SemaFor: Semantic Document Indexing using Semantic Forests

George Tsatsaronis
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
gbt@idi.ntnu.no

Iraklis Varlamis
Department of Informatics and Telematics
Harokopio University of Athens
Athens, Greece
varlamis@hua.gr

Kjetil Nørvåg
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Kjetil.Norvag@idi.ntnu.no

ABSTRACT

Traditional document indexing techniques store documents using easily accessible representations, such as inverted indices, which can efficiently scale for large document sets. These structures offer scalable and efficient solutions in text document management tasks, though, they omit the cornerstone of the documents’ purpose: meaning. They also neglect semantic relations that bind terms into coherent fragments of text that convey messages. When semantic representations are employed, the documents are mapped to the space of concepts and the similarity measures are adapted appropriately to better fit the retrieval tasks. However, these methods can be slow both at indexing and retrieval time. In this paper we propose SemaFor, an indexing algorithm for text documents, which uses semantic spanning forests constructed from lexical resources, like Wikipedia, and WordNet, and spectral graph theory in order to represent documents for further processing.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: [Retrieval models, Selection process]; H.3.1 [Content Analysis and Indexing]: [Linguistic processing, Thesauruses]

1. INTRODUCTION

Document indexing has been traditionally conducted with the use of a term to document mapping and its inverse, which takes into account only the frequency of occurrence of terms in the indexed documents. The simple, yet powerful, mechanism of inverted indexes does not consider any other type of information regarding terms, such as the semantic relatedness between terms, and their syntactic role in the document. Moreover, it is frequently the case that different meanings are conveyed in text even within the same document, as a document might address different topics.

In this paper we propose SemaFor, a new document indexing algorithm that takes into account the semantic relatedness of terms within documents. SemaFor aims at: (1) extracting information from text, namely terms, and identify their semantic connections,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM’12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.
est and then the semantic indexing and ranking module performs the similarity computation between the input semantic forest and the indexed documents. Similarity computation is performed using spectral graph algebra, as explained in detail in Section 3. Finally, documents are ranked based on the similarity values computed by this module.

2. RELATED WORK

The basic hypothesis behind our approach is that the use of semantic information for the representation of documents may improve the performance of the text clustering and retrieval tasks, both in precision and recall. The hypothesis is based on concrete scientific indications that have been published in the past, e.g., [12, 21]. The approaches of this category fall under two main schools of thinking: (a) use of statistical learning theory and machine learning techniques, which are trained on large text corpora, and (b) use of lexical resources, such as word thesauri and electronic dictionaries, in order to disambiguate document terms in their context.

The main idea behind all approaches of the first category is to group the terms of a document into subsets (topics) that contain statistically “related” terms. The terms of a group either belong in the same phrase (keyphrases [10]), or relate to the same document item (e.g., author, title [16]). In a different direction, Chang et al. [7] use probabilistic graphical models in order to group the terms of a text document in topics and represent documents as combinations of one or more topics. Buntine et al. [6] model the documents based on a hierarchy of topics built from a set of document bags, using a hierarchical version of multinomial PCA. For the partitioning of the words into topics, they use a Dirichlet distribution [4]. A major disadvantage of such approaches is that they require extensive training with large text corpora, and the produced models cannot be easily transferred across domains.

From the works that use lexical and other knowledge resources in document processing, we focus on those that aim at representing text documents as graphs of terms such as [20] that use lemma definitions, [26] that use WordNet synsets and [11] that use Wikipedia entries in order to construct a network from the terms of a sentence. According to [34] linguistic and crowdsource-based knowledge sources can be used complementary in this task.

Our work combines WordNet and Wikipedia in order to provide increased coverage and take the best of both worlds (“wisdom of linguists” and “wisdom of crowds”). The processing of document semantics in SemaFor results in a graph that contains the document terms only. Though SemaFor does not perform topic detection literally, the SSTs of each indexed document can be seen as the document topics. In addition, taking one step further to the aforementioned approaches, SemaFor indexes the document graph using a mechanism that facilitates storage and fast processing, and incorporates semantic information inside the indexing data structures.

Close to our approach are also the works that embed senses and semantic information for text document management. In this direction, Generalized Vector Space Models (GVSM) and semantic kernels for the purposes of document similarity, with application to text classification [2, 15] and text retrieval [22, 24, 28] attempt to model the semantic information of documents. The work of Henderson et al. [12] was the first approach to introduce semantic forests as defined by Schone and Nelson [20], and the results show that the use of semantic forests in information retrieval is effective.

An important point in existing approaches is the consideration of word sense disambiguation methods (WSD) which can potentially offer the transit from terms to senses. In this paper we tackle disambiguation of terms by employing a very fast, yet simple, WSD algorithm that provides state of the art performance and is used as a very competitive baseline for WSD methods; the method is called the first sense heuristic and it selects the most frequently appearing sense of each word in order to disambiguate it\(^1\). The first sense heuristic provides very high accuracy in almost every WSD benchmark data set available [18].

Finally, with regards to semantic indexing, existing methodologies that map documents to graphs using the aforementioned methodology ignore the semantic information at indexing level. In [13], each document is mapped to a graph with terms as vertices and 4 types of edges (based on WordNet relations). The graph structure is neither indexed, nor employed in the computation of similarity between documents. Instead, only the document terms with their weights are stored, with the graph aiding only in the clustering of the terms. In [29] documents are mapped to semantic forests using the co-occurrence of terms (actually stems) and their semantic relations (as given by WordNet) in order to draw semantic relations between terms. During the indexing and document similarity computation phases, the graph information is neglected and each forest is perceived as a set of terms. For the computation of the similarity between concept forests (i.e., between documents) only the percentage of the common nodes of the trees is employed.

The solution that we introduce in SemaFor is a lightweight representation of the document graph that keeps only the strongest edges of a semantic graph thus forming a spanning tree. In contrast to the aforementioned approaches, we employ spectral graph theory in order to convert the spanning trees into an indexable format (a set of points in a metric space) and a fast distance measure for measuring the distance between documents.

3. THE SEMAFOR ARCHITECTURE

In this section we present the details of the SemaFor architecture as shown in the high level representation of Figure 1. In Section 3.1, we explain the operations of the Document Processing module, which constructs the semantic spanning forests given a set of documents. Section 3.2 explains the details of the Semantic Indexing module: (a) how the semantic spanning forest is transformed into a set of points in a metric space using spectral graph theory, and (b) what information is stored in the index for each semantic spanning forest. Section 3.3 illustrates the spectral graph similarity computation module, the details of the distance metric and the algorithm employed in SemaFor for document comparison and similarity computation. Finally, Section 3.4 provides details on the document retrieval mechanism and explains how the index can be employed to provide a fast and scalable document retrieval solution.

3.1 Construction of Semantic Spanning Forests

Given a document \(D\), the semantic spanning forest construction process (Alg. 1) comprises three steps: (a) the pre-processing of the document, comprising part-of-speech tagging (POS tagging), and phrase detection, (b) word sense disambiguation, and (c) construction of the semantic spanning forest using measures of semantic relatedness.

3.1.1 Document Pre-processing

For a given document \(D\) of the collection, we initially perform POS tagging using the Stanford Part of Speech Tagger [23]. This tagger offers state of the art performance (at the levels of 95% and above), and also enables us to perform sentence splitting in the document. The set of tags produced by the tagger is mapped into the

\(^1\) In our implementation we are using WordNet as the main dictionary, or Wikipedia definitions if the term is ambiguous and does not appear in Wordnet
four basic part-of-speech (POS) tags, i.e., noun, verb, adjective, and adverb, that exist in WordNet. Once the POS for each document word is known, we perform phrase detection for recognizing terms of more than two words (e.g., “United States of America”).

The phrase recognition takes place by simple dictionary look up, and is used as a powerful, yet simple, baseline for the task[18].

In the WSD step (Step 8, Alg. 1) we use the first sense heuristic approach to disambiguate the terms into their respective sense. The first sense heuristic has shown state of the art performance in WSD and is used as a powerful, yet simple, baseline for the task[18].

3.1.2 Semantic Spanning Forest Construction using Semantic Relatedness

The algorithm of the SSF(D) construction, for a given document D is described by Alg. 1. Given that D contains a set of n term-POS pairs, namely T = tp1, tp2, ..., tp24 in the remaining of this section we describe how a semantic spanning forest is constructed from this set. Primarily, note that for any given pair (tp, tp′) with i ≠ j and both i, j ∈ [1...n] the t part of the term-pair tp may be identical with the t part of the tp′ term pair, but then the p part in the two term pairs must differ (i.e., we keep the set of all distinct term-POS pairs for D).

Initially, we compute the semantic relatedness S between every term-pair combination in T. In our implementation we are using Omiotics [25] as the semantic relatedness measure which employs WordNet2, and WLM, a Wikipedia-based measure[17]. Note that the suggested methodology is general enough to allow for the use of any other measure of semantic relatedness or similarity. Both used measures are in the range of [0, 1], with 1 meaning totally related and 0 meaning totally unrelated; they are publicly available and their performance is state-of-the-art in their category of measures [34]. Since for both measures S(tpi, tpj) = S(tpj, tpi) we need exactly \( \frac{n(n-1)}{2} \) computations of semantic relatedness. For each pair (tpi, tpj) if both tpi, tpj ∈ WordNet, we are using Omiotics, which has been shown to outperform WLM in case both terms exist in WordNet [25] (Steps 12-13, Alg. 1). In any other case, we are using WLM (Steps 14-15, Alg. 1), so as to cover the terms that do not exist in WordNet. Thus, we define \( S(tpi, tpj) = SR(tpi, tpj) \), if tpi, tpj ∈ WordNet, else \( S(tpi, tpj) = WLM(tpi, tpj) \).

After the computation of every pairwise semantic relatedness value for the pairs in T, we construct a semantic graph which initially contains all the elements of T as nodes. Each node now represents a term-POS element of D. Then, we add an edge \( e_{tpi, tpj} \) between every pair of nodes \( \{tpi, tpj\} \) for which \( S(tpi, tpj) > 0 \), with weight \( w_{tpi, tpj} = \frac{1}{S(tpi, tpj)} \), and \( i \neq j \) (Step 18, Alg. 1).

Once all the edges have been added, the semantic graph contains terms as nodes and reverse semantic relatedness values between them as edges. Then, for each connected component of the graph, we apply the computation of the minimum spanning tree algorithm of Kruskal [14] (Steps 22-23, Alg. 1).

After this computation, D is now a set of minimum semantic spanning trees. We define this set as the Semantic Spanning Forest representing document D (SSF(D)), and each i-th semantic tree of (SSF_i(D)) as one of its topics. An example of a \( SSF(D) \) of a real document D is shown in the next section.

3.1.3 An Example of Constructing Semantic Spanning Forests

In the following, we show how the full semantic graph and the graph after the computation of the SSF of a real document looks like. The following text is taken from the known CACM document collection. It is titled “Efficient Implementation of a Variable Projection Algorithm for Nonlinear Least Square Problems”, and it has document id 2670. The body of the document is as follows:

“Nonlinear least squares frequently arise for which the variables to be solved for can be separated into a linear and a nonlinear part. A variable projection algorithm has been developed recently which is designed to take advantage of the structure of a problem whose variables separate in this way. This paper gives a slightly more efficient and slightly more general version of this algorithm than has appeared earlier.”

We execute Alg. 1 on this example, and show the semantic graph G, before and after the execution of lines 22 – 24 of the algorithm. From these two graphs we can elicit important observations about the SSF of a document and the document itself. The graph in the left (we have omitted edges’ weights for simplicity of presentation) shows that some of the most interconnected are least_squares_NN, variable_NN, algorithm_NN, projection_NN, solve_V, and separate_V. These are also some of the most important keywords of the document. The graph on the right depicts the minimum spanning tree based on the edge values \( w_{tpi, tpj} \). The tree contains only \( |V| - 1 \) which are considered as the most important edges (i.e., the ones with the minimum weight). It is simpler than the initial graph, since it retained only the most important vertices (the most highly semantically inter-connections) within close distance and a sentence-to-sentence semantic relatedness version. We are using SYSR.
(e.g., variable_NN directly connects to least_squares_NN and nonlinear_NN, and algorithm_NN is directly connected with nonlinear_NN). This example shows that the SSF of the original document graph decreases by orders of magnitude the size of the graph, and keeps the most crucial semantic interconnections between the document terms, keeping within close distance the terms that best relate semantically.

### 3.2 Indexing of Semantic Spanning Forests

Having the representation of documents in the form of semantic spanning forests (SSF), we now proceed in transforming them to a metric space where we can quickly compute similarity between documents. For their similarity, we are based on the spectra of the normalized Laplacian of the two bipartite graphs, following the basis of spectral graph theory [3, 8]. The similarity between two semantic forests is eventually based on the computation of the Hausdorff distance [1] between the two SSFs, which considers the spectral properties of the two graphs, and more specifically the sectional curvatures of their edges. We give details on the Hausdorff distance in the following section.

The reason behind the selection of the Hausdorff distance for measuring the similarity between SSFs is the very good performance of this technique in the application of graph clustering in the fields of computer vision [9, 30] (i.e., images as graphs) and computational bioinformatics and biochemistry [33]. Thus, the following procedure, though not new in these other fields of research with regards to the spectral similarity of graphs, it constitutes a novel embedding in our case, since it is for the first time to the best of our knowledge, that it is applied in graphs representing documents, as a means of SSFs. In the following we explain the details of the first application of this technique in text processing and more specifically we show how SSFs are transformed to facilitate spectral similarity computation.

Initially, let \( G(V,E) \) be a graph, which in our case represents a document as a means of a SSF, where \( V \) is the set of its vertices, and \( E \) the set of its edges. For reasons of simplicity, let us also assume that \( G \) is connected, forming a spanning tree. Primarily, for every such graph in our document collection, we compute the degree \( d_v \) of each vertex \( v \in V \) as:

\[
d_v = \sum_u w(v,u)
\]

(1)

where vertex \( u \in V \) is any adjacent node to vertex \( v \) and \( w(v,u) = w(u,v) \) is the weight of the edge connecting them.

Then, the Laplacian \( L \) of \( G \) can be computed as follows:

\[
L(u,v) = \begin{cases} 
    d_v - w(v,u), & \text{if } u = v \\
    -w(v,u), & \text{if } u \text{ and } v \text{ are adjacent} \\
    0, & \text{otherwise}
\end{cases}
\]

(2)

We also construct a diagonal matrix \( D \), with \( D(v,v) = d_v \), in order to compute the normalized Laplacian \( \tilde{L} \) of \( G \). The \( \tilde{L} \) matrix is needed, as its eigenvalues constitute the spectrum of the initial graph. \( \tilde{L} \) is computed as follows:

\[
\tilde{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}
\]

(3)

and the spectral decomposition of the normalized Laplacian \( \tilde{L} \) as:

\[
\tilde{L} = \Phi \Lambda \Phi^T
\]

(4)

where \( \Lambda \) is the diagonal matrix with the ordered eigenvalues as its elements and \( \Phi \) contains the eigenvectors as columns.

Since we want to measure the Hausdorff distance between two SSFs, we can do so by embedding the nodes of each SSF into a vector space. It has been shown in the past that there is a strong connection between the heat kernel of a graph and the manifold in which its node reside [31]. Thus, we initially compute the heat kernel \( h_t \) [8], which encapsulates the way information flows through the graph edges over time. Essentially the heat kernel can be computed by exponentiating the \( \tilde{L} \) matrix using a parameter \( t \) that stands for time. Higher values of \( t \) give more focus to the full graph (i.e., trust more the edges of the entirety of the full graph), in contrast to lower \( t \) values that focus on the locality of the graph. The heat kernel in our case can be computed as follows:

\[
h_t = \exp[-\tilde{L}t] = \Phi \exp[-t\Lambda] \Phi^T
\]

(5)

Now we can also obtain the matrix that contains the coordinates for each node in this new vector space. This can be done by applying the Young-Householder decomposition [32] of the heat kernel \( h_t \) = \( Y^T \Phi \), where on \( Y \) the columns will represent the nodes as vectors in the vector space. As a result, the matrix of the resulting coordinates is expressed as:

\[
Y = \exp[-\frac{1}{2} t\Lambda] \Phi^T
\]

(6)

In this new vector space, the Euclidean distance between nodes \( u, v \) of \( G \) can then be computed as [9]:

\[
d_E(u,v) = \sum_{i=1}^{\mid V \mid} \exp[-\lambda_i t](\phi_i(u) - \phi_i(v))^2
\]

(7)

where \( \lambda_i \) is the \( i \)-th eigenvalue in \( \Lambda \) (the non-zero value in the \( i \)-th row of the \( \Lambda \) matrix), and \( \phi_i(u) \) is the value in position \( i, u \) of the eigenvector matrix \( \Phi \). Since for the computation of the Hausdorff distance we need the sectional curvature of the edges of \( G \), we require the geodesic distance of the nodes \( u, v \), in addition to their Euclidean distance. This can be computed as follows [9]:

\[
d_G(u,v) = \text{floor}_n(\sum_{i=1}^{\mid V \mid} (1 - \lambda_i)^n \phi_i(u)\phi_i(v))
\]

(8)

where \( n \) constitutes the length of the walk on the SSF with the smallest number of connecting edges. Eventually \( n \) is the smallest value for which the sum in Equation 8 becomes positive.

Eventually, the sectional curvature of the edge \( u, v \) can be computed as follows (the proof can be found in [30]):

\[
k(u,v) = \frac{2 \sqrt{\tilde{d}_E(u,v) - d_E(u,v)^2}}{d_G(u,v)^2}
\]

(9)

The sectional curvatures of the SSF are the only information that we index for our tree structures. Essentially, the sectional curvatures capture the topological structure of SSF and allows us to construct a low-dimensional feature space in which these values reside. Ultimately, we only need to index those values instead of the full SSF structure.

Algorithm 2 describes the SemaFor document indexing algorithm. It assumes that the SSF of a given document \( D \) has already been computed (using Alg 1). Eventually, a list of sets of ordered real values are indexed for the SSF. Each set represents each SST of the SSF, and the values are the respective sectional curvatures of the SST edges.

### 3.3 The Hausdorff Distance

Given two graphs \( G_1(V_1, E_1, K_1) \) and \( G_2(V_2, E_2, K_2) \), where \( V_1, V_2 \) are the respective sets of their vertices, \( E_1, E_2 \) are the respective sets of their edges, and \( K_1, K_2 \) are the respective matrices with the sectional curvatures of their edges (e.g., \( K_1(u,v) \) is the
Algorithm 2 \textit{SemaFor}(SSF, \(t\))

1: \textbf{INPUT}: A semantic spanning forest \(SSF\), the parameter \(t\) of the heat kernel.
2: \textbf{OUTPUT}: The indexing of \(SSF\) as a set of ordered lists of real values in a low-dimensional space.
3: \(L, \Lambda, \Phi, K\): Initially empty matrices
4: \(K_{ss}:\) An initially empty set of ordered real values
5: \(L\): An initially empty list of \(K_{ss}\)
6: \(SST\): An initially empty set of trees
7: \textbf{for all} \(s \in SST\) \textbf{do}
8: \(L := \text{NormalizedAplician}(s)\)
9: \(\Lambda, \Phi := \text{EigenValueDecomposition}(L)\)
10: \textbf{for all} \((u,v)\) pairs \(s \in \Phi\) \textbf{do}
11: \(d_{G}^{2}(u,v) := \sum_{i=1}^{n} (\lambda_{i})^{-1} ||\phi(u) - \phi(v)||^{2}\)
12: \(d_{G}(u,v) := \text{floor}\{\sum_{i=1}^{n} (1 - \lambda_{i})^{-1} \phi(u) \phi(v)\}\)
13: \(K[u,v] := \frac{2 \sqrt{\pi(d_{G}(u,v) - d_{G}(u,v))}}{d_{G}(u,v)^{2}}\)
14: \textbf{end for}
15: \(K_{st} := \text{OrderValuesOf}(K)\)
16: \(L := \text{AddToList}(K_{st})\)
17: \textbf{end for}
18: Store \(SST\) as \(L\)

The sectioned curvature of edge \((u, v)\) in \(G_{1}\) we are using the Hausdorff distance \([9]\) to compute the distance between \(G_{1}\) and \(G_{2}\) as follows:

\[
\text{Hausdorff}(G_{1}, G_{2}) = \max_{i,j \in V_{1}} \min_{i,j \in V_{2}} ||k_{2}(i, j) - k_{1}(i, j)||
\]

The Hausdorff distance in our case is a maximin function between the sectioned curvature matrices. Since we have assumed that the \(SSF\)s are ordered, we are examining \(10\) to capture all the cases, i.e., cases that \(SSF\) may contain several semantic spanning trees (\(SST\)). The generalization takes place in a similar manner that the average-link works during the agglomeration step in the hierarchical agglomerative clustering (HAC). The reason is simple: given two sets (i.e., the \(SSF\)s) of elements (their \(SST\)s), we estimate the distance between sets based on the Hausdorff distance between elements. This is exactly the problem faced by the HAC algorithm.

The possible solutions are: (a) single-link, with the caveat of the effect of chaining, (b) complete-link, with the caveat that can be sensitive to outliers (i.e., small \(SST\) in our case), describing small document topics that are quite distant from the larger \(SST\), meaning the larger document topics), and (c) average-link, which is a compromise between the sensitivity of complete-link to outliers and the lack of compactness of single-link. It solution (c) is chosen, the generalization of the Hausdorff distance between \(G_{1}\) and \(G_{2}\) becomes:

\[
\text{Hausdorff}(G_{1}, G_{2}) = \sum_{i=1}^{\text{|SST}_{G1}|} \sum_{j=1}^{\text{|SST}_{G2}|} \frac{\text{Hausdorff}(\text{SST}_{i}, \text{SST}_{j})}{\text{|SST}_{G1}| \cdot \text{|SST}_{G2}|}
\]

where \(|SST_{G1}|, |SST_{G2}|\) is the number of \(SST\) in \(G_{1}\) and \(G_{2}\) respectively, and \(SST_{i}, SST_{j}\) are the \(i\)-th and \(j\)-th \(SST\) of \(G_{1}\) and \(G_{2}\) respectively.

In the remaining of the paper, we will be using Equation 11 whenever the distance computation between documents is required, e.g., document clustering using HAC, and its inverse, whenever similarity is required (e.g., similarity between a query and a document). Note that for two documents \(D_{1}\) and \(D_{2}\) that are identical, their \(SSF\) are identical. In this case we do not use Equation 11, because it uses the average-link, and we assume Hausdorff \((G_{1}, G_{2}) = 0\) and the respective similarity being a very large positive constant.

Algorithm 3 describes the computation of Hausdorff distance for a pair of documents or a document and a query. Since the \(SSF\)s have been indexed as ordered lists of real values, the computation of the Hausdorff distance between two \(SSF\) is now reduced to a maximim problem between the two lists, as shown by Alg. 3.

3.4 Efficient Document Retrieval

In a large scale retrieval task, user queries are matched against large document collections. Even if the computation of similarity between a query and a document is done in millisecond, it is infeasible to check against all the documents in the collection. An inverted term-to-document index will definitely reduce the amount of candidate documents. However, existing inverted-index solutions are keyword-based and thus will miss all documents that semantically relate to the query but use different terminology. As a consequence it is essential to have an equivalent of the inverted-index, that uses semantic information. The structure of the information that \textit{SemaFor} indexes, facilitates the creation of such an index that will quickly distinguish between documents with high and low semantic similarity. In the following, we present the rationale behind our inverted-spectral-index and explain how it can accelerate retrieval, without affecting \textit{SemaFor} performance.

Since our index contains the sectioned curvatures of the edges of an \(SSF\), stored as an ordered list of positive real values, we can think of each document \(D\) as a \(1 - d\) segment \(S_{D}\) running from \(k_{D_{\text{min}}}\) to \(k_{D_{\text{max}}}\). The same holds for each query \(Q\), which is represented as a segment \(S_{Q}\) running from \(k_{Q_{\text{min}}}\) to \(k_{Q_{\text{max}}}\). According to [27] the two segments (see Figure 2) can: (a) partially overlap, (b) fully overlap, or, (c) have no overlap. Based on Equa-
tion 10, it is straightforward to show that the Hausdorff distance will be smaller in the two first cases where there is an overlap between documents. For example, the Hausdorff distances for the three cases depicted in figure 2 are:

\[ \text{Hausdorff}(D, Q) = |k_D - k_Q| \]

and it is obvious that Hausdorff distance is larger in the third case. As a consequence, the document retrieval mechanism should first retrieve documents, whose \( |k_{min} - k_{max}| \) segment overlaps with the query, compute the Hausdorff distances between the retrieved ordered sets of values and rank the documents accordingly. An R-tree that indexes the \( k_{min} \) and \( k_{max} \) values for each document will significantly improve the retrieval time and will select the best candidates for the matching and ranking process.

4. EXPERIMENTAL EVALUATION

We experimentally evaluate SemaFor in the text clustering and retrieval tasks. Traditionally, semantic-based indexing approaches, or approaches that utilize a WSD method to handle query and/or document ambiguity, are being evaluated in data sets like the Reuters, or the 20 Newsgroups for clustering and classification experiments [5, 15] and TREC collections for information retrieval [13, 22, 19]. We follow the same experimental methodology to evaluate SemaFor.

4.1 Indexing Size and Time

An important aspect of SemaFor’s scalability is that it incrementally builds the index without revisiting the whole document collection. In contrast to LSI-based techniques, TF-IDF, or probabilistic term scoring techniques, which require knowledge of the whole document collection, the indexing of a new document in SemaFor is done using only the document content. Additionally, the final size of SemaFor’s index is comparable to traditional TF-IDF and probabilistic-based indexing schemes, since it stores only the sectional curvatures of the SSF edges as an ordered set of real values, of size smaller or equal to the number of terms in the document. All the information produced in the intermediate phases of SemaFor is discarded after the computation of these values.

Concerning the time and space complexity of the computations of the sectional curvatures for a document, it does not restrict the scalability of SemaFor, since the constructed SSF is usually in the order of magnitude of \( 10^3 \) nodes (i.e., typical documents have at most few thousands of distinct terms, excluding stopwords). Thus, in each case the processed matrix is very small, compared to traditional term-to-document matrices used in LSI-based techniques.

The computational costs of the semantic relatedness measure is not trivial. However, with proper indexing of the knowledge bases and services we are using, we significantly alleviate the execution time. Indicative processing times for the TREC2 collection are depicted in Figure 3.

4.2 Text Documents Clustering

One imminent application of SemaFor and the Hausdorff distance between documents’ SSFs is text clustering. The application of SemaFor in clustering is straightforward, and can be easily embedded into the hierarchical agglomerative algorithm (HAC) (i.e., a distance between two documents is the Hausdorff distance of their SSF).

In order to evaluate the performance of SemaFor in text clustering, we use the Reuters-21578 data set, comprising approximately 21,500 files organized in 132 (possibly overlapping) categories. We are comparing the performance of our proposed indexing algorithm and its respective distance measure between documents against a standard baseline, namely vector space document representation with TF-IDF term weights, LSI, and the Concept Forest text document similarity approach [29]. In order to be compatible with the results presented in [29], we are using the same document subsets, produced as described in their respective work: (1) C1, comprising 50 documents in total from the Oil and Nat-Gas categories (25 documents from each category), (2) C2, comprising 100 documents in total from the Coffee and Sugar categories (50 documents from each category), and (3) C3, comprising 200 documents from the Grain, Wheat, Ship and Crude categories (50 documents in each category). The selected documents had a number of word occurrences (excluding stopwords and common words) ranging from 12 to 400.

For our evaluation, we compute precision, recall, and F-Measure (or F1 score) for each category in every case (C1, C2, and C3), as well as macro-averaged precision, recall, macro-F1 score, and overall accuracy. The accuracy results, that are directly comparable with the results reported in [29] are shown in Table 1. Table 2 contains the detailed results of SemaFor for each category, in each subset. We also report the macro-averaged precision (MP), recall (MR), and the macro-F1 score (MF1).

4.3 Text Retrieval

For the text retrieval evaluation of SemaFor we are using the TREC2 document collection, and more specifically the Wall Street Journal articles from 1990, so that we can directly compare with the semantic indexing approach proposed by Kang and Lee [13]. This document set comprises 21,705 articles, and authors used the 50 query topics 101 – 150 from the respective collection.

The whole process is depicted in Figure 3. The upper part of the figure constitutes the off-line procedure of a single document indexing with SemaFor, and reports on the absolute and relative execution times (i.e., \( \text{overall execution time} \)). The reported execution times are the average times measured per document. The lower part of the figure reports the online procedure, given a user query, reporting again on the execution times needed per processing phase for the TREC2 topics (average over the 50 topics 101 – 150). The time measurements were taken using a single machine with 2.2 GHz dual core AMD Opteron processor, and 3 GB of RAM. The Wikipedia database was stored in an external hard drives connected to the machine, running at 5400 rpm. As shown from the execution times, 84.1% of a document’s indexing time (the highest) is consumed by its SSF construction. Complexity of all the other components is trivial, compared to this part of the indexing. Given a TREC2 query, the 69.9% on average was consumed again by its SSF construction. The retrieval time (4.6% corresponding to 0.9 sec) was measured by having the top-50 documents ranked by the standard TF-IDF VSM weighting using cosine in the Terrier retrieval platform, and re-ranking them based on our index.

Figure 4 shows the average precision results of top N documents over all queries for SemaFor, the SW-IDF semantic indexing ap-
graphs. SemaFor algorithm that is based on the spectra of the documents’ semantic content of the indexed documents, by processing the textual content SemaFor precision of SW-IDF SemaFor Overall, the prototype implementation of the spectrum-based graph representation of the documents can im-

thermore, our evaluation in text clustering experiments shows that as also our experimental evaluation in text retrieval has shown. Fur-

distance based measure of tation for each document. Based on this representation and the fast graph theory in order to provide a reduced and compact represen-
dexing algorithm employs algebraic transformations from spectral indexing and searching large document collections.

Table 1: Overall clustering accuracies on the Reuters subsets.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Cat.</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oil</td>
<td>0.64</td>
<td>0.25</td>
<td>0.74</td>
<td>0.71</td>
</tr>
<tr>
<td>Nat-Gas</td>
<td>0.64</td>
<td>0.62</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td>Coffee</td>
<td>0.95</td>
<td>0.95</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>Sugar</td>
<td>0.96</td>
<td>0.93</td>
<td>0.91</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Table 2: Detailed clustering results on the Reuters subsets.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Cat.</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>MP</th>
<th>MR</th>
<th>MF1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Oil</td>
<td>0.92</td>
<td>0.958</td>
<td>0.938</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Nat-Gas</td>
<td>0.96</td>
<td>0.923</td>
<td>0.941</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>C2</td>
<td>Coffee</td>
<td>0.693</td>
<td>1.0</td>
<td>0.819</td>
<td>0.847</td>
<td>0.875</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>Sugar</td>
<td>1.0</td>
<td>0.75</td>
<td>0.857</td>
<td>0.875</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>C3</td>
<td>Grain</td>
<td>0.51</td>
<td>0.91</td>
<td>0.66</td>
<td>0.69</td>
<td>0.84</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>Wheat</td>
<td>0.51</td>
<td>0.86</td>
<td>0.64</td>
<td>0.69</td>
<td>0.84</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>Ship</td>
<td>1.0</td>
<td>0.6</td>
<td>0.75</td>
<td>0.69</td>
<td>0.84</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>Crude</td>
<td>0.77</td>
<td>1.0</td>
<td>0.86</td>
<td>0.69</td>
<td>0.84</td>
<td>0.76</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS AND FUTURE WORK

In this work, we presented SemaFor, a novel document indexing algorithm that is based on the spectra of the documents’ semantic graphs. SemaFor captures the semantic information carried by the content of the indexed documents, by processing the textual content using popular knowledge sources (WordNet and Wikipedia) in order to extract the semantic relations between documents’ terms. The indexing algorithm employs algebraic transformations from spectral graph theory in order to provide a reduced and compact representation for each document. Based on this representation and the fast distance based measure of Hausdorff, we are able to quickly answer user queries by delivering precise results without loosing in recall, as also our experimental evaluation in text retrieval has shown. Furthermore, our evaluation in text clustering experiments shows that the spectrum-based graph representation of the documents can improve significantly the performance of the text clustering process. Overall, the prototype implementation of SemaFor demonstrated promising results, which outperform popular statistical models as well as state of the art semantic models in text retrieval and clustering tasks. Concerning the execution time of the pairwise document similarities computation, we showed that the response time is comparable to the popular cosine similarity measure. Our next steps involve the optimization of the current implementation, the decrease of the indexing time, and the increase of its scale on larger document collections, so as to provide a very large-scale document indexing mechanism.

6. REFERENCES


