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, (2) store the semantic information in an efficient manner that can accommodate fast processing of documents, and, (3) use publicly available resources for the task, and an efficient methodology that does not require any type of training, so that it can scale up for large document collections (e.g., Web documents), and be applied across different domains (domain agnostic).

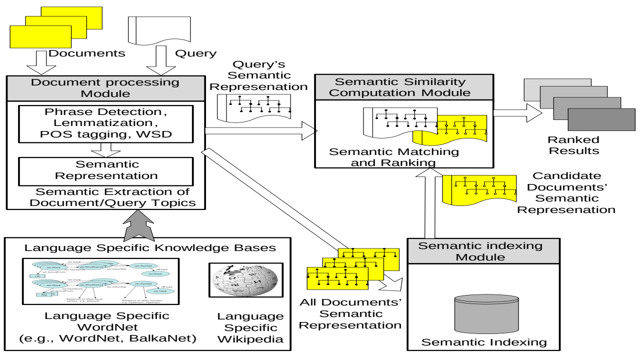
Documents
Query
Query's Semantic Represenation
Document processing Module
Phrase Detection, Lemmatization, POS tagging, WSD
Semantic Representation
Semantic Extraction of Document/Query Topics
Semantic Similarity Computation Module
Semantic Matching and Ranking
Ranked Results
Candidate Documents' Semantic Represenation
Language Specific Knowledge Bases
Language Specific WordNet (e.g., WordNet, BalkaNet)
Language Specific Wikipedia
All Documents' Semantic Representation
Semantic indexing Module
Semantic Indexing

Figure 1: The high-level representation of the *SemaFor* process flow.

The overall idea of *SemaFor* lies in the formulation of semantic spanning trees (SSTs) and semantic spanning forests (SSFs). Each document is first parsed and transformed into a set of SSTs, each one corresponding to a document topic. The SSF that contains the respective SSTs is the document's semantic representation. The forests are consequently indexed in the database, following an efficient methodology that allows fast retrieval of potential matches at query time. A similarity measure for semantic spanning forests, which is based on spectral graph algebra, is introduced in order to provide the ability of the index to produce similarity scores between documents (SSFs),i.e., for the purposes of documents clustering, or create a ranking of the potential matching *SSF*s to a given user query for the purposes of document retrieval. The constructed index allows for fast search and ranking of the semantic forests (documents), given a user query.

A high level representation of the process flow in *SemaFor*, is shown in Figure 1. Given an initial set of input text documents, the *semantic extraction of document topics* module pre-processes the texts and creates a semantic spanning forest for each document, as explained in Section 3. In the *semantic indexing* module documents are indexed, in the form of semantic spanning forests, in a format that allows the fast computation of similarity between them. Any query or input document is first transformed into a semantic forest 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 uses *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

Algorithm 1 *SSF*(D)

```
1: INPUT: A text document $D$.
2: OUTPUT: The Semantic Spanning Forest (SSF) of $D$, SSF$(D)$
3: $T$: A set of term-POS pairs
4: $G$, SSF: Initially empty graphs
5: $V$: The set of vertices of $G$, $E$: The set of edges of $G$
6: $T$ := PreProcessDoc($D$)
7: for all $tp \in T$ and $tp \in$WordNet do
8:     Disambiguate($tp$)
9: end for
10: for all $i = 1$ to $|T| - 1$ do
11:     for all $j = i + 1$ to $|T|$ do
12:         if $tp_i, tp_j \in$ WordNet then
13:             $S(tp_i, tp_j) = SR(tp_i, tp_j)$
14:         else
15:             $S(tp_i, tp_j)$ =WLM$(tp_i, tp_j)$
16:         end if
17:         if $S(tp_i, tp_j) > 0$ then
18:             Add $tp_i, tp_j$ in $V$ and $\frac{1}{S(tp_i,tp_j)}$ in $E$
19:         end if
20:     end for
21: end for
22: for all $c \in$ connected components of $G$ do
23:     SSF$\cup$ MinimumSpanningTree($c$)
24: end for
25: RETURN SSF
```

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, which in our case consists of *WordNet* and *Wikipedia*, examining only the noun phrases of each sentence. This first step of document pre-processing (Step 6, Alg. 1, computes for D all the terms of each sentence, and their *POS*. Finally, we remove all the stopwords[2].

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 $T = tp_1, tp_2, ..., tp_n$, 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_i, tp_j) with $i \neq j$ and both $i, j \in [1..n]$ the t part of the term-pair tp_i might be identical with the t part of the tp_j 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 *Omiotis* [25] as the semantic relatedness measure which employs *WordNet*[3], 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 $S(tp_1, tp_2) = S(tp_2, tp_1)$ we need exactly $\frac{n\cdot(n-1)}{2}$ computations of semantic relatedness. For each pair (tp_i, tp_j) if both $tp_i, tp_j \in$ *WordNet*, we are using *Omiotis*, 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 $S(tp_i, tp_j) = SR(tp_i, tp_j)$, if $tp_i, tp_j \in$ WordNet, else $S(tp_i, tp_j) = WLM(tp_i, tp_j)$.

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_{tp_i,tp_j} between every pair of nodes (tp_i, tp_j) for which $S(tp_i, tp_j) > 0$, with weight $w_{tp_i,tp_j} = \frac{1}{S(tp_i,tp_j)}$, 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 D (*SST*$_i(D)$) 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 $\frac{1}{SR}$. The tree contains only $|V| - 1$ which are considered as the most important edges (i.e., the ones with the minimum $\frac{1}{SR}$ 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

[2]There is a large number of stopword and common word lists publicly available, e.g., the lists used in the *Ellogon* language engineering platform (http://www.ellogon.org/) or *GATE* (http://gate.ac.uk/).

[3]*Omiotis* offers a term-to-term semantic relatedness version (*SR* and a sentence-to-sentence semantic relatedness version. We are using *SR*).

(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 *SSF*s, 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 *SSF*s 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 *SSF*s. In the following we explain the details of the first application of this technique in text processing and more specifically we show how *SSF*s 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:

$$d_v = \sum_u w(v,u) \tag{1}$$

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:

$$L(u,v) = \begin{cases} d_v - w(v,v), & \text{if } u = v \\ -w(u,v), & \text{if u and v are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

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

$$\hat{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{3}$$

and the spectral decomposition of the normalized Laplacian $\hat{L}$ as:

$$\hat{L} = \Phi \Lambda \Phi^T \tag{4}$$

where Λ is the diagonal matrix with the ordered eigenvalues as its elements and Φ contains the eigenvectors as columns.

Since we want to measure the *Hausdorff* distance between two *SSF*s, 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 $\hat{L}$ 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:

$$h_t = exp[-\hat{L}t] = \Phi exp[-t\Lambda]\Phi^T \tag{5}$$

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 $h_t = Y^T Y$, 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:

$$Y = exp[-\frac{1}{2}t\Lambda]\Phi^T \tag{6}$$

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

$$d_E^2(u,v) = \sum_{i=1}^{|V|} exp[-\lambda_i t](\phi_i(u) - \phi_i(v))^2 \tag{7}$$

where λ_i is the i-th eigenvalue in Λ (the non-zero value in the i-th row of the Λ matrix), and $\phi_i(u)$ is the value in position (i,u) of the eigenvector matrix Φ. 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]:

$$d_G(u,v) = floor_n\{\sum_{i=1}^{|V|}(1-\lambda_i)^n \phi_i(u)\phi_i(v)\} \tag{8}$$

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]):

$$k(u,v) = \frac{2\sqrt{6}(d_G(u,v) - d_E(u,v)^{\frac{1}{2}}}{d_G(u,v)^{\frac{3}{2}}} \tag{9}$$

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 *SemaFor*(*SSF*, t)

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

sectional 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:

$$Hausdorff(G_1, G_2) = \max_{i,j \in V_1} \min_{I,J \in V_2} ||k_2(I, J) - k_1(i, j)|| \quad (10)$$

The *Hausdorff* distance in our case is a *maximin* function between the sectional curvature matrices. Since we have assumed that the *SSF*s we are examining are connected, we will generalize Equation 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. If solution (c) is chosen, the generalization of the *Hausdorff* distance between G_1 and G_2 becomes:

$$Hausdorff^*(G_1, G_2) = \frac{\sum_{i=1}^{|SST_{G_1}|} \sum_{j=1}^{|SST_{G_2}|} Haudorff(SST_i, SST_j)}{|SST_{G_1}| \cdot |SST_{G_2}|} \quad (11)$$

where $|SST_{G_1}|, |SST_{G_2}|$ 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 *Hausdorff*(L_1, L_2)

```
1: INPUT: $L_1, L_2$, two indexes of respective SSF.
2: OUTPUT: The Hausdorff distance between $L_1, L_2$.
3: $s$: A real value, initially set to 0
4: if $L_1 = L_2$ then
5:     RETURN 0
6: end if
7: for all $l_1 \in L_1$ do
8:     for all $l_2 \in L_2$ do
9:         $s := s + \max_{i \in l_1} \min_{j \in l_2} ||l_1(i) - l_2(j)||$
10:    end for
11: end for
12: $s := \frac{s}{|L_1| * |L_2|}$
13: RETURN $s$
```

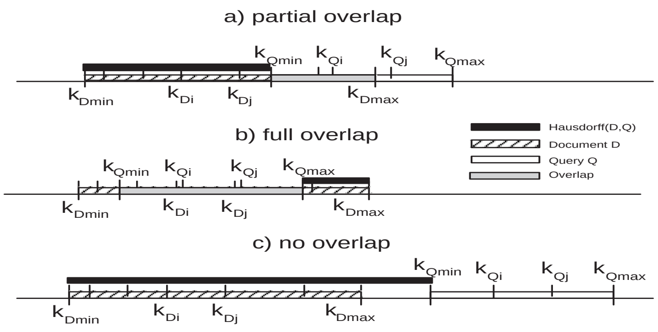

Figure 2: Query-Document overlap options.

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 maximin 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 milliseconds, 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 *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 *SemaFor* performance.

Since our index contains the sectional 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 $k_{D_{min}}$ to $k_{D_{max}}$. The same holds for each query Q, which is represented as a segment S_Q running from $k_{Q_{min}}$ to $k_{Q_{max}}$. According to [27] the two segments (see Figure 2) can: (a) partially overlap, (b) fully overlap, or, (c) have no overlap. Based on Equa-

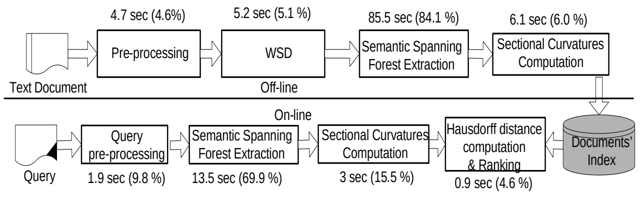

Figure 3: Indexing and retrieval execution times of *SemaFor* in TREC2, Wall Street Journal Articles of 1990.

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:

Hausdorff$(D,Q)_a$=$|k_{D_{min}} - k_{Q_{min}}|$,
Hausdorff$(D,Q)_b$=$|k_{D_{max}} - k_{Q_{max}}|$ and
Hausdorff$(D,Q)_c$=$|k_{D_{min}} - k_{Q_{min}}|$

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 $\{kmin, kmax\}$ 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 $k_{D_{min}}$ and $k_{D_{max}}$ values for each document D 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' *SSF*s 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 F_1 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., $\frac{\textit{absolute execution time}}{\textit{overall execution time}}$). 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 $0.9sec$) 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-

Reuters Subset	VSM	LSI	CF	SemaFor
C1	0.64	0.64	0.74	0.94
C2	0.5	0.62	0.8	0.84
C3	0.25	0.34	0.48	0.71

Table 1: Overall clustering accuracies on the Reuters subsets.

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

Table 2: Detailed clustering results on the Reuters subsets.

proach introduced in [13] and the standard baseline. The *SW-IDF* and *TF-IDF VSM* results are taken from [13].

The results in Figure 4 are very enlightening, showing that the precision of *SemaFor* is higher than that of *SW-IDF* ([13]) in the top-10 ($k = 10$) and top-20 ($k = 20$) documents, which is the typical amount of retrieval results that a user examines in a search. Precision is always higher than that of the baseline method. The top results of *SemaFor* are better than that of its competitors. Moreover, the response times of *SemaFor* are comparable to those of its popular competitor (TF-IDF), which makes it an ideal solution for indexing and searching large document collections.

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

[1] M. Barnsley. *Fractals Everywhere*. Morgan Kaufmann, 2000.

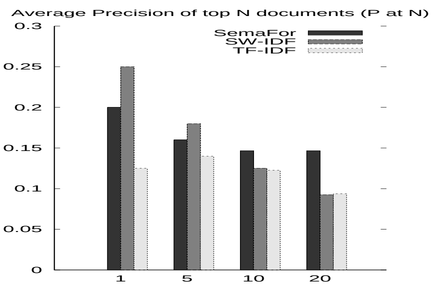

Figure 4: Average Precision of top N documents.

[2] R. Basili, M. Cammisa, and A. Moschitti. A semantic kernel to exploit linguistic knowledge. In *Proc. of the AI*IA*, 2005.

[3] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 1993.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] S. Bloehdorn and A. Moschitti. Exploiting structure and semantics for expressive text kernels. In *Proc. of the CIKM 2007*, pages 861–864, 2007.

[6] W. Buntine, J. Löfström, J. Perkiö, S. Perttu, V. Poroshin, T. Silander, H. Tirri, A. Tuominen, and V. Tuulos. A scalable topic-based open source search engine. In *Proc. of the IEEE/WIC/ACM WI-04*, pages 228–234, 2004.

[7] J. Chang, J. Lee, Y. Kim, and B. Zhang. Topic extraction from text documents using multiple-cause networks. In *Proc. of PRICAI*, pages 434–443, 2002.

[8] F. Chung. *Spectral Graph Theory*. CBMS 92, American Mathematical Society, 1997.

[9] H. ElGhawalby and R. Hancock. Measuring graph similarity using spectral geometry. In *Proc. of the 5th International Conference on Image Analysis and Recognition*, 2008.

[10] E. Frank, G. Paynter, I. Witten, C. Gutwin, and C. Nevill-Manning. Domain-specific keyphrase extraction. In *Proc. of IJCAI*, pages 668–673, 1999.

[11] M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multi-theme documents. In *Proc. of WWW*, pages 661–670, 2009.

[12] G. Hendeson, P. Schone, and T. Crystal. Text retrieval via semantic forests. In *TREC7*, 1998.

[13] B. Kang and S. Lee. Document indexing: A concept-based approach to term weight estimation. *Information Processing and Management*, 41:1065–1080, 2005.

[14] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[15] D. Mavroeidis, G. Tsatsaronis, M. Vazirgiannis, M. Theobald, and G. Weikum. Word sense disambiguation for exploiting hierarchical thesauri in text classification. In *Proc. of the 9th PKDD*, pages 181–192, 2005.

[16] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *Proc. of IJCAI*, pages 662–667, 1999.

[17] O. Milne and I. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proc. of the first AAAI Workshop on Wikipedia and AI*, 2008.

[18] R. Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2), Article 10, 2009.

[19] M. Sanderson. Ambiguous queries: Test collections need more sense. In *Proc. of the 31st SIGIR*, pages 499–506, 2008.

[20] P. Schone and D. Nelson. A dictionary-based method for determining topics in text and transcribed speech. In *Proc. of the Acoustics, Speech, and Signal Processing*, pages 295–298, 1996.

[21] P. Schone, J. Towsend, T. Crystal, and C. Olano. Text retrieval via semantic forests. In *Proc. of the Sixth Text Retrieval Conference (TREC6)*, pages 761–773, 1997.

[22] C. Stokoe, M. Oakes, and J. Tait. Word sense disambiguation in information retrieval revisited. In *Proc. of the 26th SIGIR*, pages 159–166. ACM, 2003.

[23] K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*. ACM, 2003.

[24] G. Tsatsaronis and V. Panagiotopoulou. A generalized vector space model for text retrieval based on semantic relatedness. In *Proc. of the EACL 2009 (Student Research Workshop)*, 2009.

[25] G. Tsatsaronis, I. Varlamis, and M. Vazirgiannis. Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, 37:1–39, 2010.

[26] G. Tsatsaronis, M. Vazirgiannis, and I. Androutsopoulos. Word sense disambiguation with spreading activation networks generated from thesauri. In *Proc. of the 20th IJCAI*, pages 1725–1730, 2007.

[27] M. Vazirgiannis, Y. Theodoridis, and T. Sellis. Spatio-temporal composition and indexing for large multimedia applications. *Multimedia Syst.*, 6(4), 1998.

[28] E. Voorhees. Using wordnet to disambiguate word sense for text retrieval. In *Proc. of the 16th SIGIR*. ACM, 1993.

[29] J. Wang and W. Taylor. Concept forest: A new ontology-assisted text document similarity measurement method. In *Proc. of the IEEE/WIC/ACM WI-07*, 2007.

[30] B. Xiao and E. Hancock. Geometric characterisation of graphs. In *Proc. of the International Conference on Image Analysis and Processing (ICIAP)*, pages 471–478, 2005.

[31] S. Yau and R. Scoen. *Differential Geometry*. Science Publication, 1988.

[32] G. Young and A. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrica*, 3, 1938.

[33] J. Zeng, C. Tripathy, P. Zhou, and B. Donald. A Hausdorff-based NOE assignment algorithm using protein bacbone determined from residual dipolar couplings and rotamer patterns. In *Proc. of the IEEE Computational Systems Bioinformatics Conference*, pages 169–181, 2008.

[34] Z. Zhang, A. Gentile, and F. Ciravegna. Recent advances in methods of lexical semantic relatedness - a survey. *Natural Language Engineering*, 2012.