

To determine the edge labels in the new graph $\mathcal{C}$, we can perform depth-first search (DFS) or breadth-first search (BFS) traversals from each vertex in the graph, limiting the depth to d. The complexity for calculating paths of length up to d from a source vertex to all other vertices using either DFS or BFS is at most $\mathcal{O}(b^d)$, where b is the average branching factor. The branching factor depends on the average degree of the vertices of the graphs-of-words $\mathcal{G}$ which, in its turn, depends on the selected size of the sliding window. For $W = 2$, the average degree of the vertices will be typically only slightly above 2 and the branching factor will be only slightly above 1. Calculating paths of length up to d for all vertices takes thus $\mathcal{O}(nb^d)$ time. This still yields reasonable time complexity estimates for small values of d.

After our original graphs have been transformed into the graphs described above, we can measure their similarity using the following kernel:

Definition 1 (Custom shortest-path graph kernel)**.** *Let $\mathcal{G}_1$, $\mathcal{G}_2$ denote two graph-of-words representations of two textual documents d_1, d_2 that are*

transformed into graphs $\mathcal{C}_1$, $\mathcal{C}_2$ through the process described above. The proposed Shortest-Path Graph Kernel (SPGK) on $\mathcal{C}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{C}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ is defined as follows:

$$k(d_1, d_2) = \frac{\left(\begin{array}{l}\sum_{v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2} k_{node}(v_1, v_2) \\ \quad + \sum_{e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2} k^{(1)}_{walk}(e_1, e_2)\end{array}\right)}{norm} \tag{1}$$

where k_{node} is a positive definite kernel for comparing two vertices, $k^{(1)}_{walk}$ a positive definite kernel for comparing two edge walks of length 1 in $\mathcal{C}$ (i. e. up to d in $\mathcal{G}$) and $norm$ a normalization factor described next.

The similarity value generated by our custom shortest-path graph kernel is equal to the sum over the kernel values of all pairs of vertices on the transformed graphs plus the sum over the kernel values of all pairs of edge walks of length 1 over a positive normalization factor. The k_{node} kernel is a function for comparing two vertices. In practice, we use a delta kernel defined as:

$$k_{node}(v_1, v_2) = \begin{cases} 1 & \text{if } \ell(v_1) = \ell(v_2), \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

but other works have considered distances in word embeddings for instance to account for word similarity at the cost of having to compare every node of a graph to every other nodes of the other graph (Srivastava et al., 2013).

The normalization factor is introduced because the nominator of the proposed kernel depends on the length of the compared documents. Specifically, given the adjacency matrices of the transformed graph representations of two documents $\mathbf{A}_1, \mathbf{A}_2$ where the value of each entry in the adjacency matrix is set equal to the label of the corresponding edge, and the diagonal matrices $\mathbf{D}_1, \mathbf{D}_2$ with diagonal entries set to 1 if the corresponding term exists in the corresponding document, we first compute the matrices M_1, M_2 as shown below:

$$\mathbf{M}_1 = \mathbf{A}_1 + \mathbf{D}_1$$

$$\mathbf{M}_2 = \mathbf{A}_2 + \mathbf{D}_2$$

and we then compute the normalization factor using the following formula:

$$norm = \|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F$$

where $\|\cdot\|_F$ is the Frobenius norm for matrices.

The $k^{(1)}_{walk}$ kernel can be expressed as the product of kernels on vertices and edges along the walk. Only walks of length 1 in $\mathcal{C}$ are considered, therefore, $k^{(1)}_{walk}$ can be calculated in terms of the original vertex, the destination vertex, and the edge connecting them.

Definition 2 (Custom edge walk kernel)**.** *Let u_1, v_1 be two vertices of graph $\mathcal{C}_1$ ($u_1, v_1 \in \mathcal{V}_1$) and e_1 the edge connecting them. Let also u_2, v_2 be two vertices of graph $\mathcal{C}_2$ ($u_2, v_2 \in \mathcal{V}_2$) and e_2 the edge connecting them. The edge walk kernel is defined as follows:*

$$k^{(1)}_{walk}(e_1, e_2) = k_{node}(u_1, u_2) \times k_{edge}(e_1, e_2) \times k_{node}(v_1, v_2) \tag{3}$$

where k_{node} is the kernel function defined above and k_{edge} is a kernel function for comparing two edges defined as follows:

$$k_{edge}(e_1, e_2) = \begin{cases} \ell(e_1) \times \ell(e_2) & \text{if } e_1 \in \mathcal{E}_1 \wedge \\ & e_2 \in \mathcal{E}_2, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The measure of similarity between two graphs depends on the kernel values corresponding to the vertices and edges that compose each walk, while the matching between two vertices or two edges is determined by comparing their labels. The values of our kernel function lie in the interval $[0, 1]$. It takes a value equal to 0 for documents with no common terms and a value equal to 1 for identical documents.

Lemma 1. *SPGK is a valid kernel.*

Proof. Based on the proofs presented in (Borgwardt and Kriegel, 2005) and (Borgwardt et al., 2005), we show that our custom shortest-path graph kernel is positive definite. The k_{node} kernel is a delta kernel, which is known to be positive definite (Schölkopf and Smola, 2002) and therefore a valid kernel. The k_{edge} kernel is also a delta kernel multiplied by a positive real number. Since the multiplication of a kernel by a positive constant preserves positive definiteness, this kernel is also valid. Regarding the $k^{(1)}_{walk}$ kernel, it is positive definite as the point-wise multiplication of positive definite kernels (k_{node}, k_{edge}) preserves positive definiteness. The $\sum_{v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2} k_{node}(v_1, v_2)$ function is the sum of valid kernel functions,

hence, it is also positive definite. Regarding the $\sum_{e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2} k^{(1)}_{walk}(e_1, e_2)$ function, it is a walk kernel that takes into account only walks of length 1 on the transformed graphs and is zero-extended to the whole set of pairs of walks that do not satisfy the above constraint. Therefore, kernel values for walks with length greater than 1 are set to zero. This zero-extension is known to preserve positive definiteness (Haussler, 1999). This function is a convolution kernel, which is proven to be positive definite (Haussler, 1999). Finally, the kernel is divided by a positive constant and its positive definiteness is preserved. □

3.4 Run Time Complexity

We now determine the time complexity of our proposed kernel for measuring the similarity between two documents. Let us assume that the graph-of-words representations of the two documents consist of n vertices each. To determine the shortest paths of length at most d from a root vertex to all other vertices, we need $\mathcal{O}(b^d)$ time when using a graph traversal algorithm (depth-first or breadth-first search). There are also n vertices in the transformed graph, hence, the transformation will require $\mathcal{O}(nb^d)$ time for each graph. In order to determine the kernel value, it is necessary to compute the value of $k^{(1)}_{walk}$ for all pairs of edges between the two transformed graphs. The number of edges in the transformed graph can be at most n^2 in the case all the shortest paths in the original graph are no longer than d. Thus, there are at most $n^2 \cdot n^2 = n^4$ pairs of edges. However, due to the label enrichment that has been applied to the vertices of the transformed graphs, the number of matching nodes in the two graphs has been radically reduced and the number of pairs of edges that have to be considered is also reduced. Specifically, we have to consider n^2 pairs of edges as only paths between vertices whose label is the same in the two graphs are considered. The kernel value can thus be computed in $\mathcal{O}(n^2 + nb^d)$ time.

3.5 Alternative Computation Method for $d = 1$

In the case we consider only the common paths of length 1, there is a more efficient algorithm to compute the kernel values. The common paths of length 1 correspond to common edges between the graph representations of the documents. The emerging kernel takes into account the number of common vertices (terms) between the two graphs and the number of common edges (terms co-occurring in the same window) as well. More specifically, given two documents d_1 and d_2, the adjacency matrices of their graph representations $\mathbf{A}_1, \mathbf{A}_2$ where each entry in the adjacency matrix is set to 1 if the corresponding edge exists in the graph and the diagonal matrices $\mathbf{D}_1, \mathbf{D}_2$ with diagonal entries set to 1 if the corresponding term exists in the document, we first compute the matrices $\mathbf{M}_1, \mathbf{M}_2$ as described previously and then we compute the kernel value using the following formula:

$$k(d_1, d_2) = \frac{\sum \mathbf{M}_1 \circ \mathbf{M}_2}{\|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F} \quad (5)$$

where $(\cdot \circ \cdot)$ is the Hadamard or element-wise product between matrices.

If n is the number of unique node labels, i. e. the length of the vocabulary, and m the number of edges, the computation of the kernel values requires $\mathcal{O}(n + m)$ time in the worst case scenario. For the baseline similarity measures, with unigram features, the computational cost is $\mathcal{O}(n)$ time but it goes up as we consider higher order n-grams.

4 Experiments and Evaluation

In this section, we present the experiments we conducted to evaluate and validate our proposed kernel between documents.

4.1 Evaluation Metrics

To assess the effectiveness of the different approaches, we employed a set of well-known evaluation metrics inherited from Information Retrieval: *accuracy*, *macro-average F1-score* and for the story link detection task *DET curves* (Martin et al., 1997).

The DET curve is a variant of the ROC curve that plots the *missed detection probability* ($P_{miss} = {}^{fn}/_{(tp+fn)}$) versus the *false alarm probability* ($P_{fa} = {}^{fp}/_{(tn+fp)}$) for various system operating points, which allows someone to get a greater insight into the effectiveness of the evaluated approaches. A method is considered to perform best at thresholds that correspond to points that are close to the lower-left of the graph (i. e. lower error probabilities) and the area under the curve should be minimal.

For the story link detection experiments, we also computed the normalized C_{Det} costs, the

Dataset	# training examples	# test examples	# classes	vocabulary size	avg. terms per document	avg. degree
WebKB	2,803	1,396	4	7,772	77.93	2.54
News	32,604	CV	7	34,131	25.57	2.08
Subjectivity	10,000	CV	2	21,335	20.74	2.08
Amazon	6,400	1,600	4	39,133	86.96	2.67
Polarity	10,662	CV	2	18,777	18.44	2.00

Table 1: Summary of the 5 datasets that were used in our text categorization experiments.

standard performance measure of TDT as described in (Fiscus and Wheatley, 2004).

4.2 Datasets

We evaluate the SPGK and the baselines on 5 standard datasets for text categorization: (1) `WebKB`: Web pages collected from Computer Science departments of various Universities manually classified into 7 categories (we removed Web pages that belong to the classes "staff", "department" and "other") (Craven et al., 1998). (2) `News`: News extracted from RSS feeds of popular newspaper websites classified into 7 categories based on the taxonomies of their publishing websites (Vitale et al., 2012). (3) `Subjectivity`: Subjective and objective sentences corresponding to movie reviews from Rotten Tomatoes and to plot summaries gathered from the Internet Movie Database respectively (Pang and Lee, 2004). (4) `Amazon`: Product reviews over four different sub-collections (Blitzer et al., 2007). (5) `Polarity`: Positive and negative snippets acquired from Rotten Tomatoes (Pang and Lee, 2005). Table 1 shows statistics of the datasets that were used for the evaluation. For the Story Link Detection task, we employed the TDT-5 corpus that contains stories from various newswire sources (Glenn et al., 2006; Graff and Kong, 2006). We only used the English part of the dataset for our experiments consisting of 221,306 documents.

4.3 Baselines

The similarity measure presented in this paper is best suited for settings where the concept of a predefined corpus does not exist. For example, it could find applications in plagiarism detection and in cases where independent pairs of documents must be compared to each other. In such settings, due to the absense of a corpus, we cannot learn mappings of terms to a vector space (i. e. word embeddings) or use methods that take advantage of the corpus to increase their performance. Hence, our set of baselines includes methods that take as input two documents and output their similarity.

More specifically, the performance of our proposed kernel was compared to the performances of three baseline kernels based on similarity measures between pairs of documents $< d_1, d_2 >$ in the n-gram feature space (up to 4-grams):

1. The linear kernel, which uses the *dot product* as similarity measure: $k_{dp}(\vec{d_1}, \vec{d_2}) = \vec{d_1} \cdot \vec{d_2}$ where $\vec{d}$ is the n-gram feature vector associated with the document d;

2. *Cosine*, which measures the cosine of the angle between the two vectors: $k_c(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\| \times \|\vec{d_2}\|}$ where $\| \cdot \|$ is the L_2-norm.

3. *Tanimoto coefficient* (also known as Jaccard coefficient), which measures the intersection of features divided by their union: $k_{tc}(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\|^2 + \|\vec{d_2}\|^2 - \vec{d_1} \cdot \vec{d_2}}$

In the task of text categorization, we also compared the proposed kernel against the so-called Dynamic Convolutional Neural Network (*DCNN*) which is capable of generating representations for larger pieces of text such as sentences and documents (Kalchbrenner et al., 2014) and a convolutional neural network (CNN) architecture that has recently showed state-of-the-art results on many NLP sentence classification tasks (Kim, 2014). We used two variants of the CNN: (1) a model where all words are initialized to random vectors and are kept static during training (*CNN static,rand*), and (2) a model where again all words are initialized to random vectors, but are modified during training (*CNN non-static,rand*). The second model as well as DCNN have access to the whole corpus to generate word/document embeddings. Hence, it is not fair in a sense to compare the proposed kernel against these methods.

Method \ Dataset		WebKB		News		Subjectivity		Amazon		Polarity	
		Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Dot product	$n=1$	0.9026	0.8923	0.8110	0.7764	0.8992	0.8992	0.9188	0.9188	0.7627	0.7626
	$n=2$	0.9047	0.8950	0.8091	0.7732	0.9101	0.9101	0.9200	0.9202	0.7746	0.7745
	$n=3$	0.9026	0.8917	0.8072	0.7710	0.9090	0.9090	0.9181	0.9185	0.7741	0.7740
	$n=4$	0.8940	0.8813	0.8031	0.7651	0.9039	0.9039	0.9131	0.9133	0.7719	0.7718
Cosine	$n=1$	0.9248	0.9188	0.8117	0.7766	0.9003	0.9002	0.9400	0.9400	0.7670	0.7669
	$n=2$	0.9305	0.9275	**0.8149**	**0.7797**	0.9094	0.9094	0.9413	0.9413	0.7756	0.7756
	$n=3$	0.9298	0.9259	0.8097	0.7738	0.9099	0.9099	0.9419	0.9418	0.7765	0.7765
	$n=4$	0.9248	0.9208	0.8076	0.7709	0.9076	0.9075	0.9413	0.9413	0.7753	0.7753
Tanimoto	$n=1$	0.9062	0.8983	0.8155	0.7815	0.9094	0.9093	0.9225	0.9226	0.7749	0.7748
	$n=2$	0.9040	0.8945	0.8075	0.7700	0.9061	0.9060	0.9181	0.9185	0.7735	0.7735
	$n=3$	0.9241	0.9180	0.7980	0.7575	0.9021	0.9020	0.9344	0.9347	0.7648	0.7648
	$n=4$	0.9176	0.9084	0.7899	0.7483	0.8953	0.8952	0.9300	0.9300	0.7586	0.7586
DCNN		0.8918	0.8799	0.7991	0.7615	0.9026	0.9026	0.9181	0.9181	0.7326	0.7326
CNN	static,rand	>1 day		0.7757	0.7337	0.8716	0.8715	0.8881	0.8882	0.7150	0.7150
	non-static,rand	>1 day		0.8113	0.7749	0.8961	0.8960	0.9356	0.9356	0.7654	0.7653
SPGK	$d=1$	0.9327	0.9278	0.8104	0.7749	**0.9148***	**0.9148**	0.9400	0.9401	0.7776	0.7775
	$d=2$	**0.9370***	**0.9336**	0.8089	0.7729	0.9146*	0.9146	0.9413	0.9413	**0.7789***	**0.7788**
	$d=3$	0.9291	0.9233	0.8078	0.7703	0.9137*	0.9137	0.9444	0.9444	0.7761	0.7760
	$d=4$	0.9291	0.9223	0.8097	0.7730	0.9118	0.9118	**0.9463**	**0.9463**	0.7780	0.7780

Table 2: Performance of the 6 approaches in text categorization. * indicates statistical significance in accuracy improvement at $p < 0.05$ using the micro sign test against the Cosine ($n = 2$) baseline of the same column. >1 day indicates that the computation did not finish after 1 day.

4.4 Text Categorization

To perform text categorization, for all methods except the DCNN and the two CNNs, we employed a Support Vector Machine (SVM) classifier (Boser et al., 1992). It is interesting to note that all we need to train an SVM classifier is the kernel matrix of the training examples. We optimized the parameter C of the SVM by performing 10-fold cross-validation on the training set. We then made predictions on the test set using the optimal value of C. For DCNN the dimensionality of the generated embeddings was set to 100, while for the two CNNs it was set to 300. For DCNN and the two CNNs, the number of training epochs was set to 25. All similarity measures were coded in Python[1].

For each value of the parameter d, we obtain a new kernel and in turn the resultant kernel matrix contains different values. To study the effect of parameter d on the classification performance, we performed tests for values of d ranging from 1 to 4. We did not further increase the value of d since in most cases, for values greater than 4, the performance of the classifier stayed the same.

Table 2 shows the performance of the baseline methods and the proposed shortest-path graph kernel (SPGK), on the five datasets. Bold font marks the best performance in a column, while $*$ indicates statistical significance in accuracy improvement at $p < 0.05$ using the micro sign test (Yang and Liu, 1999) against the Cosine ($n = 2$) baseline of the same column. We chose to test for significance against that measure, as it corresponds to the best-performing baseline. On all datasets except one (News), SPGK outperforms the other three similarity measures and the neural network architectures. In addition, the results show a statistically significant improvement of at least one of our kernels over the Cosine ($n = 2$) approach on all datasets except two (News, Amazon). In general, our kernel is followed in performance by Cosine, Tanimoto, Dot Product in that order. The three neural network architectures fail to outperform the proposed kernel even on a single dataset. Furthermore, the approaches that make use of the whole corpus to generate embeddings (DCNN and CNN non-static,rand) do not seem to gain any advantage from having access to the whole dataset. This may be due to the fact that the size of the datasets is not large enough for learning high-quality representations.

4.5 Story Link Detection

Story link detection, as defined by the Topic Detection and Tracking (TDT) research program (Allan, 2002), is the task of determining whether two stories, such as news articles and radio broadcasts, are "linked" by the same event. According to TDT,

[1]Code available at: http://www.db-net.aueb.gr/nikolentzos/code/spgk.zip

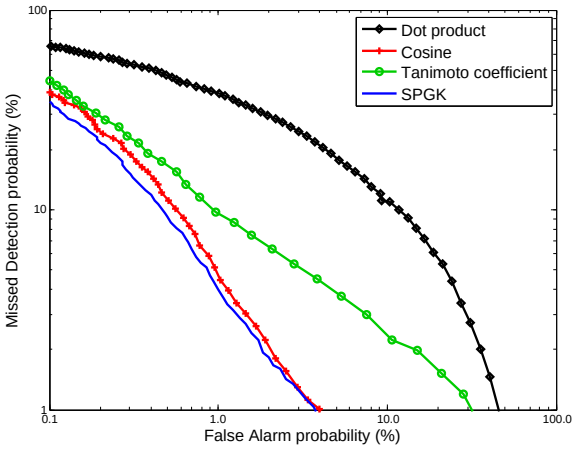

Figure 2: DET curves for all similarity measures on story link detection track.

Similarity measure		$(C_{det})_{norm}$
Dot product		0.3908
Cosine		0.0953
Tanimoto coefficient		0.1453
SPGK	$d = 1$	**0.0883**
	$d = 2$	0.0884
	$d = 3$	0.0888
	$d = 4$	0.0888

Table 3: Performance of all similarity measures in story link detection.

an event is something that happens at some specific time and place and two stories are "linked" if they discuss the same event.

In Figure 2, we plot the DET curves comparing the proposed approaches. For clarity, we only plot one curve for our SPGK approach ($d = 1$) since the plots overlapped, and the best performing curve for each of the baseline approaches. It is clear that our approach outperforms the baselines over the whole set of operating points. We also searched for the threshold values for which each approach maximizes its performance. Our next step was to compare the four systems in terms of detection effectiveness at that optimal threshold. Table 3 illustrates the normalized C_{det} of the proposed methods and the baselines. We can see that the proposed methods are better than baseline methods in terms of the normalized C_{det} metric.

5 Conclusion

In this paper, we presented a graph kernel for measuring the similarity between pairs of documents. The graph-of-words representation of textual documents allows us to model relationships between terms in documents and, hence, to go beyond the limits of the vector-space model. At the same time, it allows us to measure the similarity between two documents by comparing their graph representations using kernel functions. The effectiveness of the proposed kernel was empirically tested on two different tasks, namely text categorization and story link detection. The proposed measure showed improved performance on both tasks compared to the baselines.

References

J. Allan. 2002. Introduction to Topic Detection and Tracking. In *Topic Detection and Tracking*, pages 1–16.

G. Altmann and M. Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238.

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

S. Bleik, M. Mishra, J. Huan, and M. Song. 2013. Text Categorization of Biomedical Data Sets using Graph Kernels and a Controlled Vocabulary. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(5):1211–1217.

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, Bollywood, Boomboxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

K. M. Borgwardt and H. Kriegel. 2005. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 74—81.

K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H. Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152.

N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. 2003. Word-Sequence Kernels. *The Journal of Machine Learning Research*, 3:1059–1082.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the 10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 509–516.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management*, pages 148–155.

J. Fiscus and B. Wheatley. 2004. Overview of the TDT 2004 Evaluation and Results. In *TDT Workshop*.

T. Gärtner, P. Flach, and S. Wrobel. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*, pages 129–143.

M. Glenn, S. Strassel, J. Kong, and K. Maeda. 2006. TDT5 Topics and Annotations. *Linguistic Data Consortium (LDC)*.

D. Graff and J. Kong. 2006. TDT5 Multilingual Text. *Linguistic Data Consortium (LDC)*.

S. Hassan, R. Mihalcea, and C. Banea. 2007. Random-Walk Term Weighting for Improved Text Classification. *International Journal of Semantic Computing*, 1(04):421–439.

D. Haussler. 1999. Convolution Kernels on Discrete Structures. Technical report, UCSC-CRL-99-10, Department of Computer Science, University of California, Santa Cruz.

T. Horváth, T. Gärtner, and S. Wrobel. 2004. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167.

C. Jiang, F. Coenen, R. Sanderson, and M. Zito. 2010. Text classification using graph mining-based feature extraction. *Knowledge-Based Systems*, 23(4):302–308.

T. Joachims. 1998. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2:419–444.

F. Malliaros and K. Skianis. 2015. Graph-Based Term Weighting for Text Categorization. In *Proceedings of the 2015 International Conference on Advances in Social Networks Analysis and Mining*, pages 1473–1479.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. 1997. The DET Curve in Assessment of Detection Task Performance. Technical report, DTIC Document.

P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. 2015. Degeneracy-based Real-Time Sub-Event Detection in Twitter Stream. In *Proceedings of the 9th AAAI Conference on Web and Social Media*, pages 248–257.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Y. Ohsawa, N. E. Benson, and M. Yachida. 1998. KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor. In *Proceedings of the Advances in Digital Libraries Conference*, pages 12–18.

B. Pang and L. Lee. 2004. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 271–278.

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

J. Ramon and T. Gärtner. 2003. Expressivity versus Efficiency of Graph Kernels. In *1st International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74.

F. Rousseau, E. Kiagias, and M. Vazirgiannis. 2015. Text Categorization as a Graph Classification Problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712.

F. Rousseau and M. Vazirgiannis. 2013. Graph-of-word and TW-IDF: New Approach to Ad Hoc IR. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 59–68.

G. Salton, A. Wong, and C. Yang. 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press.

N. Shervashidze, T. Petri, K. Mehlhorn, K. M. Borgwardt, and S. Vishwanathan. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 488–495.

S. Srivastava, D. Hovy, and E. H. Hovy. 2013. A Walk-Based Semantically Enriched Tree Kernel Over Distributed Word Representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1416.

S. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. 2010. Graph Kernels. *The Journal of Machine Learning Research*, 11:1201–1242.

D. Vitale, P. Ferragina, and U. Scaiella. 2012. Classification of Short Texts by Deploying Topical Annotations. In *Advances in Information Retrieval*, pages 376–387.

P. Wang and C. Domeniconi. 2008. Building Semantic Kernels for Text Classification using Wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–721.

Y. Yang and X. Liu. 1999. A Re-examination of Text Categorization Methods. In *Proceedings of the 22nd International SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49.