Semantic smoothing for text clustering

Jamal A. Nasir\textsuperscript{a}, Iraklis Varlamis\textsuperscript{b,\ast}, Asim Karim\textsuperscript{a}, George Tsatsaronis\textsuperscript{c}

\textsuperscript{a} KADE Lab, Lahore University of Management Sciences, Pakistan
\textsuperscript{b} Department of Informatics and Telematics, Harokopio University of Athens, Greece
\textsuperscript{c} BIOTEC, Technical University of Dresden, Germany

A R T I C L E   I N F O

Article history:
Received 26 February 2013
Received in revised form 5 September 2013
Accepted 7 September 2013
Available online xxxx

Keywords:
Text clustering
Semantic smoothing kernels
WordNet
Wikipedia
Generalized vector space model kernel

A B S T R A C T

In this paper we present a new semantic smoothing vector space kernel (S-VSM) for text documents clustering. In the suggested approach semantic relatedness between words is used to smooth the similarity and the representation of text documents. The basic hypothesis examined is that considering semantic relatedness between two text documents may improve the performance of the text document clustering task. For our experimental evaluation we analyze the performance of several semantic relatedness measures when embedded in the proposed (S-VSM) and present results with respect to different experimental conditions, such as: (i) the datasets used, (ii) the underlying knowledge sources of the utilized measures, and (iii) the clustering algorithms employed. To the best of our knowledge, the current study is the first to systematically compare, analyze and evaluate the impact of semantic smoothing in text clustering based on ‘wisdom of linguists’, e.g., WordNets, ‘wisdom of crowds’, e.g., Wikipedia, and ‘wisdom of corpora’, e.g., large text corpora represented with the traditional Bag of Words (BoW) model. Three semantic relatedness measures for text are considered; two knowledge-based (Omioiris\textsuperscript{[1]} that uses WordNet, and WLM\textsuperscript{[2]} that uses Wikipedia), and one corpus-based (PMI\textsuperscript{[3]} trained on a semantically tagged SemCor version). For the comparison of different experimental conditions we use the BCubed F-Measure evaluation metric which satisfies all formal constraints of good quality cluster. The experimental results show that the clustering performance based on the S-VSM is better compared to the traditional VSM model and compares favorably against the standard GVSM kernel which uses word co-occurrences to compute the latent similarities between document terms.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Document clustering plays an important role in indexing, retrieval, browsing and mining of large and high dimensional text data. Document clustering algorithms aim at organizing documents into meaningful groups that contain highly similar documents, and which are distant from the documents of other groups\textsuperscript{[4]}. For this purpose, they rely on document similarity or distance measures, with which they typically compare pairs of text documents. Therefore, similarity measures play a crucial role in the task of document clustering. The performance of similarity measures in data mining tasks depends on the type of data, on the particular domain, on the dataset and on the nature of the examined task. In the case of document clustering, the textual data have usually large volume, they are high-dimensional, and carry also semantic information, i.e., meaning conveyed by the text terms. Therefore, the clustering algorithm and the similarity measures that are employed for the task should be able to address these parameters effectively.

In the task of document clustering documents are typically represented by their terms. Terms are either single words or composite (multi-word terms), which form as a whole the language vocabulary of the underlying text corpus. Terms of either category are usually associated with a positive real value acting as a weight for the respective term. Furthermore, the weight of each term corresponds to its importance/relevance to the document it appears in.

More formally, given a collection of documents $D$, the vocabulary $V$ of $D$ may be defined as the set of all distinct terms appearing in $D$. For each term $t_i$ of a document $d_j \in D$, the weight $w_{t_i} > 0$ of the $t_i$ in $d_j$ may be computed, usually, through a measure that takes into account the frequency of occurrences of $t_i$ in $d_j$. This representation is widely known as the Vector Space Model (VSM)\textsuperscript{[5]}.

VSM is very simple and commonly used; yet, it has several limitations. Its main restriction is that it assumes independence between the vocabulary terms and ignores all the conceptual relations between terms that potentially exist. As a consequence, two terms that are semantically close, e.g., synonyms, are treated differently. Furthermore, polysemous terms, i.e., terms with multiple meanings, are considered the same in all contexts they

\textsuperscript{*} Corresponding author. Tel.: +30 2109549029; fax: +30 2109549405.
E-mail addresses: varlamis@hua.gr, varlamis@gmail.com (I. Varlamis).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.09.012

Please cite this article in press as: J.A. Nasir et al., Semantic smoothing for text clustering, Knowl. Based Syst. (2013), http://dx.doi.org/10.1016/j.knosys.2013.09.012
appear. For example the term 'bank' may have the meaning of a financial institution when it appears in a context related to economy, or the meaning of a river side when it appears in a context that refers to landscapes or geographical locations. In principle a document contains usually more terms that are 'general terms', i.e., that may appear in all clusters, than 'clutter dependent terms', i.e., 'core terms' [2] that characterize the documents of a single cluster. VSM cannot consider that differentiation as it cannot examine similarities between terms that have different surface strings. For the VSM model, the similarities between documents and the similarities between a document and the cluster centroid are only based on the matched term strings. Hence, the need of smooth semantically the VSM model, i.e., by employing semantic smoothing VSM kernels, arises. This embedding may increase the importance of core words by considering the terms’ relations, and in parallel downsize the contribution of general terms, leading to better text clustering results.

In this article, we propose a novel semantic smoothing VSM kernel (S-VSM), which smooths the VSM representation with the semantic similarity between terms. The proposed S-VSM allows any semantic similarity or relatedness measure to be employed, both measures that use linguistic resources, e.g., knowledge bases, ontologies, and thesauri, but also measures that are based on statistical information extracted from the analysis of large text corpora. Hence, the first advantage of the suggested solution is that it offers a very flexible kernel that can be applied within any domain or with any language. To showcase the wide applicability of the suggested kernel, for the purposes of this work we examine the embedding of three novel semantic relatedness measures into the S-VSM; the first employs the WordNet-based similarity measure of Omimitis [1], the second is Wikipedia-based and employs the measure of Milne and Witte [2], and the third is based on statistical analysis of text corpora and uses a Pointwise Mutual Information similarity measure for the computation of terms’ similarity [3].

The second advantage of the suggested solution is the ability of the S-VSM to perform much better than the VSM in the task of text clustering. In addition, an extension of the S-VSM that we propose, namely the top-k S-VSM, which considers only the top-k semantically related terms, does not only perform better than the VSM, but it also conducts the task of text clustering very efficiently in terms of time and space complexity. The proposed S-VSM and its extension are evaluated on five datasets: (1) Reuters-Transcribed-set, (2) R8 Reuters-21578, (3) 4 Universities Data Set (WebKB), (4) Email-1431, and (5) Movie Reviews. To evaluate S-VSM and top-k S-VSM we use both agglomerative and partition clustering for conducting the experiments, and two baselines; the traditional Bag of Word (Bow) model which uses the VSM model for document representation, and the standard Generalized Vector Space Model kernel (GVSM), which considers the term-to-document matrix to compute latent similarities between terms based on their co-occurrence.

The clustering results show significant improvements in the clustering accuracy when S-VSM and top-k S-VSM are used, compared to the performance of the two aforementioned baselines. In addition, we provide a thorough analysis on the effect of the number of the top-k semantically related terms used for the smoothing, which, to the best of our knowledge, is conducted for the first time in the bibliography, and gives important insights on how the semantic smoothing can be optimized computationally.

This work capitalizes on our previous work on semantic kernels [7]. The main contributions of the current work, which differentiate it from our former work on S-VSM kernels and expand it, can be summarized in the following:

1. Extension of the $S-VSM$ to embed only the top-k semantically related terms.
2. Application to the task of text clustering.
3. An extended and thorough evaluation in text clustering, using a large variety of text datasets, employed clustering algorithms, and evaluation metrics.
4. Comparative evaluation against the standard GVSM kernel and the semantic kernel presented in [8], which shows that the suggested expanded $S-VSM$ performs favorably against these two approaches.

The rest of the paper is organized as follows: Section 2 discusses the related work in the field of semantic smoothing kernels, with emphasis to the task of text clustering. Section 3 provides preliminary related information. Section 4 introduces the semantic smoothing kernel (S-VSM) and its top-k extension. Section 5 presents the experimental setup, and Section 6 presents and analyzes the experimental results. Finally, we conclude the paper in Section 7 and provide a discussion on the possible expansions of the current work.

2. Related work

The idea of using background knowledge or gathered statistical information from large text corpora analysis in order to compute text similarity is well studied in the past [9,10], and there exist many research works that introduce efficient similarity or relatedness measures between terms. With regards to works that employ such measures for document clustering, WordNet is one of the most widely used lexical thesauri [11,12]. In principle, research works in document clustering, but also in text retrieval, that incorporate semantics in the VSM representation can be classified in three categories, depending on the type of information or the features used to index the document terms and expand the index with additional features: (i) embedding of concept features, (ii) embedding of multi-word phrases, and (iii) employing semantic kernels to embed semantically related terms or semantic relation information between terms to the documents’ representation; the semantic similarity and relations may be retrieved from a word thesauri or ontology, or may be computed based on statistical analysis of a large text corpus.

Works in the first category, e.g., [11], use conceptual features to improve the clustering performance. WordNet is typically used as a background knowledge to obtain concept features, which are defined as set of words that describe the same high level concept; for example penny, nickel, dime and quarter describe the concept coin. The weights of both concepts and terms are employed to represent documents, usually leading to a type of hybrid document representation in the vector space, that contains both concepts, i.e., meanings, but also concrete terms. Such representations were also applied in the past in text retrieval, with mixed performance outcome [14]. Another recent representative example of a work

---

1. Though there are slight conceptual differences between the terms ‘semantic similarity’ and ‘semantic relatedness’, for the purposes of this work this differentiation is not important. Therefore, the two terms might be used interchangeably for the remaining of the paper.

---

7. Usually such hybrid representations improve the recall in information retrieval systems, due to the expansion in the concepts’ dimension, but might drop precision due to the difficulty of transitioning from terms to concepts, with the task of word sense disambiguation having major role and innate limitations [13].
in this category is the method proposed by Egozi et al. [15]. The authors in that work proposed an augmentation of the keyword-based representation of documents with concept-based features that are extracted from Wikipedia. The overall methodology is based on Explicit Semantic Analysis (ESA), and their experimental evaluation shows an important improvement when the suggested approach is used in information retrieval, compared to the standard BOW document representation.

The idea behind the works of the second category is that phrases have more expressive power than single words, and, thus, they expand the vocabulary index of the terms with associated word phrases to enhance document representation. For example, Mao et al. [16] expand the VSM model with phrase-based representations of documents that they obtain from the Unified Medical Language System (UMLS), which is one of the largest ontologies for the biomedical domain. In another work of the same category, Sanjuan et al. [17] combine the VSM representation of documents with what they call symbolic approach, to enrich the representation with multi-word terms. The application in both works was in the domain of text retrieval, and more precisely, using text data from the medical domain, and the results in both cases show promise in expanding the VSM with phrase-based features. In another work, Zhan et al. [18] embedded multi-word phrases in the document representation, with their approach being domain agnostic. For the purposes of covering any domain, they used in their approach a context-sensitive smoothing method that statistically maps phrases to individual document terms.

Methods of both the aforementioned categories, however, share some limitations, such as: (i) increased dimensionality, as they actually both expand the VSM and the indexing size of the documents, (ii) increased computational cost, since the index size is increased, (iii) information loss, e.g., in cases when the ontology concepts replace the original terms, and noise when the ontology concepts are used as additional features due to poor word sense disambiguation performance, (iv) limited coverage when a comprehensive ontology is not available; though in the case of the biomedical domain the UMLS thesaurus is considered a very rich resource that describes in large coverage the life sciences, such a resource is not always available for other domains, and (v) limited usage of the available semantic relationships, as in the majority of the cases in these two categories, the structure of the underlying ontologies or resources is not taken into account, and they are rather used as dictionaries, and not so much as ontologies.

To remedy the shortcomings of these two categories of works, research efforts that may be classified in the third category attempt to exploit efficiently semantic information from ontologies and thesauri, or results from statistical analysis of large text corpora. Such works utilize as many features as possible from the underlying ontologies, e.g., structure and relations, when ontologies are used, or they compute semantically related terms by analyzing text corpora. The resulting information in both cases is embedded into the document similarity computation by employing semantic kernels [19], and their optimizations to cover also semi-supervised learning [20]. Semantic kernels constitute a very efficient means of incorporating background knowledge in the task of computing similarity between documents, and, thus, allowing applications in text clustering and text classification. The notion of kernel usage to embed background information allows straightforward application of the resulting function to popular learners such as Support Vector Machines and kernel Fischer discriminant analysis, which are based on kernels.

Concerns regarding efficiency of computation through the embedding of such large scale background knowledge, e.g., semantic similarity information between the term features, can be addressed by utilizing sparse representations of the prediction function [21], a priori computation of similarities between the term features [7], and retaining the core semantic features through the usage of synonyms [22]. In principle, the past few years several works have started appearing that apply efficiently semantic smoothing kernels, overcoming the scalability and complexity problems.

Examples of such recent works with application to text clustering, which exploit and embed efficiently background semantic knowledge for documents’ similarity are: the work of Zheng et al. [23], the approach by Jing et al. [8], and the method suggested by Hu et al. [24]. Zheng et al. [23] explore the effectiveness of hypernymy, hyponymy, holonymy and meronymy information for nouns and noun phrases, using WordNet to retrieve the related semantic information. Their experimental results show that the inclusion of such semantic information improves the clustering performance. Jing et al. [8] authors introduce a knowledge-based VSM, where the semantic information is used to re-weight term frequency in the original VSM. Semantic similarity is computed with two methods; one based on the structure of hierarchies such as WordNet and Medical Subject Headings, and another based on extracting the semantic relations from analyzing text corpora. Their experimental evaluation shows, like in the previous case, that the inclusion of the semantic similarity information between terms improves the clustering performance compared to the traditional VSM document representation. Finally, Hu et al. [24] enhanced the document representation with Wikipedia concept and category information. The idea is based on mapping text documents to Wikipedia concepts, and from there to Wikipedia categories. The associated Wikipedia concept and category information is then utilized in the similarity measure of the documents.

Motivated by the encouraging results reported in the literature on the merits of incorporating semantic information as a background knowledge to perform text clustering, in this work we propose a semantic smoothing kernel which is based on our previous work [7], and we analyze, for the first time to the best of our knowledge, the application of three different semantic similarity measures using the suggested kernel to the task of document clustering. In addition, to address space and time efficiency issues, we expand the suggested kernel to incorporate information only regarding the top-k semantically related terms to the original term features, and we report on the effect of the chosen value of k. Overall, the semantic kernel (S-VSM) and its top-k extension (top-k S-VSM) that we propose in this paper may be applied using any measure of semantic similarity or relatedness. The embedding of the background knowledge is made through the suggested kernel, via smoothing the weights of the feature terms and their similarity values, and, thus, the original dimensions of the VSM are not augmented. In order to experimentally evaluate the suggested kernel and its extension we use three similarity measures; one based on WordNet, one based on Wikipedia, and a corpus-based measure to compute semantic similarity between terms.

3. Preliminaries

In document clustering, the input is typically a collection of documents, and the aim is to split it in subgroups of documents (clusters) that share similar concepts or discuss the same or similar topics. Each document \( d \) in the collection is made up of words and is represented using the Vector Space Model (VSM or Bag Of Words-BOW representation) in a vector space where each distinct term constitutes a dimension of the collection. In the following we are summarizing the most important preliminary concepts in document representation, semantic relatedness or similarity and document clustering algorithms, that are used in the remaining of the paper. We will use the symbols n, m and c to denote the number of documents, the number of terms, and the number of
clusters, respectively, and \( D \) to denote the set of documents in the collection.

### 3.1. Document representations

#### 3.1.1. The Vector Space Model (VSM)

The VSM defines for each term \( t_i \), a vector \( \mathbf{v}_i \) in the vector space that represents it. It then considers the set of all term vectors \( \mathbf{v}_i \) to be the orthonormal basis of the vector space, thus the space basis. In the standard VSM model, each document, \( d \), is represented as follows:

\[
\phi : d \mapsto \phi(d) = [w_{1,d}, w_{2,d}, \ldots, w_{n,d}]^\top \in \mathbb{R}^D
\]

where \( w_{t,d} = \text{TF-IDF}(t, d) \) is the TF-IDF weight of term \( t_i \) in document \( d \) and the superscript \( \top \) denotes the transpose operator. In the above expression, the function \( \phi(d) \) represents the document \( d \) as a weighted term vector in the \( m \)-dimensional space of all terms in the collection. This function, however, can be any other mapping from a document to its vector space representation. To represent the whole corpus of the \( n \) documents in the collection, the term to document matrix, \( D \), is introduced. \( D \) is a \( m \times n \) matrix whose rows are indexed by the words and whose columns are indexed by the documents.

One of the main issues in any document clustering algorithm is to define the similarity measure between documents. A popular measure for text similarity, is the computation of the cosine of the angle between the two document vectors in the VSM representation, which may be computed fast, is scale invariant, and does not depend on the text length. However, cosine similarity is bounded by the limitations of VSM, so it assumes pairwise orthogonality between terms, i.e., the vector space dimensions, and overlooks the semantic relatedness or similarity between terms. The cosine similarity measure between two documents in the VSM representation is given by Eq. 1.

\[
sim_{\text{VSM}} = \cosine(d_p, d_q) = \frac{\sum_{i=1}^{m} w_{t_i,d_p} w_{t_i,d_q}}{\sqrt{\sum_{i=1}^{m} w_{t_i,d_p}^2} \sqrt{\sum_{i=1}^{m} w_{t_i,d_q}^2}}
\]

where \( d_p \) and \( d_q \) are two documents in the collection, and \( w_{t_i,d_p} \), \( w_{t_i,d_q} \) denotes the weight of term \( t_i \) in document \( d_p \), \( w_{t_i,d_q} \) respectively denotes the weight of \( t_i \) in \( d_q \).

#### 3.1.2. The Generalized Vector Space Model (GVSM)

The Generalized Vector Space Model (GVSM) [25], also known as the ‘dual space’ approach [26], extends VSM, by introducing term-to-term correlations, which deprecate the pairwise orthogonality assumption, but it keeps the assumption that the term vectors are linearly independent.\(^8\) GVSM considers a new space, where the original \( m \)-dimensional term vector is expressed as a linear combination of vectors in the \( 2^m \) dimensional space. The similarity between two documents \( d_p \) and \( d_q \) can be then defined as:

\[
sim_{\text{GVSM}}(d_p, d_q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} w_{t_i,d_p} w_{t_j,d_q} + \sum_{i=1}^{m} w_{t_i,d_p} * (\mathbf{T}_i, \mathbf{T}_j)}{\sqrt{\sum_{i=1}^{m} w_{t_i,d_p}^2} \sqrt{\sum_{i=1}^{m} w_{t_i,d_q}^2}}
\]

Term correlation \( \langle \mathbf{T}_i, \mathbf{T}_j \rangle \) can be implemented in several ways. It can be based on frequency co-occurrence statistics gathered from large corpora [25], or on the semantic correlations between terms [27]. Using large corpora to compute the term-to-term correlations, makes the assumption that two terms are considered semantically related if they frequently co-occur in the same documents. Thus, a document is represented by the embedding shown in Eq. 3.

\[
\phi(d) = \phi(d) D
\]

The corresponding GVSM kernel between two documents \( d_p \) and \( d_q \) is shown in Eq. 4.

\[
K(d_p, d_q) = \langle \phi(d_p) D \phi(d_q)^\top \rangle
\]

In Eq. 4, the \((ij)^{th}\) entry of the matrix \( \mathcal{D} \mathcal{D}^\top \) is given by the following formula, given that the term-document matrix \( D \) contains the TF-IDF weights for each of the terms, in each of the documents:

\[
\langle \mathcal{D} \mathcal{D}^\top \rangle_{ij} = \sum_{y} \text{tfidf}(t_i, d) \cdot \text{tfidf}(t_j, d)
\]

The matrix \( \mathcal{D} \mathcal{D}^\top \) has a non-zero entry \( \langle \mathcal{D} \mathcal{D}^\top \rangle_{ij} \) if there is a document \( d \) in which the corresponding terms \( t_i \) and \( t_j \) co-occur.

### 3.2. Relatedness measures

The problem of computing semantic relatedness or similarity between two terms has been studied for many decades in the bibliography, and there are many approaches that may be categorized according to their design principles into: (i) knowledge-based, (ii) corpus-based, and (iii) hybrid. Approaches of the first category use a word thesaurus or an ontology and its structure to compute the semantic similarity or relatedness between two terms. The second category approaches analyze statistically large corpora to infer latent similarities between terms, typically based on their co-occurrence in the analyzed corpus. Finally, approaches of the latter category combine typically the structure of an ontology and statistical information gathered for the concepts and their labels from the analysis of a large text corpus. For an overview of the most representative approaches in each category, the reader may wish to consult the work of Budanitsky and Hirst [9], or the more recent work of Zhang et al. [10]. In the following subsections we focus on the three measures that we are employing for the implementation of the suggested kernel.

#### 3.2.1. Omiotis

The first measure that we use, is the Omiotis measure [1]. Omiotis defines the semantic relatedness between a pair of terms as shown in Eq. 6, where the knowledge-base \( O \) is WordNet (WN).

\[
\text{SR}_{\text{omiotis}}(t_i, t_j) = \max_{m} \max_{k} \{ \text{SCM}(S_{t_i}^{O}, P_{t_j}^{O}) \cdot \text{SPE}(S_{t_j}^{O}, P_{t_i}^{O}) \}
\]

where SCM and SPE are called Semantic Compactness and Semantic Path Elaboration respectively. Their product measures the weight of the path connecting the senses that can be assigned to the terms \( S_{t_i}^{O} \), taking into account: the path length considering all possible paths connecting them \( P_{t_i}^{O} \), the type of the semantic edges comprising it, and the depth of the intermediate nodes in the WN senses hierarchy. The semantic relatedness between two terms \( t_i, t_j \), when \( t_i \in \text{WN} \) and \( t_j \notin \text{WN} \), or vice versa, is considered 0. The intuition behind Eq. 6 is that the semantic relatedness between two terms should be computed based on the highest value path connecting any pair of senses of the two terms. The computation of the value takes into account in tandem all of the aforementioned factors.

#### 3.2.2. Wikipedia-based relatedness

The WLM Wikipedia-based measure of Milne and Witten [2], is a low-cost solution for measuring relatedness between terms using the Wikipedia articles and link structure as a knowledge base. The semantic relatedness between two terms \( t_i \) and \( t_j \) according to WLM is defined as shown in Eq. 7. The intuition behind this formula is that the semantic similarity between two terms becomes higher, as the number of articles pointing to both respective Wikipedia articles increases (i.e., as the percentage of the articles linking to both pages compared to the number of articles linking to either of them increases).
SR_{Omiotis}(t_i, t_j) = \frac{\log(\max(\| \ln(a_i) \|, |\ln(a_j)|)) - \log(\| \ln(a_i) \cap \ln(a_j) \|)}{\log(|\ln(a_j)|) - \log(\min(|\ln(a_j)|, |\ln(a_i)|))}

(7)

where \( \ln(a_j) \) is the number of Wikipedia articles that point to article \( a_j \), which corresponds to term \( t_j \) (similarly for \( \ln(a_i) \)) and \( |W| \) is the number of Wikipedia articles.

3.2.3. Average of Omiotis and Wikipedia-based relatedness

Given Eqs. 6 and 7, we can combine their values into a single relatedness value that considers them both simply by averaging the two scores. This average Omiotis–Wikipedia relatedness score, is shown in the following equation for two terms \( t_i \) and \( t_j \):

\[
SR_{\text{Omiotis}}(t_i, t_j) = \frac{SR_{\text{Omiotis}}(t_i, t_j) + SR_{\text{Wikipedia}}(t_i, t_j)}{2}
\]

(8)

3.2.4. Pointwise mutual information

Pointwise Mutual Information (PMI) is a statistical measure of association, which has been used, for example, to discover collocations by examining how often words co-occur in a corpus [28,29]. In our case, PMI is used to measure the relatedness of two words \( w_1, w_2 \) by examining how often these two words co-occur in sentences of a corpus. The intuition is that if \( w_1 \) and \( w_2 \) are related, the words will co-occur frequently. In the experiments, we used the Tipster corpus, which contains approximately 953 million tokens to estimate PMI \((w_1, w_2)\).

The following definition of PMI can be used for our purposes, where \( P(w) \) is the probability of encountering a sentence containing a word occurrence \( w \), and \( P(w_1, w_2) \) is the probability of encountering a sentence with (at least) two word occurrences (not necessarily adjacent).

\[
\text{PMI}(w_1, w_2) = \log \frac{P(W_1, W_2)}{P(W_1) \cdot P(W_2)}
\]

(9)

If the occurrences of \( w_1 \) and \( w_2 \) in the corpus are completely independent, then \( \text{PMI}(w_1, w_2) = 0 \). If \( w_1 \) and \( w_2 \) always occur together, their PMI score is maximum, equal to \(-\log P(W_1, W_2) = -\log P(W_1) \cdot P(W_2) \). If \( w_1 \) and \( w_2 \) never occur together, then their PMI score approaches \(-\infty\).

More precisely, we use the following normalized form of PMI [30], which returns values in the range \([0, 1]\).

\[
\text{PMI}(w_1, w_2) = \frac{1}{2} \left( 1 - \frac{1}{-\log P(W_1, W_2)} \cdot \log \frac{P(W_1, W_2)}{P(W_1) \cdot P(W_2)} + 1 \right)
\]

(10)

We note that Pecina [29] found PMI to be the best collocation extraction measure; and Newman et al. [31] found it to be the best measure of ‘topical coherence’ for sets of words.

3.3. Document clustering

Document clustering algorithms can be categorized into two classes: (i) partitioning algorithms and (ii) hierarchical algorithms. Partitioning algorithms divide the dataset into a number of clusters which are usually optimal in terms of some predefined criterion functions (in Section 3.3.3 representative functions are analyzed). Hierarchical algorithms group the data points into a hierarchical tree structure using bottom-up or top-down approaches. Agglomerative (bottom-up) approaches initially consider each data point as a single cluster and in each iteration build bigger clusters by grouping similar data points or clusters together until the entire dataset is merged into a single cluster in the top level of the hierarchy. Divisive (top-down) approaches split the dataset into smaller clusters until each data point is assigned a single cluster. Spectral clustering algorithms attempt to solve the clustering problem as an optimal partitioning problem.

3.3.1. Clustering algorithms

The K-means [32] is the most known partitioning clustering algorithm. It takes as input the dataset \( D \) and the desired number of clusters \( c \). It initially randomly selects \( c \) documents from \( D \) as the cluster representatives (centroids) and then assigns each document to the most similar centroid. It iteratively recalculates centroids based on the clusters’ documents and reassigns documents to the most similar centroid. The clustering iterations stop when the condition defined by the criterion function is met, or when the maximum number of iterations is exceeded.

The Repeated Bisection (RB) algorithm [33] obtains a \( c \)-way clustering solution by first bisecting the entire collection. Next, one of the two clusters is selected and it is further bisected leading to a total of three clusters. The process of selecting and bisecting a particular cluster is repeated until a number of \( c \) clusters is obtained. The aim of each bisection is to optimize a particular criterion function.

The Biased Agglomerative (Bagglo) algorithm [33] builds on the agglomerative paradigm, whose goal is to locally optimize (minimize or maximize) a particular clustering criterion function by merging two clusters. The merging process is repeated, as previously, until a number of \( c \) clusters is obtained.

Non-negative Matrix Factorization (NMF) [34–36] is a spectral clustering algorithm which finds an optimal partitioning of the document set by finding two non-negative matrices whose product can well approximate the initial non-negative data matrix. The method assumes that \( D \) consists of \( c \) clusters and aims in factorizing the term-document matrix \( D \) into the non-negative \( m \times c \) matrix \( W \), and the non-negative \( c \times n \) matrix \( H \) that minimizes an objective function. \( W \) can be regarded as containing a basis that is optimized for the linear approximation of the data in \( D \).

3.3.2. Algorithms complexity

The various clustering algorithms have different scalability characteristics. Table 1 summarizes the time and space complexity of the discussed clustering algorithms. With respect to time and memory, the most scalable method is the repeated-bisecting algorithm [37]. The least scalable of the algorithms are the ones based on hierarchical agglomerative clustering.

3.3.3. Clustering criterion functions

Clustering algorithms use a global criterion function whose optimization controls the whole clustering process. Therefore, clustering algorithms typically compute a clustering solution for which the value of a particular criterion function is optimized. In this work we use the following three different clustering criterion functions:

1. \( I_2 \) Criterion function: In this approach each cluster \( S_i \), is represented by its centroid \( C_i \) and the \( I_2 \) criterion function (Eq. 11) maximizes the similarity between each document and the centroid of the cluster that is assigned to.

\[
I_2 \text{ maximizes } \sum_{i=1}^{c} \sum_{d \in S_i} \cos(d, C_i) = \sum_{i=1}^{c} \| D_i \|
\]

(11)

The term \( \| D_i \| \) is square-root of the pairwise similarity between all documents in \( S_i \).

2. \( \epsilon_1 \) criterion function is the combination of \( I_2 \) (Eq. 11) and \( \epsilon_1 \) (Eq. 12).

\[
\epsilon_1 \text{ minimizes } \sum_{i=1}^{c} n_i \cos(C_i, C) \iff \text{minimize } \sum_{i=1}^{c} n_i \frac{D_i D_j}{\| D_i \|} \]

(12)

For reference in the algorithm complexities the reader may wish to consult the manual of Chuto clustering toolkit [37].
where $D_r^c D_r$ is sum of all document vectors in the corresponding cluster $r$ and $n_r$ is the size of corresponding cluster $r$.

The $\mathcal{E}_1$ criterion function (Eq. 12) computes the clustering by finding a solution that separates the documents of each cluster from the entire collection. so $H_2$ would be:

$$
\mathcal{H}_2 \overset{\text{max}}{\min} \frac{I_2}{I_2} \iff \minimize \frac{\sum_{r=1}^c |D_r^c|}{\sum_{r=1}^c n_r D_r^c D_r^c / |D_r^c|} \quad (13)
$$

3. $\mathcal{E}_2\mathcal{C}$ Criterion function: Given a non-negative matrix $T$, Non-negative Matrix Factorization (NMF) aims to factor $T$ into two non-negative matrices $W$ and $H$. Lee and Seung [35] formulated NMF as two types of optimization problems in which the approximation error should be minimized under the constraint that $W$ and $H$ are non-negative. In the case of the Euclidean distance, the optimization problem is expressed as follows:

$$
\minimize \frac{f(W, H)}{\|T - WH\|^2}
$$

subject to $H_{ij} \geq 0$, $W_{al} \geq 0, \forall a, l$, where $\| \cdot \|$ represents the Frobenius norm, that is,

$$
\|T - WH\|^2 = \sum_{l=1}^d (T_{lj} - (WH)_{lj})^2
$$

as the objective function $f(W, H)$ is non-convex. Therefore, Lee and Seung [35] proposed the multiplicative update rule for finding a local optimal solution.

4. Semantic smoothing kernels

The Vector Space Model (VSM) represents a document collection by a term-to-document matrix, thus, resulting in a very sparse representation in a high dimensional space. Among its deficiencies is that it assumes term independence and thus fails to capture semantics and that its sparse representation is susceptible to noise [25]. The Generalized Vector Space Model (GVSM) improves VSM by removing the pairwise orthogonality assumption and taking into account the term-to-term correlations, which are based either on the semantic correlations between terms or on the frequency co-occurrence statistics computed in large text corpora. GVSM results in a denser representation model. Latent Semantic Indexing (LSI) attempts to improve GVSVM by means of dimensionality reduction using a new space with fewer orthogonal dimensions (principal dimensions), which are computed on a training corpus and convey most of the variance of the observed data. This work focuses on GVSM; it uses a statistics based kernel as a baseline, evaluates a similarity measure and, finally, introduces a space efficient GVSM (top-k S-VSM) kernel in order to confront the high dimensionality problem.

4.1. A GVSM-based semantic kernel S-VSM

In order to overcome the limitations of VSM in taking into account the semantic similarity of terms, we enrich the BOW representation with semantic information. Instead of extending it with additional dimensions that correspond to concepts, we embed the conceptual relationships of terms in the VSM document representation. More specifically, we re-adjust the weight values of each term in a document using the weights of its semantically related terms according to their pairwise similarities. This transformation maintains the original VSM dimensions, and, thus, does not increase the computational space complexity of the similarity function, and discounts the effects of very general terms in favor of conceptually related terms.

To enrich the BOW representation with semantic information, we construct the semantic relatedness matrix $R$ using a semantic relatedness measure. Specifically, the $R_{ij}$ element of matrix $R$ is given by a knowledge or corpus-based similarity measure, which quantifies the semantic relatedness between terms $T_i, T_j$. Thus, $R$ is a $D \times D$ symmetric matrix with 1’s in the principal diagonal. This smoothing matrix can be used to transform the documents’ vectors in such a way that semantically related documents are brought closer together in the transformed (or feature) space (and vice versa). More formally, the semantically enriched BOW representation of a document $d$ is given as

$$
\phi(d) = \left( \phi(d_i)^T R \right)^T
$$

Following the notation introduced in Section 3.1.1 we define the feature space implicitly via the kernel function. This is particularly important in kernel-based methods or kernel machines when the feature space is very large or even infinite in size. By definition, the kernel function computes the inner product between documents $d_p$ and $d_q$ in the feature space. For our case, this can be written as

$$
K(d_p, d_q) = \phi(d_p)^T \phi(d_q) = \phi(d_p)^T R R^T \phi(d_q)
$$

For this to be a valid kernel function, the Gram matrix $G$ (where $G_{pq} = K(d_p, d_q)$) formed from the kernel function must satisfy the Mercer’s conditions [38]. These conditions are satisfied when the Gram matrix is positive semi-definite. It has been shown in [38] that the matrix $G$ formed by the kernel function (Eq. 15) with the outer matrix product $R R^T$ is indeed a positive semi-definite matrix.

After semantic smoothing, the new weight value of a term $t_i$ in document $d_p$ is calculated as shown in Eq. 16.

$$
\hat{w}_{t_i, d_p} = w_{t_i, d_p} + \sum_{j=1}^m \Psi_{t_j} \cdot w_{t_j, d_p}
$$

where $w_{t_i, d_p}$ is the original weight for term $t_i$ in $d_p$, $w_{t_j, d_p}$ are the original weights for all other terms and $\Psi_{t_j}$ are the relatedness value between terms.

4.2. Top-k S-VSM

The semantic smoothing presented in Eq. 16 corrects the weight of $t_i$ by taking into account the relatedness of $t_i$ with all the terms in the collection. It is straightforward to think that highly related terms will affect the weight of $t_i$ more than terms with limited semantic relation or no relation at all. Though this is taken into account by considering lower weights for less semantically related terms, i.e., embedded as a property directly semantic relatedness is computed, there might be still many weakly semantically related terms that affect the overall re-weighting of the semantic smoothing.

In order to address this problem, in the following we extend the notion of the S-VSM semantic kernel that was presented in the previous section, so that it takes into account the background knowledge of the semantic similarities between terms that are highly related. The kernel retains only the top-k highly semantically

---

**Table 1**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Method</td>
<td>$O((cNNNZ + m))$</td>
<td>$O(NNZ + m + c)$</td>
</tr>
<tr>
<td>RB Method</td>
<td>$O((cNNNZ + \log(c))$</td>
<td>$O(NNZ)$</td>
</tr>
<tr>
<td>Bagllo</td>
<td>$O(n^2 + \log(n))$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>NMF</td>
<td>$O(c + m + n)$</td>
<td>$O(m + n)$</td>
</tr>
</tbody>
</table>

---

11 in our case we employ Omimot, Wikipedia-based or PMI similarities; the three measures that were discussed in the preliminaries section.
related terms (top-k S-VSM) to a given dictionary term \( t_i \). The basic intuition behind the following formulation is that each term \( t_i \) should be reweighed considering only terms with which \( t_i \) shares many highly semantically related neighbors. A similar formulation of this intuition was created by Bloehdorn et al. [39], with the difference being that they considered only weighted parents (hypernyms) of the terms, in an effort to expand the approach of MAVROEIDIS et al. [40] who considered unweighted parents (hypernyms). In the following formulation we consider any semantically related term, in contrast to the previous works that considered only hypernyms, and retain only the top semantically related terms for each term.

We define a virtual document collection \( V \), where each document \( d \in V \) has been created for every term \( t_i \in T \). The document-to-term matrix \( V \) is, hence, a square matrix \( nxn \), where \( n = |T| = |V| \). Given the initialization of a positive integer parameter \( k \), where \( k \leq n \), in every cell of row \( i \) in \( V \) we add \( V_{ij} = \text{Omi}(t_i, t_j) \) iff \( t_j \) belongs to the set of the top \( k \) semantically related terms of \( t_i \), or we add \( V_{ij} = 0 \) in any other case.\(^{12} \)

Finally, for two documents \( d_p \) and \( d_q \) from the original collection \( D \) we apply the GVSM semantic kernel, given \( V \), as shown in the next equation:

\[
k(d_p, d_q) = \sum_{ij} \phi(d_p)_i \phi(d_q)_j
\]

where

\[
(VV^T)_{ij} = \sum_{j=1}^{n} V_{ij} V_{jd}
\]

In the experimental evaluation, we present an analysis of how the selection of \( k \), when using the top-k S-VSM, affects the clustering performance. The rationale behind its design is that the selection of a small \( k \) is a way of speeding up the clustering process. In contrast to the previous works that considered only weighted parents or hypernyms, the top-k S-VSM is definitely computationally faster than the top-k S-VSM, and the experimental evaluation actually shows that the clustering performance is not harmed, compared to the performance of the S-VSM, even when a small \( k \) is selected as a value.

5. Experimental setup

To evaluate the effectiveness of S-VSM and its top-k extension for text clustering we conduct a series of experiments in benchmark text clustering datasets. We employ seven clustering algorithms (four partitioning, two hierarchical, and the non-negative matrix factorization), which we execute with and without semantic enrichment of the VSM, and compare the clustering performance in both cases over five datasets. This section presents our experimental setup; the next section analyzes the results of our experiments.

5.1. Data sets

We used six real data sets, namely Reuters-Transcribed-set, R8 Reuters-21578, ReutersTD, WebKB, Email, and Movie Reviews to evaluate the clustering performance of the aforementioned approaches. In the following, a description of each data set is provided.

1. Movie Reviews (Movies): It contains 2000 reviews of movies from the Internet Movie Database Archive. Half of them are positive sentiments about the movie, and half express a negative opinion. For the purpose of text clustering, we concentrated in discriminating between positive and negative ratings.

2. Email: This dataset comprises e-mails classified into three classes; conference, job, and spam.

3. WebKB: The WebKB dataset comprises web pages collected by the World Wide Knowledge Base project of the CMU text learning group. These pages are manually classified into seven different classes: student, faculty, staff, department, course, project, and other. We discarded the classes staff and department because there are only a few pages from each university. We also discarded the class other because pages were very different in content within this class.

4. Reuters-Transcribed-set (RTS): The dataset is created by selecting 20 files from each of the 10 largest classes in the Reuters-21578 collection. It is available from the UCI KDD Archive.

5. R8 Reuters-21578: It contains 7,674 documents from the Reuters-21578 collection. We are following Sebastian’s convention to selection 8 of the 10 most frequent classes.

6. ReutersTD: It contains 1,504 documents that belong to 13 out of the 135 topic categories Reuters-21578 collection. The same dataset has been employed by [8] and we use it only in order to compare against their knowledge-based VSM model.

Table 2 summarizes the description of the five used datasets for the evaluation and comparison of the clustering performance.

5.2. Document clustering

5.2.1. Algorithms

We use seven different algorithms to test whether the clustering performance could be improved by using the S-VSM or its top-k extension. The selection of such a number of clustering algorithms is imperative, to ensure that the reported results are not dependent on any particular clustering algorithm. Four of the seven algorithms are four K-means variants: (1) Direct K-means with criterion function \( l_2(d - \mu) \), (2) Direct K-means with criterion function \( H_2(d - h) \), (3) Repeated Bisectioning K-means with criterion function \( l_2(rb - \mu) \), and (4) Repeated Bisectioning K-means with criterion function \( h_2(rb - h) \). These four implementations are provided from the CLUTO Toolkit,\(^{13} \) and their selection was based on their very good performance reported in the literature [6]. The fifth clustering algorithm is the RBR with criterion function \( H_2(rb - h) \). Its implementation can also be found in the CLUTO toolkit. The algorithm is similar to the repeated-bisectioning method but the overall solution is globally optimized to further satisfy best the clustering criterion function. The sixth algorithm is a hybrid hierarchical agglomerative clustering algorithm, which can also be found in CLUTO. The algorithm is called Biased Agglomerative (Baggio), with criterion function \( H_2(b - h) \). For clustering \( n \) objects, Baglio first

\(^{12} \) We can also normalize \( V \) per row and column, so that the sum of each row adds to 1; the same with the sum of each column.

\(^{13} \) Available for download from http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.
computes a $\sqrt{n}$-way clustering solution using repeated bisections of individual objects or clusters of objects (rb method). Consequently, it augments the original feature space by adding $\sqrt{n}$ new dimensions. Each object has a non-zero value in the dimension that corresponds to its cluster, which is proportional to the similarity between the object and its cluster centroid. Then, given this augmented representation, the overall clustering solution is obtained by using the traditional agglomerative algorithm. It has been shown in the past that the Baggio algorithm always produces better results than agglomerative clustering algorithms [41]. The seventh algorithm employs Non-negative Matrix Factorization (NMF) and the implementation for text clustering provided by DTU:Toolbox.14

5.2.2. Vector similarity

Cosine measure is the commonly used method to find the similarity between two text documents $d_i , d_k \in D$. It is measured by the cosine of the angle between the vectors $\mathbf{t}_i, \mathbf{t}_k$ representing them:

$$\cos \langle \mathbf{t}_i, \mathbf{t}_k \rangle = \frac{\mathbf{t}_i \cdot \mathbf{t}_k}{\| \mathbf{t}_i \| \cdot \| \mathbf{t}_k \|}$$

If the documents are identical then cosine measure is one, else it would be zero if there is nothing in common between them (i.e., the vectors are orthogonal to each other). For all datasets, we used a stopwords list to remove common words (non descriptive). After that, text documents are processed by the process of lemmatization. The reason behind using lemmatization is to increase the coverage. Tree Tagger15 is used for lemmatization process.

Each method is run 5 times, every time starting with a random initialization of the clusters, and results are reported as average ± standard deviation of the performance measure.

5.2.3. Evaluation measures

The evaluation of the performance of clustering algorithms is done using external criteria (the labels are known for all documents). For our experiments, we set the number of clusters equal to the number of categories (labels, usually built using human assessors) for all the clustering algorithms. To evaluate their performance, we compare the clusters generated by these algorithms with the categories by computing BCubed FMeasure [42]. Amigo et al. [43] showed that BCubed measure is the only measure that satisfy all four desirable constraints (cluster homogeneity, cluster completeness, rag bag, and cluster size versus quantity) for a good measure for cluster evaluation. Let $L(i)$ and $C(i)$ be the category (class) and the cluster of an item $i$, then correctness of the relation between $i$ and $i'$ that share a category is if and only if they are in the same cluster:

$$\text{Correctness}(i, i') = \begin{cases} 1, & \text{if } L(i) = L(i') \land C(i) = C(i') \\ 0, & \text{otherwise} \end{cases}$$

So BCubed precision and recall of an item $i$ is defined as:

$$\text{BCubed Precision}_i = \frac{\text{Number of correct elements in output cluster containing } i}{\text{Number of elements in the output cluster containing } i}$$

$$\text{BCubed Recall}_i = \frac{\text{Number of correct elements in output cluster containing } i}{\text{Number of elements in the category containing } i}$$

So the overall BCubed precision is the averaged precision of all items in the distribution, and BCubed recall is the averaged recall of all items in the distribution. To obtain a single evaluation measure, BCubed precision and recall are combined using the harmonic mean formula:

$$BF = 2 \times \frac{\text{BCubed Precision} \times \text{BCubed Recall}}{\text{BCubed Precision} + \text{BCubed Recall}}$$

The BCubed F-Measure ranges from 0 to 1 with larger values signifying better clusterings.

In addition to the BCubed FMeasure, three more measures are used in the experimental results section, in order to compare in a compatible way the suggested semantic smoothing kernel with other related knowledge-based approaches which enhance the VSM to perform document clustering. These three measures are: Entropy, Purity, and Error rate. Following the same notation as before, but also as in Section 3.3.3, let $S_i$ be a cluster $r$ out of the $k$ clusters that have been produced by the applied clustering methodology. With $L_i$ we denote a class $j$ out of the $k$ classes of documents that the clustering methodology aims to recreate. Ideally, in the optimal case, at the end of the clustering for each cluster $S_i$, with $r \in [1,k]$, there should be a category $L_i$, with $j \in [1,k]$ that it contains exactly the same documents. From this point of view, Purity is a simple measure to compute the proportion of the documents which are correctly clustered, and the Entropy is the sum of the individual cluster entropies weighted by the cluster size. More formally, if $n_i$ and $n_j$ are the numbers of documents contained in cluster $S_i$ and in class $L_j$ respectively, and $n_{rj}$ is the number of documents appearing in both cluster $S_i$ and class $L_j$, and $n$ is the total number of documents in the data set, then Purity is defined as follows:

$$\text{Purity} = \sum_{i=1}^{k} \frac{1}{n} \times \max_{j=1}^{k} n_{rj}$$

and Entropy is defined as follows:

$$\text{Entropy} = \sum_{i=1}^{k} \frac{n_i}{n} \times \left( -\frac{1}{\log k} \sum_{j=1}^{k} n_{rj} \log \frac{n_{rj}}{n_j} \right)$$

In the optimal clustering case, Purity equals to 1 and Entropy equals to 0. Thus, the larger the Purity, the better the clustering result. Similarly, the lower the Entropy, the better the clustering result. Finally, with regards to the Error rate, in the case of a clustering evaluation, it measures the percentage of the document pairs that were supposed to be set in the same cluster, because originally they belonged to the same class, but they were set instead in different clusters.

6. Experimental results and discussion

6.1. Performance of the semantic VSM on different datasets

Our first set of experiments was focused on evaluating S-VSM representations on various clustering algorithms and datasets. The BCubed results for VSM and semantic VSM are shown in Table 3, where each row corresponds to one method and each column corresponds to one representation for a particular dataset. Here results are provided primarily for completeness and in order to evaluate the various methods. The first way of summarizing the results is to average the BCubed FMeasure for each representation over the five different datasets. A number of observations can be made by analyzing the results in Table 4. First, GVSM and S-VSM methods outperform the VSM for almost all datasets. Over the entire set of experiments, GVSM is 1.5–12% and S-VSM is 3–6% better than VSM. GVSM full and OW are very competitive and lead to the best solutions for most of the datasets. GVSM performed well on Movies and WebKB data but OW outperform GVSM on R8 and RTS data. PMI also performed better than VSM in most of the datasets. PMI performed better than OMI and Wiki where sentences are not more structured. When the relative performance of different methods is similar, the average BCubed FMeasure will be quite
similar. Hence to make the comparisons of these representations easier, our second way of summarizing the results is to create a dominance matrix for various representations. We create a $6 \times 6$ dominance matrix (Table 5). The rows and columns of this matrix correspond to the various vector space models, whereas its values correspond to the number of dataset-algorithm pair for which the representation model of the row outperforms the model of the column. For example, the value in the entry of the row TFIDF and the column GVSM is 7, which means for 7 out of the 35 dataset-algorithm pairs, the TFIDF model outperforms GVSM. The values that are close to 35 indicate that the row method outperforms the column method. OMI outperforms the TFIDF as it performed 31 out of 35 times better than the VSM. Similarly Wiki and PMI were also 29 and 24 times better than VSM respectively. GVSM and S-VSM are very competitive. In most of the cases results of S-VSM are better than GVSM results. Within S-VSM, OMI and Omi–Wiki based methods are overall better than other representations.

In order to compare our $S$–VSM model against other knowledge-based VSMs, in the following we provide a comparative evaluation with two other related works. The first is the work by Jing et al. [8], and the second is the work by Liu et al. [44]. In the first comparison, we use the ReutersTD dataset employed in the work by Jing et al. [8]. We compare against the methods reported in [8] using the same evaluation metrics for clustering, i.e., purity and entropy. The results of our S-VSM model using the four different term similarity metrics (Omi, Wiki, PMI, OW) and the first two clustering alternatives ($D - i_2$ and $D - h_2$) are reported in Table 6 along with the results of the methods reported in [8]. Lower entropy and higher purity values correspond to better clustering. As shown in Table 6 our S-VSM compares favorably against the methods of Jing et al. and with the use of Omiotis similarity metric our best results outperform the best results of Jing et al.

For the purposes of the second comparison with the work of Liu et al. [44], we used the 20NG dataset, and more specifically, the five classes about computers, which contain 4881 documents in total.

In Table 7 we report on the comparison between the suggested S-VSM and the numbers reported in [44]. To make the comparison compatible, we report the error rates of the final clustering. Since

### Table 3

Text clustering performance (BCubed F-measure).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Clu. algo.</th>
<th>VSM TFIDF</th>
<th>GVSM Full</th>
<th>S-VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>$D - i_2$</td>
<td>0.319 ± 0.02</td>
<td>0.376 ± 0.02</td>
<td>0.378 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>$D - h_2$</td>
<td>0.318 ± 0.02</td>
<td>0.371 ± 0.01</td>
<td>0.362 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>RB</td>
<td>0.326 ± 0.02</td>
<td>0.332 ± 0.02</td>
<td>0.354 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>RB - h_2</td>
<td>0.306 ± 0.02</td>
<td>0.339 ± 0.01</td>
<td>0.333 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>Omi-Wiki</td>
<td>0.347 ± 0.02</td>
<td>0.345 ± 0.01</td>
<td>0.365 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td>0.358 ± 0.00</td>
<td>0.363 ± 0.00</td>
<td>0.390 ± 0.00</td>
</tr>
</tbody>
</table>

| Movies  | $D - i_2$  | 0.526 ± 0.01 | 0.659 ± 0.01 | 0.555 ± 0.00 |
|         | $D - h_2$  | 0.521 ± 0.01 | 0.659 ± 0.01 | 0.549 ± 0.01 |
|         | RB         | 0.526 ± 0.01 | 0.659 ± 0.01 | 0.555 ± 0.02 |
|         | RB - h_2   | 0.524 ± 0.01 | 0.659 ± 0.01 | 0.549 ± 0.02 |
|         | Omi-Wiki   | 0.518 ± 0.01 | 0.650 ± 0.01 | 0.546 ± 0.02 |
|         | NMF        | 0.508 ± 0.00 | 0.538 ± 0.00 | 0.503 ± 0.00 |

| WebKB   | $D - i_2$  | 0.294 ± 0.01 | 0.401 ± 0.01 | 0.370 ± 0.01 |
|         | $D - h_2$  | 0.284 ± 0.01 | 0.389 ± 0.01 | 0.344 ± 0.01 |
|         | RB         | 0.304 ± 0.01 | 0.415 ± 0.01 | 0.325 ± 0.02 |
|         | RB - h_2   | 0.297 ± 0.01 | 0.393 ± 0.02 | 0.342 ± 0.02 |
|         | Omi-Wiki   | 0.294 ± 0.00 | 0.338 ± 0.00 | 0.315 ± 0.00 |
|         | NMF        | 0.341 ± 0.01 | 0.357 ± 0.04 | 0.372 ± 0.01 |

| Email   | $D - i_2$  | 0.734 ± 0.01 | 0.712 ± 0.05 | 0.723 ± 0.01 |
|         | $D - h_2$  | 0.717 ± 0.01 | 0.701 ± 0.05 | 0.736 ± 0.01 |
|         | RB         | 0.747 ± 0.01 | 0.791 ± 0.01 | 0.774 ± 0.01 |
|         | RB - h_2   | 0.721 ± 0.01 | 0.786 ± 0.01 | 0.768 ± 0.01 |
|         | Omi-Wiki   | 0.717 ± 0.01 | 0.784 ± 0.01 | 0.744 ± 0.01 |
|         | NMF        | 0.747 ± 0.02 | 0.753 ± 0.01 | 0.773 ± 0.01 |

| RB      | $D - i_2$  | 0.519 ± 0.02 | 0.559 ± 0.04 | 0.536 ± 0.02 |
|         | $D - h_2$  | 0.516 ± 0.02 | 0.561 ± 0.04 | 0.540 ± 0.02 |
|         | RB         | 0.550 ± 0.04 | 0.486 ± 0.02 | 0.581 ± 0.03 |
|         | RB - h_2   | 0.517 ± 0.04 | 0.473 ± 0.01 | 0.532 ± 0.03 |
|         | Omi-Wiki   | 0.515 ± 0.03 | 0.481 ± 0.01 | 0.530 ± 0.03 |
|         | NMF        | 0.439 ± 0.00 | 0.463 ± 0.00 | 0.528 ± 0.00 |

### Table 4

Average BCubed F-measure for various vector space models on different datasets.

<table>
<thead>
<tr>
<th>Representation dataset</th>
<th>VSM TFIDF</th>
<th>GVSM full</th>
<th>Omi</th>
<th>Wiki</th>
<th>PMI</th>
<th>OWavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>0.330</td>
<td>0.355</td>
<td>0.355</td>
<td>0.349</td>
<td>0.325</td>
<td>0.361</td>
</tr>
<tr>
<td>Movies</td>
<td>0.519</td>
<td>0.544</td>
<td>0.544</td>
<td>0.534</td>
<td>0.560</td>
<td>0.536</td>
</tr>
<tr>
<td>WebKB</td>
<td>0.300</td>
<td>0.347</td>
<td>0.347</td>
<td>0.362</td>
<td>0.340</td>
<td>0.340</td>
</tr>
<tr>
<td>Email</td>
<td>0.710</td>
<td>0.752</td>
<td>0.752</td>
<td>0.708</td>
<td>0.690</td>
<td>0.710</td>
</tr>
<tr>
<td>RB</td>
<td>0.523</td>
<td>0.512</td>
<td>0.512</td>
<td>0.544</td>
<td>0.554</td>
<td>0.555</td>
</tr>
</tbody>
</table>
Table 5
Dominance matrix for various vector space models evaluated by BCubed FMeasure.

<table>
<thead>
<tr>
<th>Representation</th>
<th>VSM Tfidf</th>
<th>GVSM full</th>
<th>Omi</th>
<th>Wiki</th>
<th>PMI</th>
<th>OWavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM Tfidf</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>GVSM full</td>
<td>28</td>
<td>0</td>
<td>22</td>
<td>26</td>
<td>26</td>
<td>22</td>
</tr>
<tr>
<td>Omi</td>
<td>31</td>
<td>13</td>
<td>0</td>
<td>20</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Wiki</td>
<td>29</td>
<td>9</td>
<td>15</td>
<td>0</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>PMI</td>
<td>24</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>OWavg</td>
<td>29</td>
<td>13</td>
<td>11</td>
<td>17</td>
<td>19</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6
Comparing text clustering performance of knowledge based VSMs in the ReutersTD dataset. Lower entropy and higher purity values in the table designate better clustering performance.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Entropy</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jing et al. [8]</td>
<td>Term</td>
<td>0.3789</td>
<td>0.6418</td>
</tr>
<tr>
<td></td>
<td>Term + concept</td>
<td>0.3358</td>
<td>0.7092</td>
</tr>
<tr>
<td></td>
<td>Term + HS</td>
<td>0.3463</td>
<td>0.6851</td>
</tr>
<tr>
<td></td>
<td>Term + SO</td>
<td>0.3247</td>
<td>0.7013</td>
</tr>
<tr>
<td></td>
<td>Term + AL</td>
<td>0.3225</td>
<td>0.6932</td>
</tr>
<tr>
<td>S-VSM</td>
<td>Omi + D - l2</td>
<td>0.304</td>
<td>0.713</td>
</tr>
<tr>
<td></td>
<td>Omi + D - h2</td>
<td>0.309</td>
<td>0.716</td>
</tr>
<tr>
<td></td>
<td>Wiki + D - l2</td>
<td>0.335</td>
<td>0.698</td>
</tr>
<tr>
<td></td>
<td>Wiki + D - h2</td>
<td>0.341</td>
<td>0.661</td>
</tr>
<tr>
<td></td>
<td>PMI + D - l2</td>
<td>0.459</td>
<td>0.583</td>
</tr>
<tr>
<td></td>
<td>PMI + D - h2</td>
<td>0.471</td>
<td>0.602</td>
</tr>
<tr>
<td></td>
<td>OW + D - l2</td>
<td>0.331</td>
<td>0.697</td>
</tr>
<tr>
<td></td>
<td>OW + D - h2</td>
<td>0.336</td>
<td>0.701</td>
</tr>
</tbody>
</table>

Table 7
Comparing text clustering performance of knowledge based VSMs in the 20NC dataset. Lower error rate in the table designates better clustering performance.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training sample size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Liu et al. [44]</td>
<td>Cosine similarity</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>SNOS</td>
<td>0.3</td>
</tr>
<tr>
<td>S-VSM</td>
<td>Omi</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>Wiki</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>PMI</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>OW</td>
<td>0.29</td>
</tr>
</tbody>
</table>

6.2. Performance of clustering algorithms with different representations

To summarize the results for clustering algorithms we create a 7 × 7 dominance matrix that is shown in Table 8. The rows and columns of this matrix correspond to the various clustering algorithms whereas its values correspond to the number of dataset-VSM pair for which the clustering algorithm of the column outperforms the model of the column. The values that are close to 30 indicate that the row method outperforms the column method. For example, the value in the entry of the row D2 and the column D2 is 26, which means that for 26 out of the 30 dataset-VSM pairs, the D2 outperforms the D2 algorithm. Results validated the previous findings that variants of K-means perform better than other algorithms. D2 performed 26 and 21 times better than D2 and Bagglo respectively. Similarly D2 outclass 25 and 22 times RBB2 and RBB2 respectively. NMF is better than Bagglo in most of the cases. Overall selection criterion l2 is better than all other criterions.

6.3. Performance of the GVSM semantic kernel using the top-k% related terms

Based on their performance among all S-VSM representations, we decided to conduct experiments only for Omiotis and Omiotis-Wiki average based semantic kernels using an increasing number of top-k% terms (from 5% to 95% with a step of 5%). The reason for conducting these experiments is to observe:

1. the impact of less related terms on cluster quality,
2. the algorithms’ stability with respect to varying k (top-k) values.

Figs. 1 and 2 provide in-depth views of the performance of the GVSM-Semantic Kernel methods when the two relatedness measures and only the top-k most related terms are used for smoothing (K – VSMs). The horizontal axis corresponds to the percentage of terms selected each time, whereas the vertical axis shows the BCubed FMeasure values in each dataset. To summarize the results of top-k% terms, we divided the top-k% related terms in 5 ranges, i.e.
5–20, 25–40, 45–60, 65–80, and 85–100. We create a range won matrix that is shown in Table 9. Range won matrix shows the number of times a particular range has better results on a particular dataset-algorithm pair. For example, the value for the entry of the row 5–20% top terms is 35, which means for 35 out of the 70 dataset-algorithm pairs, the 5–20% term range has better results than the other ranges.

From the results, we can easily conclude that:

1. Competitive results are achieved even when only 5% top- \( k \) terms are used. Friedman test also validate that there is no overall statistically significant difference between the 20 slots.
2. In most of the cases, after 60% top values Clustering algorithms stabilize. Bagglo algorithm is more effected due to variation of top- \( k \) values. This is because hierarchical nature of the algorithm. Direct clustering algorithm is the most stabilized algorithm.

![Fig. 1. Omiotis-based top- \( k \) S-VSM.](image1)

![Fig. 2. Top- \( k \) percent based on Omiotis–Wiki average K-VSM.](image2)
3. Reduction in the number of terms used results in reduction of time and space complexities of clustering algorithms and the overall clustering process.

6.4. Significance analysis

To verify the consistency and ranking of the observed results presented in Table 3, we applied the Friedman test with post hoc tests on the observed differences in performances of all methods on all the data sets. The Friedman test is a non-parametric equivalent of ANOVA. The Friedman test works on the assumption that the data come from populations with the same continuous distributions and all observations are mutually independent. These assumptions are desirable for our case because clustering results from separate algorithms may be extremely variable [45].

The null hypothesis for the Friedman test is that there are no differences between the variables. If the calculated probability is low ($P < 0.05$) the null-hypothesis is rejected. There was a statistically significant difference in perceived effort depending on which type of clustering algorithm is used, $\chi^2(2) = 31.030, \rho = 0.000$. Post-hoc analysis with Wilcoxon Signed-Rank Tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $\rho < 0.005$. There was a statistically significant difference in $D - i_2$ vs. Bagglo trial ($Z = -3.436, \rho = 0.001$).

6.5. Performance under varying data size

In order to test the effect of the data size on the performance of our S-VSM in text clustering we used subsets of the WebKB dataset. Subsets were created using stratified sampling of 500, 1000, 2000 and 3000 examples (there are total 4199 examples). To account for the high inherent sampling variance, this approach was repeated 10 times for each of the 4 subset sizes. Direct – $i_2$, Direct – $h_2$, and $RB - i_3$ clustering algorithms are run 5 times each on these subsets, resulting in a total 600 experiments. Fig. 3 presents the performance of the three clustering algorithms using WebKB subsets of varying size. The x-axis shows the sample size and the y-axis shows the average BCubed Fmeasure values for each dataset. We see that the performance is not affected by the number of samples and thus by the dataset size.

6.6. Performance with varying number of clusters K

The next step was to test the effect of the number of clusters in the performance of our S-VSM models. For this reason, we used documents from four classes of the WebKB dataset and the Direct-i2 algorithm. As a result, the expected number of clusters was 4. We evaluated several clustering schemas as output, with a varying number of desired clusters (from 2 to 10), and measured their BCubed Fmeasure. Fig. 4, shows that S-VSM (more especially the Wikipedia based metric) are more resilient than VSM to an increase in the number of clusters. In the case of Wikipedia metric the “knee” in four clusters is obvious.

<table>
<thead>
<tr>
<th>Top k% term range</th>
<th>No. of times win</th>
</tr>
</thead>
<tbody>
<tr>
<td>5–20</td>
<td>35</td>
</tr>
<tr>
<td>25–40</td>
<td>16</td>
</tr>
<tr>
<td>45–60</td>
<td>11</td>
</tr>
<tr>
<td>65–80</td>
<td>7</td>
</tr>
<tr>
<td>85–100</td>
<td>1</td>
</tr>
</tbody>
</table>

Similarly we conducted a Friedman test for the clustering algorithms. There is a statistically significant difference in perceived effort depending on which type of clustering algorithm is used, $\chi^2(2) = 31.030, \rho = 0.000$. Post-hoc analysis with Wilcoxon Signed-Rank Tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $\rho < 0.005$. There was a statistically significant difference in $D - i_2$ vs. Bagglo trial ($Z = -3.436, \rho = 0.001$).

Fig. 3. Performance on subsets of WebKB datasets.

Please cite this article in press as: J.A. Nasir et al., Semantic smoothing for text clustering, Knowl. Based Syst. (2013), http://dx.doi.org/10.1016/j.knosys.2013.09.012
In this paper, we presented a semantic smoothing vector space kernel (S-VSM) for document clustering, and exhaustively tested its performance against VSM and GVSVM using different semantic relatedness measures between words, several document clustering algorithms and five popular document sets. The evaluation results demonstrated that S-VSM dominates VSM in performance in most of the combinations and compares favorably to GVSVM, which uses word co-occurrences to compute the latent similarities between document terms and has increased space complexity. In order to further reduce the complexity of S-VSM we introduced an extension of it, namely the top-k S-VSM, which considers only the top-k semantically related terms. The top-k S-VSM outperforms VSM and is very efficient in terms of time and space complexity. As a next step, we plan to test our representation models to text classification. We also examine the possibility of defining a reduction of the GVSVM semantic kernel, which will allow us to reach top performance, while reducing the overall complexity.

References

