Knowl Inf Syst
DOI 10.1007/s10115-010-0367-z

REGULAR PAPER

Statistical semantics for enhancing document clustering

Ahmed K. Farahat · Mohamed S. Kamel

Received: 30 January 2010 / Revised: 30 July 2010 / Accepted: 9 October 2010

Abstract Document clustering algorithms usually use vector space model (VSM) as their underlying model for document representation. VSM assumes that terms are independent and accordingly ignores any semantic relations between them. This results in mapping documents to a space where the proximity between document vectors does not reflect their true semantic similarity. This paper proposes new models for document representation that capture semantic similarity between documents based on measures of correlations between their terms. The paper uses the proposed models to enhance the effectiveness of different algorithms for document clustering. The proposed representation models define a corpus-specific semantic similarity by estimating measures of term–term correlations from the documents to be clustered. The corpus of documents accordingly defines a context in which semantic similarity is calculated. Experiments have been conducted on thirteen benchmark data sets to empirically evaluate the effectiveness of the proposed models and compare them to VSM and other well-known models for capturing semantic similarity.

Keywords Document clustering · Statistical semantics · Semantic similarity · Term–term correlations

A preliminary version of this paper appeared as Farahat and Kamel (Proceedings of the 2009 IEEE international conference on data-mining workshops, IEEE Computer Society, Washington, DC, pp 459–464, 2009, Proceedings of the eighth workshop on text mining at the tenth SIAM international conference on data mining, SIAM, Philadelphia, pp 83–92, 2010).

A. K. Farahat (✉) · M. S. Kamel
Department of Electrical and Computer Engineering, University of Waterloo,
Waterloo, ON N2L 3G1, Canada
e-mail: afarahat@pami.uwaterloo.ca

M. S. Kamel
e-mail: mkamel@pami.uwaterloo.ca

1 Introduction

Document clustering is a fundamental task in text mining that is concerned with grouping documents into clusters according to their topics. Algorithms for document clustering have been used since the early years of text retrieval systems to organize documents for end-user browsing and to improve the effectiveness and efficiency of the retrieval process. In recent years, document clustering has received considerable attention because of the huge amount of documents that become available on the Web. This drives more research work to improve the efficiency of document clustering algorithms while maintaining their scalability.

Many clustering algorithms have been applied to document data sets. These algorithms include, but not limited to, hierarchical methods [21], k-means [21], spectral clustering [35], and recently non-negative matrix factorization [25]. Most of these algorithms use the vector space model (VSM) [30] as their underlying model for document representation. VSM represents documents as vectors in the space of terms and measures proximity between documents based on the inner-product of their vectors. Some of document clustering algorithms, like k-means and non-negative matrix factorization, are applied directly to document vectors, while others, like hierarchical methods and spectral clustering, are applied to the matrix of cosine similarities.

The use of VSM as the underlying model for document representation totally ignores any semantic relations between terms. This means that documents with no common terms are considered dissimilar even if they have many terms that are semantically related. It accordingly becomes difficult for the clustering algorithm to group these documents together in one cluster. On the other hand, documents with many terms in common are considered similar even if these common terms are noisy and other terms in the two documents have no semantic relatedness. These documents are more likely to be assigned to the same cluster. Although some preprocessing techniques (like stemming and stop-words removal) can be used to reduce lexical variation of terms, these techniques still do not capture aspects of semantic similarity between terms.

Statistical semantics [14] is a general term that is used to describe an aspect of semantic similarity between units of text that is estimated based on the statistical analysis of term occurrence patterns. Different document representation models have been proposed to overcome the limitations of the VSM by capturing statistical semantics between documents. Some of these models, like the generalized vector space models (GVSM) [37], capture statistical semantics in an explicit way by directly estimating measures of correlations between terms. These models, however, have not achieved considerable improvement in related tasks, like information retrieval. In addition, the use of these models with the clustering task has not been fully addressed in the literature. Other models, like latent semantic analysis (LSI) [4], implicitly capture statistical semantics by mapping documents to a low-dimension space. Dimension reduction models, however, are computationally very expensive as they depend on singular value decomposition. In addition, dimension reduction models do not completely capture semantic similarity between terms as they essentially perverse the VSM-based similarity between documents in the low-dimension space.

In this work, we propose new models for document representation that capture semantic similarity between documents using measures of correlations between their terms. We focus on estimating these measures from the corpus of documents to be clustered. The proposed models accordingly capture semantic similarity within the context of this corpus. The contributions of this paper can be summarized as follows:

- We propose new models for document representation that explicitly estimate measures of correlations between terms and use them to quantify semantic similarity between documents. The proposed models extend the original generalized VSM model (GVSM) [37] by developing better estimates of correlations between terms. In comparison with traditional GVSM models, the proposed estimates have many desirable properties and they considerably enhance the effectiveness of well-known algorithms for document clustering.
- We propose new hybrid models that combine the proposed explicit models with latent indexing to represent documents in a low-dimension latent space where document clustering is performed.
- We use a technique for approximate matrix multiplication to reduce the computational complexity of calculating semantic similarity and accordingly the clustering algorithms.
- We conduct an extensive empirical evaluation on thirteen benchmark data sets and compare the proposed models to VSM and other well-known models for estimating semantic similarity.

The rest of the paper is organized as follows. Section 2 presents the necessary background and reviews some related work in document clustering. Section 3 proposes new representation models for estimating semantic similarity based on term–term correlations. Section 4 presents hybrid models that combine explicit with latent semantic analysis. Section 5 discusses how to reduce computational complexity using approximate matrix multiplication. Experiments and results are presented in Sect. 6. Section 7 finally concludes the paper and discusses future work.

2 Background and related work

2.1 Document clustering

Document clustering is an unsupervised learning task that aims at organizing documents into groups according to their similarity. Different aspects of similarity between documents can be defined. The most commonly used aspect is the topic similarity, which is usually estimated based on the proximity of document vectors in the space of terms.

Data clustering algorithms can be generally categorized into hierarchical and partitional [21]. Hierarchical clustering constructs a hierarchy of nested clusters, while partitional clustering divides data points into non-overlapped clusters such that a specific criterion function is optimized.

Hierarchical algorithms are either agglomerative or divisive [21]. Agglomerative algorithms start with singleton clusters, and then successively merge the most similar pair of clusters until all data points are grouped in one cluster. Divisive algorithms start with all data points in one cluster and then successively partition the data points into two dissimilar groups until singleton clusters are obtained. In document clustering, hierarchical agglomerative clustering (HAC) is more common. HAC algorithms differ in the way they calculate similarity between clusters. The most commonly used algorithms are single-link, complete-link, and average-link clustering [21], which measure similarity between two clusters based on the maximum, minimum, and average of their pairwise document similarities, respectively.

Partitional algorithms used for document clustering include, but not limited to, the k-means algorithm [21], spectral clustering [35], and non-negative matrix factorization [25]. The k-means algorithm [21] is the most widely used algorithm for data clustering. The goal of the algorithm is to group data points into k clusters such that the sum of the Euclidean distances between data points in each cluster and its centroid is minimized. Spherical k-means [7] is a variant of the basic k-means algorithm that uses cosine similarity between data points instead of the Euclidean distance. Spherical k-means is usually used with document data sets where the cosine similarity is a more indicative measure of proximity between documents.

Spectral clustering [35] is based on spectral partitioning of a graph whose adjacency matrix encodes measures of similarity between data points. Spectral clustering can be applied to a document similarity graph [17] or to a bipartite graph of terms and documents [5]. The advantage of spectral clustering compared to other algorithms is that the solution is deterministic, and it corresponds to the global minimum of the clustering criterion function. On the other hand, spectral clustering is computationally expensive as it is based on eigenvalue decomposition.[1]

Non-negative matrix factorization (NMF) [25] is a multivariate analysis method that decomposes a non-negative matrix as the product of two non-negative matrices. NMF was introduced in the field of machine learning by Lee and Seung [25] as a method to learn a parts-based representation of data objects. NMF can also be used for data clustering. In this case, the columns of one matrix represent the cluster centroids, and the columns of the other matrix encode the membership values of data points into clusters. Based on this interpretation, Xu et al. [39] suggested a document clustering algorithm that factorizes each document as a linear combination of concepts (or clusters).

Although empirical comparisons showed that partitional algorithms are more efficient and effective than hierarchical algorithms in the document clustering task [41], a hierarchy of document groups is more informative than a flat partitioning as it naturally represents the hierarchy of topics. In addition, hierarchical algorithms are suitable for application in which the number of clusters are not predefined.

2.2 Document representation

Most of the models proposed for document representation originated in information retrieval systems to calculate the relevance of documents to a user query. The most commonly used model for document representation is the vector space model (VSM) [30]. VSM represents documents as vectors in the space of index terms, and measures proximity between documents using the inner-product of their vectors. Given a set of n documents $D = \{d_j : j = 1, \ldots, n\}$ and a set of m terms $T = \{t_i : i = 1, \ldots, m\}$. Let X be an $m \times n$ term–document matrix whose element x_{ij} represents the weight of term t_i inside document c_j. VSM uses the columns of X to directly represent documents. The matrix that encodes the inner-products of document vectors can be calculated as:

$$K_{\text{VSM}} = X^T X, \tag{1}$$

where K_{VSM} is an $n \times n$ matrix, which is called the kernel matrix [33]. The matrix of cosine similarities can be calculated from the kernel matrix as:

$$\text{Sim} = L^{-1/2} K L^{-1/2}, \tag{2}$$

where L is an $n \times n$ diagonal matrix whose elements are the diagonal elements of K.

VSM has been used in many text-mining tasks where it has achieved good results as well as an acceptable computational complexity. However, VSM in its original form assumes that terms are independent and accordingly ignores any semantic relations between them.

[1] The computational complexity of eigenvalue decomposition is $\mathcal{O}(n^3)$, where n is the number of documents.

This assumption implies that proximity between documents does not reflect its true topic similarity. In addition, redundancy in representation increases the dimensionality of document vectors and negatively affects the performance of the underlying algorithms.

The generalized vector space model (GVSM) is a document representation model that was proposed by Wang et al. [37] to overcome the limitations of VSM. The model estimates similarity between documents based on how their terms are related. Wang et al. highlighted that VSM in its original form assumes that term vectors form an orthonormal basis and proposed a new model that removes this orthogonality assumption. In GVSM, term vectors are used as a non-orthogonal basis in which documents are represented. The kernel matrix in the new basis is calculated as:

$$K_{\mathrm{GVSM}} = X^T G X, \tag{3}$$

where G is an $m \times m$ Gram matrix[2] (also called the association matrix), which represents the inner-products of term vectors in some space. The GVSM model in its original form estimates the Gram matrix G by representing terms in an orthonormal basis of 2^m vectors, which represent the min-terms that can be formed by taking different combination of terms. The association measures between terms are calculated as the cosine of the angle between their vectors in the new space. However, as the maximum number of min-terms that appear in n documents is n, the GVSM is usually simplified by assuming that each document has a unique min-term. This assumption means that terms are represented as vectors in the dual space of documents. Accordingly, G can be calculated as: $G = L^{-1/2} X X^T L^{-1/2}$, where L is a diagonal matrix whose elements are the lengths of term vectors in the dual space. Other versions of GVSM [2] calculate G as the inner-products of term vectors in the dual space of documents: $G = X X^T$.

GVSM has been used in information retrieval where it has not achieved much improvement. However, it has been successfully used in multi-lingual information retrieval [2] where documents available in different languages are used to construct the G. The similarity between a user query in one language and documents in another language can be calculated using the GVSM similarity kernel.

Latent semantic indexing (LSI) originally proposed by Deerwester et al. [4] is another document representation model, which is based on decomposing the term–document matrix using singular value decomposition (SVD): $X = U \Sigma V^T$. The leading left and right singular vectors are used to represent terms and documents in some semantic space. Let U_d and V_d be $m \times d$ and $n \times d$ matrices whose columns are the leading d left and right singular vectors of X, respectively, Σ_d is a $d \times d$ diagonal matrix whose elements are the largest d singular values of X. The matrix U_d is directly used to represent terms in the d-dimensional semantic space. Each document is then represented in the semantic space as a linear combination of their term vectors: $X_d^{(\mathrm{LSI})} = U_d^T X$. The corresponding kernel matrix is:

$$K_{\mathrm{LSI}} = X^T U_d U_d^T X. \tag{4}$$

The use of SVD obtains the best rank-d approximation of K in terms of the Frobenius norm[3] of the approximation error: $\|K - K_{\mathrm{LSI}}\|_F$. This means that LSI essentially preserves similarity between documents in VSM.

[2] In linear algebra, the Gram (or Gramian) matrix of a set of vectors is a matrix of the inner-products between these vectors.

[3] The Frobenius norm $\|K\|_F$ is calculated as $\sqrt{\Sigma_{i,j} |k_{ij}|^2}$.

LSI is highly related to the principal component analysis (PCA) [23], which is a well-known method for dimension reduction. PCA is equivalent to LSI if the data matrix used by LSI is column-centered.

Latent semantic kernel (LSK) [3] is a kernel-based version of the LSI method. Given a kernel matrix K, which represents the inner-product of documents in some space, the eigenvalue decomposition of K can be computed as:

$$K = U \Lambda U^T,$$

where U is an $n \times n$ matrix whose columns are the eigenvectors of K, and Λ is an $n \times n$ diagonal matrix whose elements are the eigenvalues of X. The latent semantic indexing could be performed by taking the d eigenvectors of K, which corresponds to the largest eigenvalues U_d, and use them to represent document vectors: $X_k^{(\text{LSK})} = \Lambda_d^{1/2} U_d^T$. Similarly, kernel PCA [31] is a kernel-based version of the PCA method.

Other models for document representation are based on using lexical ontologies, such as WordNet [28], to represent documents in the space of concepts and calculate their similarity. Similar representation models are based on exploiting knowledge from an encyclopedia (like Wikipedia). Explicit semantic analysis (ESA) [15] is such a model that represents terms as vectors in a space of concepts represented by articles from Wikipedia. Wikipedia-based representation models have recently been used to enhance the performance of different text-mining tasks [19,36].

2.3 Representation models for document clustering

Most of document clustering techniques use the VSM for document representation. The clustering algorithm is either applied directly to document vectors (like k-means and NMF) or to the matrix of cosine similarities Sim (Eq. 5) (like hierarchical and spectral clustering). In both cases, the clustering algorithm groups documents into clusters based on the VSM-based similarity, which does not reflect how terms are semantically related.

Dimension reduction techniques have been used with VSM to obtain a low-dimension representation for document clustering algorithms. Some work include the use of latent semantic indexing (LSI) [32], and locality preserving indexing (LPI) [1]. Although dimension reduction techniques implicity capture some aspect of semantic similarity between terms, they essentially map documents to a low-dimension space such that VSM-based similarity is preserved as much as possible. In addition, dimension reduction techniques are computationally very complex as they depend on the singular value decomposition of the data matrix.

Ontology-based models have also been used with document clustering algorithms. Hotho et al. [18] proposed an approach in which terms are augmented or replaced by vectors based on related concepts from WordNet. The concepts vectors are then used with traditional clustering algorithms. Jing et al. [22] similarly tackled the clustering problem by using a new measure of term similarity, which is based on relations between terms in an ontology, such as WordNet. They also proposed an approach to generate an ontology from a text corpus. Recently, Wikipedia has been used to construct concept vectors for document clustering algorithms [19,20]. However, ontology and Wikipedia-based techniques are computationally complex as they depend on mining term–term correlations from extremely large knowledge sources. In addition, general-purpose thesauri, like WordNet, suffer from the presence of noise and irrelevant information. On the other hand, building a domain-specific thesaurus is a difficult task.

Other work on document clustering directly exploits some sort of correlation between terms by dividing them into groups. These groups are then used to guide the partitioning of documents into clusters. Slonim and Tishby [34] proposed an algorithm which first groups terms into clusters such that information about the document corpus is preserved, and then groups documents into clusters such that information about term clusters is preserved. Simultaneous clustering of terms and documents [5] is a related approach which is based on spectral partitioning of a bipartite graph of documents and terms. Recently, Pessiot et al. [29] proposed a similar approach which finds word groups by fitting parametric models. Fung et al. [13] suggested a hierarchical algorithm for document clustering that uses frequent itemsets of terms for building a tree of topics. As the number of terms is much larger than the number of documents, finding a grouping of terms is computationally very complex. In addition, the clustering of terms into groups captures only positive correlations between terms but does not differentiate between uncorrelated and negatively correlated terms.

3 Document clustering using semantic similarity based on term–term correlations

In this section, a method for exploiting correlations between terms in document clustering is described. The method calculates similarity between documents based on the statistical correlations between their terms and then uses these similarities to group documents into clusters. We analyze different similarity models and show how these models can be efficiently used with existing algorithms for document clustering.

3.1 Document similarity models

Let $D = \{d_j : j = 1, \ldots, n\}$ be a set of n documents that contain m terms, X be an $m \times n$ matrix whose element x_{ij} represents the weight of term t_i inside document d_j. The used document similarity model is based on the generalized VSM (GVSM) [37], which assumes that term vectors are linearly independent and represents documents as a linear combination of term vectors. Let $\overrightarrow{d_j}$ and $\overrightarrow{t_i}$ be the vectors of document d_j and term t_i, respectively. These vectors represent documents and terms in some semantic space. We represent the similarity between documents using a kernel matrix [33, Chap. 10] (referred to as semantic kernel), which encodes the inner-products of document vectors in this space. The semantic kernel K that corresponds to GVSM can be generally expressed as:

$$K = X^T G X,$$

where $K = \left[k\left(\overrightarrow{d_i}, \overrightarrow{d_j}\right)\right]_{n \times n}$ is an $n \times n$ kernel (Gram) matrix of document vectors, and $G = \left[\left\langle \overrightarrow{t_i}, \overrightarrow{t_j} \right\rangle\right]_{m \times m}$ is an $m \times m$ kernel (Gram) matrix of term vectors. The cosine similarity between documents can be calculated from the semantic kernel K as:

$$\text{Sim} = L^{-1/2} K L^{-1/2}, \tag{5}$$

where L is a $n \times n$ diagonal matrix whose elements are the diagonal elements of K.

The Gram matrix of term vectors G encodes measures of statistical correlations between terms. These measures could be estimated from the documents to be clustered or a subset of them. In general, this subset could be selected using random sampling or any other deterministic algorithm for identifying representative documents. In Sect. 5, we employ a non-uniform sampling mechanism to select this subset of documents.

Our approach requires G to be positive semi-definite as it represents the inner-products of term vectors in the semantic space. In general, any positive semi-definite matrix that represents some notion of correlation between terms can be used as an estimate of G.

In this section, we analyze different estimates of G, which are based on the association and covariance matrices between terms.

Let $C = \{c_j : j = 1, \ldots, n_c\}$ be the set of n_c documents that are used to estimate G, $Q = \left[q_{ij}\right]_{m \times n_c}$ be an $m \times n_c$ matrix whose element q_{ij} represents the weight of term t_i inside document d_j. According to the simplified GVSM, the correlations between terms can be estimated as the inner-products or cosine similarities of term vectors in the dual space of documents. The matrix of inner-products (also called un-normalized association matrix) can be calculated as: $G_{\text{ASSC}} = QQ^T$, while the matrix of cosine similarities (also called normalized association matrix) can be obtained by dividing the elements of G_{ASSC} by the square root of the Euclidean norm of term weighs: $G_{\text{ASSC\_N}} = L_Q^{-1/2} QQ^T L_Q^{-1/2}$ where L_Q is a $m \times m$ diagonal matrix whose elements are the diagonal elements of QQ^T. We also suggest the use of covariance matrix between terms, and the matrix of Pearson's correlation coefficients to estimate G. The use of these estimates implies the assumption that terms are random variables with Gaussian distributions. Let $\widetilde{Q}$ be a matrix that is obtained from Q by centering its columns. The sample covariance matrix of terms can be calculated as: $G_{\text{COV}} = \frac{1}{n_c - 1} \widetilde{Q}\widetilde{Q}^T = \frac{1}{n_c - 1} QHQ^T$, where $H = I - \frac{1}{n_c} ee^T$ is an $n_c \times n_c$ centering matrix, and e is the all-ones vector of size n_c. Note that $H = HH$ as H is a projection matrix. The matrix of Pearson's correlation coefficients can be calculated by normalizing the covariance matrix: $G_{\text{PCORR}} = \frac{1}{n_c - 1} L_{\widetilde{Q}}^{-1/2} QHQ^T L_{\widetilde{Q}}^{-1/2}$, where $L_{\widetilde{Q}}$ is a $m \times m$ diagonal matrix whose elements are the diagonal elements of QHQ^T.

One problem with the above formulas is that G is non-sparse and its size is $\mathcal{O}\left(m^2\right)$. As the number of terms m in a moderate-size corpus (about $2K$ documents) is more than $30K$ terms, the storage of G in four-bytes single precision requires more than 3 gigabytes of memory. To solve this problem, the semantic kernel $K = X^T GX$ can be calculated without explicitly storing G. In the suggested formulas, G is estimated from the term–document matrix Q using an equation of the form $G = ZZ^T$, where $Z = f(Q)$ is an $m \times n_c$ matrix that is function of Q. In this case, the semantic kernel matrix can be calculated by first calculating matrix $W = Z^T X$, and then calculating $K = W^T W$. The size of matrix W is $\mathcal{O}(nn_c)$, which is very small compared to that of G (as $n,\ n_c \ll m$). W can be used to represent documents in the semantic space which capture correlations between terms, and it can be used directly with vector-based clustering algorithms. The W matrices for different correlation matrices can be written as:

$$W_{\text{ASSC}} = Q^T X, \tag{6}$$

$$W_{\text{ASSC\_N}} = Q^T L_Q^{-1/2} X, \tag{7}$$

$$W_{\text{COV}} = \frac{1}{\sqrt{n_c - 1}} HQ^T X, \quad \text{and} \tag{8}$$

$$W_{\text{PCORR}} = \frac{1}{\sqrt{n_c - 1}} HQ^T L_{\widetilde{Q}}^{-1/2} X. \tag{9}$$

The non-zero elements of L_Q and $L_{\widetilde{Q}}$ are of $\mathcal{O}(m)$, and they can be calculated in an efficient way. The time complexity of calculating W is generally $\mathcal{O}(nn_c m)$. However, X and Q are very sparse term–document matrices and accordingly can be efficiently multiplied. In addition, the multiplication to H can be factorized and calculated in an efficient way with no need to explicitly store H. The semantic kernels that correspond to different estimates of G are:

$$K_{\text{ASSC}} = X^T QQ^T X, \tag{10}$$

$$K_{\text{ASSC\_N}} = X^T L_Q^{-1/2} Q Q^T L_Q^{-1/2} X, \tag{11}$$

$$K_{\text{COV}} = \frac{1}{n_c - 1} X^T Q H Q^T X, \text{ and} \tag{12}$$

$$K_{\text{PCORR}} = \frac{1}{n_c - 1} X^T L_{\tilde{Q}}^{-1/2} Q H Q^T L_{\tilde{Q}}^{-1/2} X. \tag{13}$$

The time complexity of calculating a semantic kernel K is $\mathcal{O}\left(n^2 n_c\right)$ given W, while the time complexity of calculating the basic kernel matrix K_{VSM} given X is $\mathcal{O}\left(n^2 m\right)$. However, W contains few zero elements compared to X, which is very spare. This makes the calculations of semantic kernels more computationally complex than the basic kernel. We will discuss one method to reduce this complexity in Sect. 5.

In the next section, the suggested semantic kernels are analyzed from a geometric perspective.

3.2 Geometric interpretation

The Gram matrix G represents the inner-products of term vectors in some semantic space. In this section, we study, for different estimates of G, the properties of term vectors in the semantic space.

One property to study is the length of a term vector $\overrightarrow{t_a}$ in the semantic space. It is desirable that the length of a term vector reflects its importance in the corpus of documents. This length is equal to the square root of the inner-product of the term vector with itself: $\left\| \overrightarrow{t_a} \right\| = \sqrt{G\left(a, a\right)}$. In the case of G_{ASSC}, the length of a term vector is equal to the sum of its squared weights in all documents of C:

$$\left\| \overrightarrow{t_a} \right\|_{\text{ASSC}} = \sqrt{\sum_{i=1}^{n} q_{ai}^2}.$$

This somehow quantifies how important is the term in the documents of C. If G_{COV} is used, this length is equal to the standard deviation of the term weights in C:

$$\left\| \overrightarrow{t_a} \right\|_{\text{COV}} = \sqrt{\frac{1}{n_c - 1} \sum_{i=1}^{n} \left(q_{ai} - \mu_a\right)^2} = \sigma_a$$

where $\mu_a = \frac{1}{n_c} \sum_{i=1}^{n} q_{ai}$ and σ_a are the mean and standard deviation of the weights of term t_a in C. Standard deviation of term weights is a better measure of the ability of the term to discriminate between documents than the sum of squared weights. Terms with almost the same weight across documents (even if this weight is high) will have very small variance and accordingly very small vector length. On the other hand, terms with higher variance appear frequently in some documents and rarely in others. This measure is very similar to the term variance quality (TVQ) [6], which has been used for feature selection in document clustering. On the other hand, if $G_{\text{ASSC\_N}}$ and G_{PCORR} are used, the length of all term vectors in the semantic space is 1. This means that all terms are assigned the same importance value which is undesirable.

Another property to consider is the cosine similarity between term vectors in the semantic space. It is desirable that the semantic kernel maps positively correlated terms to close points in the semantic space while mapping negatively correlated terms to faraway points. These cosine similarities can be calculated by normalizing elements of G using the lengths of term vectors:

$$\cos\left(\overrightarrow{t_a}, \overrightarrow{t_b}\right) = \frac{G(a,b)}{\sqrt{G(a,a)}\sqrt{G(b,b)}}.$$

In the case of G_{ASSC} and G_{COV}, the cosine similarities between term vectors are equal to $G_{\mathrm{ASSC\_N}}$ and G_{PCORR}, respectively, while in the case of $G_{\mathrm{ASSC\_N}}$ and G_{PCORR}, the cosine similarities between term vectors are equal to their inner-products as all term vectors have unit length. Association-based matrices, G_{ASSC} and $G_{\mathrm{ASSC\_N}}$, do not encode negative correlations between terms. Vectors of negatively correlated and uncorrelated terms are mapped to near-orthogonal directions ($\cos \approx 0$). On the other hand, covariance-based matrices, G_{COV} and G_{PCORR}, map uncorrelated terms to near-orthogonal directions, and negatively correlated terms to opposite directions in the semantic space ($cos \approx -1$).

Based on this analysis, it can be seen that using un-normalized association matrix G_{ASSC} and the covariance matrix G_{COV} could achieve better performance for document clustering because of their ability to automatically weight terms according to their discrimination ability. We can also see that the use of covariance matrix G_{COV} could achieve the best results because negatively correlated terms are mapped to vectors with almost opposite directions.

3.3 Clustering in the semantic space

In this section, the use of existing document clustering techniques with the proposed models is discussed. In general, clustering algorithms that have kernel-based versions can be applied directly to the semantic kernel K. Other algorithms that work on the proximity matrix of data points can be applied to the cosine similarity matrix Sim (Eq. 5). However, the kernel-based versions of some algorithms, like k-means, are more computationally demanding than their original vector-based algorithms. Moreover, some other algorithms, like spherical k-means, cannot be easily kernelized.

One way to avoid using kernel-based versions of these algorithms is to first get an orthonormal basis for the semantic space represented by G, and then apply vector-based clustering algorithms on document vectors in this basis. An orthonormal basis for the semantic space can be obtained by calculating eigenvalue decomposition of G: $G = U\Lambda U^T$, where U is an $m \times m$ matrix whose columns are the eigenvectors of G, and Λ is a $m \times m$ diagonal matrix whose diagonal elements are the eigenvalues of G. The term and documents vectors can accordingly be represented in the semantic space as:

$$T_s = \Lambda^{1/2}U^T,$$
$$D_s = \Lambda^{1/2}U^T X,$$

where the columns of T_s and D_s are the vectors of terms and documents, respectively. Document clustering algorithms that are vector based can be directly applied on the columns of D_s. This method, however, is computationally infeasible because the Gram matrix G cannot even be stored as discussed in Sect. 4.1.

A more efficient method is to approximate the document vectors in the semantic space by representing the semantic kernel as $K = W^T W$ and then using the columns of W to represent document vectors. We show that this method when applied with k-means and spherical k-means is equivalent to applying the algorithms on the document vectors in the orthonormal basis.

We first consider the case when the k-means algorithm is applied on the document vectors in the orthonormal basis. The centroids of clusters can be expressed in terms of document vectors in the orthonormal basis as:

$$\vec{\mu_j} = \frac{1}{|\pi_j|} \sum_{d_i \in \pi_j} \vec{d_i} = \Lambda^{1/2} U^T \vec{\tilde{x}_j}$$

where $\vec{\mu_j}$ is the centroid of cluster π_j, $\vec{\tilde{x}_j} = \frac{1}{|\pi_j|} \sum_{d_i \in \pi_k} \vec{x_i}$ is the mean of column vectors of X that represent documents of cluster π_j. During the assignment step, the new cluster labels of documents are calculated as:

$$y_i = \arg\min_j \left\| \vec{d_i} - \vec{\mu_j} \right\|$$

The square of the Euclidean distance can be written as:

$$\begin{aligned} \left\| \vec{d_i} - \vec{\mu_j} \right\|^2 &= \left(\vec{d_i} - \vec{\mu_j} \right)^T \left(\vec{d_i} - \vec{\mu_j} \right) \\ &= \left(\vec{x_i} - \vec{\tilde{x}_j} \right)^T U \Lambda U^T \left(\vec{x_i} - \vec{\tilde{x}_j} \right) \end{aligned}$$

If the semantic kernel K is represented as $K = W^T W$ such that $W = Z^T X$. The k-means algorithm can be applied on the columns of W. In this case, the centroids of clusters, and the square of the Euclidean distances can be written as:

$$\vec{\hat{\mu}_j} = \frac{1}{|\pi_j|} \sum_{d_i \in \pi_j} \vec{w_i} = Z^T \tilde{x}_j$$

$$\left\| \vec{w_i} - \vec{\hat{\mu}_j} \right\|^2 = \left(\vec{x_i} - \vec{\tilde{x}_j} \right)^T Z Z^T \left(\vec{x_i} - \vec{\tilde{x}_j} \right)$$

Although $\vec{\hat{\mu}_k} \neq \vec{\mu_k}$, the Euclidean distances between documents and centroid vectors are the same as distances in the orthonormal basis:

$$\left\| \vec{w_i} - \vec{\hat{\mu}_j} \right\|^2 = \left(\vec{x_i} - \vec{\tilde{x}_j} \right)^T G \left(\vec{x_i} - \vec{\tilde{x}_j} \right) = \left\| \vec{d_i} - \vec{\mu_j} \right\|^2$$

This means that applying k-means on the columns of W is equivalent to applying the algorithm in the orthogonal space.

In the case of spherical k-means algorithm, it can be shown in a similar way that cosine similarities between documents and centroids in the orthonormal basis are equal to the cosine similarities between the columns of W and the centroid vectors calculated based on W:

$$\cos\left(\vec{w_i}, \vec{\hat{\mu}_j} \right) = \frac{x_i^T G \tilde{x}_j}{\sqrt{x_i^T G x_i} \sqrt{\tilde{x}_j^T G \tilde{x}_j}} = \cos\left(\vec{d_i}, \vec{\mu_j} \right)$$

Algorithms 3.1 and 3.2 show the steps of applying spherical k-means, and hierarchical clustering in the semantic space, respectively. k is the number of clusters, $t_{\max}$ is the maximum number of iterations, and Π is the output partitioning of documents. In the case of hierarchical clustering, similarity between clusters $\mathrm{Sim}\left(\pi_i, \pi_j\right)$ is a function of the cosine similarity between their documents $\mathrm{Sim}\left(d_a, d_b\right)$ depending on the linkage method (e.g., complete, average). In Algorithm 3.1, it should also be noted that the solution of the maximum operation at step 4 is not unique. Two documents could be at equal distance to more than one centroid. In this case, the document will be assigned to one of these clusters at random. The same applies to step 4 of Algorithm 3.2.

If G_{COV} or G_{PCORR} is used to estimate correlations between terms, the kernel matrix K as well as the matrix W will contain negative values. In this case, clustering algorithms that require only non-negative values like NMF and spectral clustering cannot be directly applied. One way to apply these algorithms is to remove negative entries from the matrices. This can be done by setting all negative values to zero or by adding some constant to all the elements of the matrix. In the case of NMF, semi-NMF [8], a variant of NMF that works on matrices with negative values can be used.

Algorithm 3.1 *Spherical k-Means in Semantic Space*
Inputs: X, Q, $t_{\max}$, k
Outputs: $\Pi = \{\pi_1, \ldots, \pi_k\}$
Steps:

```
1. $Z = f(Q)$, $W = Z^T X$
2. Initialize: $\Pi_0 = \{\pi_1, \ldots, \pi_k\}, t = 1$
3. $\overrightarrow{\mu_j} = \frac{\sum_{x_i \in \pi_j} \overrightarrow{w_i}}{\left\| \sum_{x_i \in \pi_j} \overrightarrow{w_i} \right\|}, \ j = 1..k$
4. $y_i = \arg\max_j \cos\left(\overrightarrow{w_i}, \overrightarrow{\mu_j}\right), \ i = 1..k$
5. $\pi_j = \{x_i : \ y_i = j\}, \ j = 1..k$
6. $\Pi_t = \{\pi_1, \ldots, \pi_k\}$
7. If $(\Pi_t \neq \Pi_{t-1}$ & $t < t_{\max})$ $t = t + 1$, Go to step 3.
   Else Return $\Pi_t$
```

Algorithm 3.2 *Hierarchical Agglomerative Clustering in Semantic Space*
Inputs: X, Q, $t_{\max}$, k
Outputs: $\Pi = \{\pi_1, \ldots, \pi_k\}$
Steps:

```
1. $Z = f(Q)$, $W = Z^T X$
2. $K = W^T W$, $\text{Sim} = L^{-1/2} K L^{-1/2}$
3. Initialize: $\Pi_0 = \{\pi_1, \ldots, \pi_n\}, \pi_i = \{x_i\}, i = 1..n, t = 1$
4. $\{\pi_a, \pi_b\} = \arg\max_{\pi_i, \pi_j} \text{Sim}\left(\pi_i, \pi_j\right)$
5. $\pi_c = \pi_a \cup \pi_b$
6. $\Pi_t = \{\Pi_t / \{\pi_a, \pi_b\}\} \cup \pi_c$
7. If $(|\Pi_t| > k)$, $t = t + 1$, Go to step 4.
   Else Return $\Pi_t$
```

4 Document clustering using hybrid models for semantic similarity

This section proposes new hybrid models for document representation, which combine explicit similarity models proposed in Sect. 3 with dimension reduction techniques. The models first construct a semantic space in which similarity between documents encodes how their terms are statistically correlated. Dimension reduction algorithms are then applied to document vectors in the semantic space to obtain latent concepts.

Hybrid models differ from traditional latent models in the properties of documents they preserve in the latent space. Latent semantic models map documents to a low-dimension space based on singular value decomposition (SVD). SVD obtains a low-dimension representation such that the error of approximating data and kernel matrices (in terms of Frobenius norm) is minimum. This means that low-dimension representation essentially preserves the VSM-based similarity, which ignores any semantic relations between terms. However, these models implicitly capture some correlations between terms as a result of ignoring some dimensions. The proposed hybrid models, on the other hand, preserve semantic similarity that is explicitly estimated based on term–term correlations. We empirically show that preserving explicit measures of semantic similarity is more effective than preserving VSM-based similarity, and it achieves better performance with the document clustering task.

4.1 Mapping documents to latent semantic space

Given the representation of documents in the semantic space, W (Eqs. 10–13), dimension reduction techniques can be applied to the matrix W or to the kernel matrix $K = W^T W$ to obtain a concise representation of document vectors that preserves semantic similarity between documents. Dimension reduction removes the noise or irrelevant information in the original term–document matrix and in the calculation of term–term correlations. It also allows documents to be clustered in a more efficient way.

In our experiments, we use both the latent semantic indexing (LSI), and the principal component analysis (PCA) methods for reducing the dimension of document vectors. In the case of LSI, the singular value decomposition of W is calculated as:

$$W = U \Sigma V^T, \tag{14}$$

where U and V are $n \times n$ matrices whose columns are the left and right singular vectors of W, respectively. The singular vectors that correspond to the leading singular values can be used to represent the document vectors in the latent semantic space as follows:

$$W_d = U_d^T W = \Sigma_d V_d^T, \tag{15}$$

where W_d is an $d \times n$ matrix whose columns represents the document vectors in the latent semantic space. U_d and V_d are $n \times d$ matrices whose columns are the leading d left and right singular vectors of W, respectively.

In the case of PCA, the representation of document vectors can be obtained by applying the singular value decomposition on the matrix $\widetilde{W}$ obtained by centering the columns of W.

The representation of document in the semantic space can also be obtained by applying the latent semantic kernel (LSK) to the kernel matrix $K = W^T W$ (or kernel PCA [31] in the case of PCA). The kernel matrix K is decomposed using eigenvalue decomposition: $K = U \Lambda U^T$. The leading eigenvectors U_d are then used to represent the document vectors in the low-dimension space. This method is equivalent to applying LSI on W (or $\widetilde{W}$). The kernel matrix K_d of document vectors in the low-dimension space can be obtained as: $K_d = U_d \Lambda_d U_d^T$.

4.2 Clustering in the latent semantic space

Document clustering algorithms that are vector based (such as k-means) can be applied directly in the semantic space on the columns of W_d. Other algorithms that are based on similarities between documents (like hierarchical clustering) can be applied to the matrix of cosine similarities Sim. Kernel-based algorithms can also be applied to K. Algorithm 4.1 shows the steps of applying spherical k-means in the latent semantic space. k is the number of clusters, d is the number of dimensions in the semantic space, $t_{\max}$ is the maximum number of iterations, and Π is the output partitioning of documents.

Algorithm 4.1 *Spherical k-Means with Hybrid Models*
Inputs: $X, Q, k, d, t_{\max}$
Outputs: $\Pi = \{\pi_1, \ldots, \pi_k\}$
Steps:

```
1. $W = Z^T X$,
2. $[U, \Sigma, V] = svd\,(W)$
3. $W_d = U_d^T W = \Sigma_d V_d^T$
4. Initialize: $\Pi_0 = \{\pi_1, \ldots, \pi_k\}, t = 1$
5. $\overrightarrow{\mu_j} = \frac{\sum_{x_i \in \pi_j} \overrightarrow{w_{di}}}{\left\| \sum_{x_i \in \pi_j} \overrightarrow{w_{di}} \right\|}, \; j = 1..k$
6. $y_i = \arg\max_j \; cos\left(\overrightarrow{w_{di}}, \overrightarrow{\mu_j}\right), \; i = 1..k$
7. $\pi_j = \{x_i : \; y_i = j\}, \; j = 1..k$
8. $\Pi_t = \{\pi_1, \ldots, \pi_k\}$
9. If $(\Pi_t \neq \Pi_{t-1} \; \& \; t < t_{\max})$ $t = t + 1$, Go to 5.
   Else Return $\Pi_t$
```

5 Reducing computational complexity

The calculation of semantic kernels (Eqs. 10–13) are computationally complex compared to VSM. This limits the application of the proposed representation models to large-scale data sets. The formulas used for calculating semantic kernels are in the form of $K = W^T W$, where W is an $n_c \times n$ matrix whose columns represent documents and rows represent the basis of the semantic space. One way to reduce the computational complexity of calculating the semantic kernel K is to select a subset of documents (i.e., rows of W) and use them to estimate G. To do that, we use an approach for approximation matrix multiplication that was proposed by Drineas et al. [10]. This approach approximates a matrix multiplication of the form $W^T W$ by constructing a matrix R from a subset of W's rows selected based on non-uniform random sampling (with replacement). The approximate matrix multiplication is then calculated as:

$$\tilde{K} = R^T R.$$

Let p_i is the probability of selecting document d_i, Drineas et al. [10] suggest that:

$$p_i = \frac{\left|W^{(i)}\right|^2}{\|W\|_F},$$

where $W^{(i)}$ is the row i of matrix W, $\left|W^{(i)}\right|$ is its length, and $\|W\|_F$ is the Frobenius norm of W. It has been shown by Drineas et al. [10] that if the rows of W are sampled with replacement using probabilities p_i's , and the sampled rows are used to construct a matrix R such that:

$$R^{(r)} = \frac{1}{\sqrt{l p_{i_r}}} W^{(i_r)},$$

then the approximation error is bounded as follows [10]:

$$E\left[\left\| W^T W - R^T R \right\|_F\right] \leq \frac{1}{\sqrt{l}} \|W\|_F\,,$$

where l is the number of selected rows.

Table 1 The properties of data sets used to evaluate different similarity models. n, m, and k are the number of documents, terms, and classes, respectively

Data sets	n	m	m_{doc}	k	n_{class}
20 ng	2,000	28,839	23.30 ± 49.10	20	100.0 ± 0.000
Classic	7,094	41,681	06.20 ± 7.700	4	1773.5 ± 971.4
fbis	2,463	2,000	68.50 ± 88.70	17	144.9 ± 139.3
Hitech	2,301	126,321	37.90 ± 27.90	6	383.5 ± 189.9
Reviews	4,069	126,354	43.30 ± 34.80	5	813.8 ± 520.9
la12	6,279	31,472	43.50 ± 38.00	6	1046.5 ± 526.5
tr31	927	10,128	111.9 ± 248.3	7	132.4 ± 124.0
tr41	878	7,454	66.50 ± 100.5	10	87.80 ± 80.10
re0	1,504	2,886	15.00 ± 14.50	13	115.7 ± 173.8
re1	1,657	3,758	15.40 ± 12.30	25	66.30 ± 91.80
k1a	2,340	21,839	44.50 ± 20.80	20	117.0 ± 117.5
k1b	2,340	21,839	44.50 ± 20.80	6	390.0 ± 513.3
wap	1,560	8,460	43.20 ± 20.50	20	78.00 ± 81.10

m_{doc} is the average number of terms per document (after preprocessing), and n_{class} is the average number of documents per class

We use this approach to select a subset of W's rows and use them as a basis to represent the set of documents to be clustered. We then either use R with vector-based clustering algorithms such as spherical k-means or calculate $\tilde{K} = R^T R$ and use it with similarity-based clustering algorithms such as hierarchical clustering. The computational complexity of calculating an approximate semantic kernel $\tilde{K}$ is $\mathcal{O}\left(n^2 l\right)$ given R.

6 Experiments and results

6.1 Data sets

Experiments have been conducted on thirteen benchmark data sets. The properties of different data sets are summarized in Table 1. These data sets have been previously used to evaluate different algorithms for document clustering. The 20 ng is a benchmark data set that consists of newsgroup documents. We used the mini-newsgroups version, which is available at the UCI KDD Archive.[4] The pre-processing steps include the removal of message headers, stop-word removal and stemming. The other data sets were used by Zhao and Karypis [40,41] to evaluate the performance of many algorithms for document clustering. The *classic* data set consists of CACM, CISI, CRANFIELD and MEDLINE abstracts.[5] The *fbis*, *hitech*, *reviews*, *la12*, *tr31* and *tr41* data sets are from TREC collections.[6] The *re0* and *re1* data sets are two subsets of Reuters-21578 [26]. The *k1a*, *k1b* and *wap* data sets are from the WebACE project [16]. We used the pre-processed versions of these data sets distributed with the CLUTO Toolkit [24]. The pre-processing steps that were applied to these data sets are stop-word removal and stemming. In all data sets, the words that appear in one document are removed

[4] http://kdd.ics.uci.edu.

[5] ftp://ftp.cs.cornell.edu/pub/smart.

[6] http://trec.nist.gov.

and the normalized term frequency—inverse document frequency (tf-idf) weighting scheme is used to encode the importance of terms inside documents.

6.2 Experimental setup

Different experiments have been conducted to evaluate the effectiveness of document clustering using the proposed models compared to well-known representation models.

Three document clustering algorithms are used for evaluation: spherical k-means and hierarchical agglomerative clustering (HAC) with both complete and average linkage. For spherical k-means, a MATLAB implementation of Algorithm 4.1 is used. The output clusters are refined by using an incremental optimization technique that moves individual documents between clusters. As Algorithm 4.1 is non-deterministic, it is repeated 10 times using different initial solutions, and the solution with the best value of the objective function is selected. This experiment is repeated 50 times, and the average and standard deviation of quality measures are calculated. In each run of the algorithm, the maximum number of iterations used is 100. For HAC algorithms, the MATLAB function *linkage* is used.

In each experiment, we first use the document representation model to represent document in the semantic space or calculate semantic similarity between documents, and then apply the clustering algorithm to either the representation of document or the similarity matrix.

6.3 Performance evaluation

The clusters obtained by different algorithms are compared to the ground-truth partitioning of documents. In order to evaluate the output of HAC algorithms, a flat partitioning of documents is obtained by traversing the hierarchy from the top cluster until the predefined number of clusters is reached.

We used several quality measures to evaluate the performance of the clustering algorithms. We used F-measure (F), entropy (E) and purity (P). Higher values of F and P and lower values of E indicate better clustering solutions. These measures have been widely used in the literature to evaluate the performance of different clustering algorithms. However, in a recent study by Wu et al. [38], it has been shown that entropy and purity do not capture the uniform effect of the k-means clustering. The study compared different quality measures and suggested the use of the von Dongen criterion (VG) [9] and variation of information (VI) [27]. In some of our experiments, we used normalized versions of VG and VI proposed by Wu et al. [38]. Lower values of VG and VI indicate better clustering solutions.

The used quality measures are calculated as follows. Let n be the total number of documents, n_{ij} be the number of documents that belong to class i and cluster j, n_i be the number of documents in class i, and n_j be the number of documents in cluster j. To calculate F-measure, the precision, recall, and F-measure of mapping class i to cluster j are first calculated as: $P_{ij} = n_{ij}/n_i$,$R_{ij} = n_{ij}/n_j$, $F_{ij} = 2P_{ij}R_{ij}/\left(P_{ij} + R_{ij}\right)$. The F-measure of class i is then calculated as the maximum of F-measure of mapping this class to all clusters: $F_i = \max_j \left\{F_{ij}\right\}$. The overall F-measure is calculated as:

$$F = \sum_{i=1}^{c} \frac{n_i}{n} F_i.$$

The entropy measures the homogeneity of clusters with respect to classes. Let $p_{ij} = n_{ij}/n_i$ be the probability that a member of cluster j belongs to class i, the entropy of a cluster j is calculated as: $E_j = -\sum_{i=1}^{c} p_{ij} log\left(p_{ij}\right)$. The overall entropy is then calculated as:

$$E = \sum_{j=1}^{k} \frac{n_j}{n} E_j.$$

The purity measures the average precision of clusters relative to their best matching classes. The purity of a cluster j is calculated by first assigning cluster j to the most dominant cluster j, and then dividing the number of documents that belong to cluster j and class i by the total number of documents in cluster j: $P_j = \frac{1}{n_j} \max_i \left\{ n_{ij} \right\}$. The overall purity is then calculated as:

$$P = \sum_{j=1}^{k} \frac{n_j}{n} P_j.$$

The van Dongen criterion (VG) [9] was proposed for evaluating graph-partitioning algorithms. VG measures how representative are the majority objects in each class and cluster. We used the normalized version of VG proposed by Wu et al. [38]:

$$\mathrm{VG}^n = \frac{\left(2n - \sum_i \max_j \ n_{ij} - \sum_j \max_i \ n_{ij}\right)}{\left(2n - \max_i \ n_i - \max_j \ n_j\right)}.$$

The variation of information (VI) [27] was proposed as an information theoretic criterion for comparing two clusterings. VI measures the amount of information lost and gained in changing from one clusterings to the other. We used the normalized version proposed by Wu et al. [38]:

$$\mathrm{VI}^n = 1 + 2 \frac{\sum_i \sum_j p_{ij} \log \left(p_{ij} / p_i p_j\right)}{\sum_i p_i \log \left(p_i\right) + \sum_j p_j \log \left(p_j\right)},$$

where $p_{ij} = n_{ij}/n$, $p_i = n_i/n$, and $p_j = n_j/n$.

6.4 Evaluation of explicit models

In our first set of experiments, we compare the four representation models proposed in Sect. 3: ASSC, ASSC_N, COV, and PCORR. In all experiments, the set of all documents to be clustered, D, are used to estimate the correlations between terms.

The VSM-based kernel K_{VSM} is used as our baseline and the improvements in quality measures relative to the baseline are calculated as:

$$q_{Imrov.} (\%) = \left(q - q_{\mathrm{VSM}}\right) / q_{\mathrm{VSM}} \times 100\%. \tag{16}$$

The average of relative quality measures for the 13 data sets is calculated, and the t-test is used to assess the significance of using semantic kernels compared to the VSM-based kernel. In specific, we consider each quality measure and test the null-hypothesis that using semantic similarity achieves no significant improvement in this quality measure. The t-statistic is calculated as:

$$t = \frac{\overline{q}}{s/\sqrt{n}},$$

where $\overline{q}$ and s are the average and standard deviation of relative quality measures, and n is the number of data sets. The value of t-statistic is then compared to the critical value t_{critical} obtained from the t-distribution table for a 95% confidence interval. If $t > t_{\text{critical}}$,

Table 2 The average improvements in quality measures and run-time for different models of semantic similarity relative to VSM (baseline)

Measures\models	ASSC-N	ASSC	PCORR	COV
Spherical k-means				
F-measure (Improv. %) ↑	−00.18 ± 08.59	03.11 ± 07.37	02.24 ± 08.47	**05.75 ± 08.46**
Entropy (Improv. %)↓	00.52 ± 13.58	−01.32 ± 13.00	−03.92 ± 10.08	**−07.35 ± 07.96**
Purity (Improv. %)↑	−02.19 ± 04.85	01.06 ± 05.38	02.10 ± 06.14	**04.94 ± 07.85**
VD^n (Improv. %) ↓	02.88 ± 14.95	−00.85 ± 12.67	−02.18 ± 08.47	**−05.82 ± 06.84**
VI^n (Improv. %) ↓	−02.69 ± 12.90	−03.26 ± 12.21	−03.91 ± 09.56	**−06.89 ± 08.44**
Run-time ($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	03.65 ± 04.10	03.49 ± 03.70	03.25 ± 03.95	03.21 ±03.83
HAC with complete-link				
F-measure (Improv. %) ↑	**20.03 ± 26.08**	**28.54 ± 27.71**	**36.19 ± 39.39**	**47.65 ± 45.73**
Entropy (Improv. %) ↓	−09.39 ± 24.67	**−15.16 ± 21.76**	**−27.93 ± 18.98**	**−33.53 ± 14.88**
Purity (Improv. %)↑	**18.30 ± 21.03**	**26.66 ± 27.30**	**42.56 ± 43.20**	**52.15 ± 51.98**
VD^n (Improv. %) ↓	−16.90 ± 28.20	**−22.38 ± 25.00**	**−26.70 ± 17.72**	**−33.29 ± 15.30**
VI^n (Improv. %) ↓	**−14.67 ± 22.81**	**−19.46 ± 18.13**	**−25.09 ± 16.86**	**−30.71 ± 12.41**
Run-time($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	02.12 ± 01.38	02.09 ± 01.36	01.98 ±01.45	02.00 ± 01.47
HAC with average-link				
F-measure (Improv. %) ↑	*−23.02 ±24.23*	*−11.46 ± 11.43*	**24.21 ± 34.97**	**31.70 ± 38.69**
Entropy (Improv. %) ↓	*44.11 ± 49.79*	*20.20 ± 26.60*	**−17.60 ± 13.18**	**−22.74 ± 14.57**
Purity (Improv. %) ↑	*−22.80 ± 21.88*	*−10.55 ± 10.84*	**27.20 ± 37.37**	**35.43 ± 41.24**
VD^n (Improv. %)↓	*69.92 ± 100.87*	35.83 ± 64.87	**−13.62 ± 21.20**	**−20.70 ±22.76**
VI^n (Improv. %) ↓	*50.95 ± 65.44*	*19.61 ± 31.86*	**−17.72 ± 14.93**	**−22.85 ± 16.37**
Run-time ($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	02.55 ± 01.17	02.49 ± 01.14	02.35 ± 01.29	02.38 ±01.33

Quality measures that are significantly superior/inferior to the VSM (using t-test) are highlighted in bold/italicized

the null-hypothesis is rejected and the semantic kernel is considered to achieve significant improvement in this quality measure.

We also calculate the ratio of the run-time of different methods to the run-time of the baseline method: $r_{\mathrm{model}} = t_{\mathrm{model}}/t_{\mathrm{baseline}}$. The run-time of a method includes the time of calculating the semantic kernel or the representation of documents in the semantic space, and the time taken to cluster documents.

Table 2 shows, for each semantic similarity model, the average quality measures and run-times relative to VSM-based kernels. For F-measure and purity, positive values indicate that the semantic kernel is better than the baseline, while for entropy, van Dongen and variation of information, negative values indicate better performance. For quality measures, semantic kernels that achieve significant improvement (for which the null-hypothesis of the t-test is rejected) are highlighted in bold, while semantic kernels are significantly inferior to the baseline are italicized. If the ratio of run-time is greater than 1, this means that the method takes longer time to run than the baseline method.

We can observe from Table 2 that using covariance-based semantic similarity (COV) achieves significant improvements in all quality measures with the used clustering algorithms. On the other hand, ASSC and ASSC_N achieve significant improvements with the complete-link algorithm, and no significant improvements with spherical k-means. However, they are significantly inferior to the baseline when used with the average-link algorithm. PCORR is significantly superior to the baseline for HAC algorithms.

We can also observe that improvements achieved with HAC algorithms are much higher than those achieved with spherical k-means. We believe this is because partitional algorithms, like spherical k-means, assign data points to cluster such that a global criterion function is optimized. This criterion function is calculated based on the similarity measures between cluster centroids and all data points. So, if there is errors in estimating some of these similarities (e.g., between some document and a centroid), the effect of this errors will be compensated by other measures of similarity (e.g., between the same document and other centroids). On the other hand, HAC algorithms construct clusters in a hierarchical manner, by making a local decision at each level of the hierarchy. In the case of HAC with complete linkage, clusters are merged based on the most dissimilar points in each pair of clusters. If there is an error in the estimate of similarity for this point, this will affect the rest of the cluster hierarchy. The HAC algorithm with average linkage is however less sensitive to errors in estimating similarity between documents as it merges two clusters based on the average of similarities between all points in the two clusters. This means that in case of HAC algorithms, the better the algorithm in estimating semantic similarity, the more the improvement in performance of the hierarchial algorithms.

It should also be noted that there is a large variation in quality measures. For instance, in the case of COV when used with HAC with complete-link, the F-measure is 47%45%. This indicates that semantic similarity models achieve better performance for most of data sets and worse performance for very few. For instance, semantic similarity models are not expected to achieve significant improvements compared to VSM if most of the terms are independent. In practical applications, it is suggested to test semantic similarity models on a small validation set of documents for which the clusters are known, and evaluate whether the proposed models achieve significant improvement for this particular type of documents or not.

The use of semantic similarity models, however, increases the run-time of the clustering algorithms. In the case of spherical k-means, although document vectors in the semantic space has lower dimension than document vectors in the term space ($n \ll m$), they are non-sparse. This makes the calculation of similarity between document vectors and centroids computationally more demanding. In the case of HAC algorithms, the run-time of the clustering algorithm is almost the same, and the increase in run-time is mainly due to the calculation of semantic kernels.

Based on these observations, we can conclude that using explicit models for semantic similarity based on term–term covariance matrix (COV) achieves significant improvement in the performance of the used algorithms for document clustering. This conclusion is consistent with our analysis of different semantic kernels (see Sect. 3.2). However, the improvement in performance comes at the cost of extra run-time. It should also be noted that semantic similarity models could be more useful for data sets with extremely large number of documents. For these data sets, the vector space will be extremely sparse and VSM-based similarity will be less indicative. However, applying proposed models to extremely large data sets requires distributed implementation, which is a subject of future work.

6.5 Evaluation of hybrid models

The second set of experiments evaluate the performance of the proposed hybrid models compared to other document representation models. We compare VSM, COV (semantic similarity based on covariance matrix), LSI, PCA and two hybrid models: LSI-COV (LSI with semantic similarity based on covariance matrix) and PCA-COV (PCA with semantic similarity based on covariance matrix). We select the COV model as it achieves the best performance among other models for explicitly estimating semantic similarity (as discussed in Sect. 6.4).

In the case of representation models that are based on dimension reduction, determining the intrinsic dimension of the semantic space, d, is a common problem in all dimension reduction techniques. Different heuristic exist for estimating d; however, none of them is proved to achieve the best performance. In order to compare different representation models, we used an approach similar to [1] in which the average of the best values of each quality measure for different values of d is used to estimate the performance of the representation model. In particular, we change the number of dimensions in the semantic space, d, from 5 to 100 with increments of 1, and evaluate all quality measures for each value of d. The best 10 values of each quality measure are obtained. The average and standard deviation of these best values are calculated and used to represent the quality of the representation model.

To compare two representation models: M_1, M_2 for a specific clustering algorithm and data set, the average and standard deviation of quality measures are calculated for both representation models: $(\overline{q_1}, s_1)$ and $(\overline{q_2}, s_2)$ (for deterministic models, the standard deviation is 0). The t-test is then used to assess the significance of one method with respect to the other. We consider the null-hypothesis that the two methods are equivalent. The t-statistic is calculated as:

$$t = \frac{\overline{q_1} - \overline{q_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}},$$

where n_1 and n_2 are the number of observations used to estimate $\overline{q_1}$ and $\overline{q_2}$, respectively. The value of t-statistic is then compared to the critical value t_{critical} obtained from the t-distribution table for a 95% confidence interval. If $t > t_{\text{critical}}$, the null-hypothesis is rejected, and the two representation models are considered inequivalent. In this case, if $\overline{q_1}$ is better than $\overline{q_2}$, the representation model M_1 is considered superior to M_2 for this data set.

In this set of experiments, we use an approach similar to that suggested by Zhao and Karypis [40] to compare two representation models over all data sets. In this approach, the quality measures for a particular data set and clustering algorithm are normalized relative to the best value of the quality measure obtained by using different representation models when applying the same clustering algorithm to the same data set. We used relative versions of the F-measure, quality and entropy. In the case of F-measure and purity, the relative measure can be calculated by dividing the value of the quality measure by the maximum of all values:

$$F_r = \frac{F}{\max(F)}, \quad P_r = \frac{P}{\max(P)}.$$

In the case of entropy, the relative entropy E_r is calculated by dividing the minimum value of all entropy values by the original entropy values:

$$E_r = \frac{\min(E)}{E}.$$

The relative quality measures range from 0 to 1. The best representation models have relative quality measures that are close to 1. The higher the relative measures, the better is the representation model. The relative quality measures are then averaged for different data sets. The average values of relative quality measures for two representation models are then compared by applying a statistical significance test on these averages. Similar to [40], we

Table 3 Average of relative quality measures for different representation models and clustering algorithms

Models	VSM	COV	LSI	PCA	LSI-COV	PCA-COV
Algorithms	Relative F-measure					
Spherical k-means	0.9186	0.9668	0.9750	0.9640	0.9780	0.9676
HAC-complete	0.6633	0.9113	0.8503	0.8943	0.9828	0.9742
HAC-average	0.7109	0.8605	0.9160	0.9712	0.9662	0.9557
Algorithms	Relative entropy					
Spherical k-means	0.9000	0.9720	0.9706	0.9619	0.9529	0.9383
HAC-complete	0.5941	0.8897	0.8187	0.8879	0.9732	0.9616
HAC-average	0.6066	0.7887	0.9002	0.9824	0.9462	0.9493
Algorithms	Relative purity					
Spherical k-means	0.9408	0.9826	0.9850	0.9887	0.9857	0.9797
HAC-complete	0.6583	0.9136	0.8997	0.9382	0.9869	0.9809
HAC-average	0.6775	0.8296	0.9219	0.9873	0.9470	0.9517

Table 4 Comparison between different pairs of models (A, B) for each clustering algorithm based on statistical significance (using t-test)

Models	1,2	1,3	1,4	1,5	1,6	2,3	2,4	2,5	2,6	3,4	3,5	3,6	4,5	4,6	5,6
Algorithms	Relative F-measure														
Spherical k-means	$\ll$	$\ll$	=	$\ll$	$\ll$	=	=	=	=	=	=	=	=	=	=
HAC-complete	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\gg$	=	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=
HAC-average	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=	$\ll$	$\ll$	=	=	$\ll$	=	=	=	=
Algorithms	Relative entropy														
Spherical k-means	$\ll$	$\ll$	$\ll$	=	=	=	=	=	=	=	=	=	=	=	=
HAC-complete	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=	=	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=
HAC-average	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=	=	=	=	=
Algorithms	Relative purity														
Spherical k-means	$\ll$	$\ll$	$\ll$	$\ll$	=	=	=	=	=	=	=	=	=	=	=
HAC-complete	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=	=	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=
HAC-average	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	=	$\ll$	$\ll$	$\ll$	=	=	=	=	=	=

The symbols $\gg$, $\ll$, and = indicate that A is significantly superior, inferior, and equivalent to B, respectively

use the paired t-test in which the original quality measures of one representation model M_1 are subtracted from their corresponding measures for the other representation model M_1 for all data sets. The distribution of these differences are tested for statistical significance. Pairs of representation models for which the null-hypothesis is rejected are considered statistically equivalent. In all t-tests, we use a confidence interval of 95%.

Table 3 shows the average of relative quality measures over all data sets for different clustering algorithms. Measures for different pairs of representation models are compared using significance tests. These comparisons are illustrated in Table 4. Each column represents a comparison between two methods "A, B". $\gg$, $\ll$ and = indicate that A is significantly superior, inferior and equivalent to B, respectively.

We can observe from Tables 3 and 4 that using different representation models that capture semantic similarity achieve significant improvement (based on t-test) compared to the basic VSM model for all clustering algorithms. However, the improvement is much larger in the case of agglomerative algorithms than spherical k-means. For instance, when comparing COV with VSM, the improvements in relative F-measure are around 25% for HAC-complete, 15% for HAC-average and 5% for spherical k-means, respectively.

We can also observe from Table 4 that models that capture semantic similarity are statistically equivalent when used with the spherical k-means. In addition, COV is equivalent to LSI (column 2, 3) for most of quality measures. On the other hand, for HAC-complete, PCA is equivalent to COV (column 2, 4) and superior to LSI (column 3, 4) for all quality measures. For HAC-average, PCA is superior to COV (column 2, 4) for all quality measures, and equivalent to LSI (column 3, 4) for most of quality measures.

In the case of hybrid models LSI-COV and PCA-COV, we can observe that they are superior to all other models when used with HAC-complete. For instance, when comparing LSI-COV to LSI, the improvements with HAC-complete are around 13% in F-measure, 15% in entropy and 8% in purity. However, LSI-COV and PCA-COV are statistically equivalent to LSI and PCA, respectively, for HAC-average and spherical k-means. LSI-COV and PCA-COV (column 5,6) are statistically equivalent for all clustering algorithms.

Based on these observations, we can conclude that the effectiveness of different models for estimating semantic similarity depends on how the clustering algorithm works. As discussed in Sect. 6.4, partitional algorithms, like spherical k-means, are more robust to noise in calculating similarities between documents than hierarchical algorithms. We think that is the reason why in the case of spherical k-means, there is no difference in performance when using different models for estimating semantic similarity, and also the improvements achieved by using semantic similarity are small compared to the improvement in the case of HAC algorithms. In addition, that is why some latent models are better than explicit models for HAC algorithms, and hybrid models achieve large improvements compared to latent models in the case of HAC algorithm with complete linkage.

Tables 5, 6, and 7 show, for each data set, the values of F-measure, entropy and purity for the output clustering obtained using different clustering algorithms and document representation models. For each data set and clustering algorithm (a row in the sub-table), the representation models are divided into groups according to the statistical significance between the distribution of their quality measures. The group of methods with the best values is highlighted in bold, while the group with the second best values is italicized. The numbers in these tables are rounded to two decimal places. However, methods have been divided into groups based on the statistical significance of actual numbers. We can observe that the proposed representation models achieve the best performance for many data set, especially when the hierarchical clustering with complete linkage is employed.

The improvement achieved by hybrid models with agglomerative algorithms comes at the cost of additional time complexity. Although the calculation of the W can be done in an efficient way as discussed in Sect. 4.1, the calculation of kernel K for HAC algorithms is more computationally demanding as the matrix W is non-sparse.

6.6 Evaluation of explicit models with approximate multiplication

We finally evaluate the approach suggested in Sect. 5 to reduce the computational complexity of calculating K. In these experiments, we randomly sample a subset of documents with replacement and apply the method described in Sect. 5 to calculate R and $\tilde{K}$. We then apply spherical k-means to the columns of R and HAC algorithms to the approximate semantic

Table 5 *F*-measures of different representation models for each data set

Data sets\models	VSM	GVSM	LSI	PCA	LSI-COV	PCA-COV
Spherical *k*-means						
20 ng	0.41	0.54	0.55	*0.56*	**0.56**	0.55
Classic	0.67	**0.72**	0.66	*0.70*	0.62	0.69
fbis	0.58	0.58	0.58	0.57	*0.59*	**0.59**
Hitech	0.48	*0.50*	*0.51*	*0.51*	**0.52**	0.48
Reviews	*0.74*	0.73	*0.74*	0.71	**0.77**	0.69
la12	**0.72**	**0.73**	**0.73**	**0.72**	**0.73**	**0.73**
tr31	0.67	*0.70*	0.69	0.68	**0.70**	0.69
tr41	0.67	0.67	**0.72**	*0.71*	0.68	0.70
re0	0.45	0.45	0.46	*0.47*	0.46	**0.49**
re1	0.47	0.48	**0.50**	*0.49*	0.49	0.48
k1a	**0.52**	**0.57**	**0.56**	**0.56**	**0.58**	**0.58**
k1b	0.72	0.75	**0.78**	0.69	*0.77*	0.71
wap	*0.50*	**0.57**	**0.56**	**0.55**	**0.58**	**0.58**
HAC with complete linkage						
20 ng	0.16	0.42	*0.49*	*0.49*	**0.50**	**0.50**
Classic	0.45	*0.70*	0.53	*0.65*	*0.72*	**0.75**
fbis	0.55	*0.61*	0.53	0.53	**0.64**	*0.59*
Hitech	0.33	**0.49**	*0.44*	**0.48**	**0.50**	**0.50**
Reviews	0.41	0.59	0.61	*0.71*	**0.78**	*0.70*
la12	0.32	*0.63*	0.55	*0.63*	**0.69**	*0.66*
tr31	0.73	*0.76*	0.60	0.65	**0.80**	**0.79**
tr41	0.59	*0.68*	*0.67*	*0.70*	*0.70*	**0.73**
re0	0.41	0.46	*0.48*	*0.48*	*0.47*	**0.51**
re1	0.32	*0.52*	0.47	0.48	**0.55**	*0.53*
k1a	0.46	*0.61*	*0.58*	*0.57*	*0.62*	**0.64**
k1b	0.48	*0.71*	*0.66*	*0.66*	**0.80**	*0.74*
wap	0.52	0.59	0.57	0.57	*0.62*	**0.64**
HAC with average linkage						
20 ng	0.10	0.22	0.48	*0.52*	0.47	**0.52**
Classic	0.45	*0.63*	0.49	**0.69**	*0.62*	*0.66*
fbis	0.61	0.64	0.65	0.64	**0.67**	*0.67*
Hitech	0.33	*0.53*	0.48	0.51	**0.55**	0.50
Reviews	0.41	0.63	0.54	**0.71**	*0.63*	0.57
la12	0.33	0.59	*0.71*	*0.73*	**0.76**	**0.76**
tr31	0.72	0.81	0.81	*0.84*	**0.86**	0.80
tr41	*0.65*	0.62	**0.76**	**0.76**	*0.70*	*0.73*
re0	*0.41*	**0.50**	**0.49**	**0.51**	**0.52**	**0.51**
re1	0.54	0.59	*0.62*	0.55	**0.64**	0.60
k1a	0.52	0.54	0.60	**0.63**	0.61	*0.62*
k1b	0.81	0.86	**0.90**	*0.90*	0.89	0.89
wap	0.53	0.54	0.60	**0.63**	*0.61*	**0.63**

The best group of models for each data set and algorithm is highlighted in bold, the second best group is italicized

Table 6 Entropy measures of different representation models for each data set

Data sets\models	VSM	GVSM	LSI	PCA	LSI-COV	PCA-COV
Spherical *k*-means						
20 ng	0.62	0.45	0.44	*0.44*	**0.42**	0.44
Classic	0.40	**0.35**	0.41	*0.37*	0.46	0.38
fbis	0.34	0.35	0.34	**0.33**	0.35	*0.34*
Hitech	0.65	**0.62**	0.65	0.65	*0.63*	0.65
Reviews	0.33	*0.33*	0.33	0.36	**0.32**	0.42
la12	0.40	**0.39**	*0.39*	0.39	0.39	0.4
tr31	0.30	0.28	*0.28*	**0.28**	0.30	0.29
tr41	0.26	0.26	**0.23**	*0.23*	0.26	0.24
re0	0.38	*0.37*	**0.37**	**0.37**	**0.37**	**0.36**
re1	0.33	0.31	*0.30*	**0.30**	0.31	0.31
k1a	0.37	**0.32**	0.33	*0.33*	0.33	0.33
k1b	*0.18*	**0.16**	**0.16**	0.19	**0.16**	0.18
wap	0.37	**0.32**	*0.32*	*0.32*	*0.32*	*0.32*
HAC with complete linkage						
20 ng	0.89	*0.58*	**0.50**	**0.51**	**0.49**	**0.49**
Classic	0.92	*0.44*	0.67	0.52	*0.48*	**0.39**
fbis	0.44	*0.36*	0.37	0.38	**0.34**	*0.36*
Hitech	0.92	*0.68*	*0.73*	*0.67*	**0.64**	**0.65**
Reviews	0.85	0.56	0.56	*0.42*	**0.37**	*0.43*
la12	0.91	*0.50*	0.60	*0.53*	**0.47**	*0.49*
tr31	*0.31*	**0.27**	0.40	*0.34*	**0.25**	**0.26**
tr41	*0.35*	**0.23**	**0.25**	**0.24**	**0.25**	**0.23**
re0	*0.59*	**0.40**	**0.39**	**0.38**	**0.38**	**0.38**
re1	0.59	**0.31**	*0.33*	*0.33*	**0.31**	**0.31**
k1a	0.48	0.37	0.35	0.35	*0.34*	**0.32**
k1b	0.55	*0.21*	0.30	*0.22*	**0.17**	*0.20*
wap	0.42	0.36	0.34	0.34	*0.32*	**0.32**
HAC with average linkage						
20 ng	0.97	0.77	*0.50*	**0.47**	*0.50*	**0.47**
Classic	0.93	0.50	0.71	**0.43**	0.50	*0.45*
fbis	0.43	0.37	0.34	*0.33*	**0.32**	*0.33*
Hitech	0.92	0.71	0.70	*0.68*	**0.64**	*0.68*
Reviews	0.87	0.57	0.67	**0.45**	*0.53*	0.61
la12	0.94	0.60	*0.48*	**0.43**	**0.43**	**0.42**
tr31	0.36	*0.24*	*0.24*	**0.21**	**0.21**	*0.23*
tr41	0.35	0.34	**0.23**	**0.23**	0.26	*0.25*
re0	0.61	0.46	**0.39**	**0.38**	**0.39**	*0.40*
re1	0.40	0.34	*0.30*	*0.29*	*0.29*	**0.28**
k1a	0.47	0.44	*0.36*	**0.35**	0.37	*0.35*
k1b	0.31	0.19	**0.15**	*0.16*	**0.15**	**0.15**
wap	0.45	0.44	*0.34*	**0.34**	0.36	0.34

The best group of models for each data set and algorithm is highlighted in bold, the second best group is italicized

Table 7 Purity measures of different representation models for each data set

Data sets\models	VSM	GVSM	LSI	PCA	LSI-COV	PCA-COV
Spherical *k*-means						
20 ng	0.40	0.52	0.54	*0.55*	**0.55**	0.54
Classic	0.76	0.75	0.75	**0.79**	0.72	*0.78*
fbis	0.68	0.68	0.69	*0.69*	0.69	**0.70**
Hitech	0.54	*0.57*	0.56	*0.57*	**0.57**	0.55
Reviews	*0.80*	**0.82**	*0.80*	0.79	**0.82**	0.75
la12	0.78	**0.79**	*0.79*	0.78	*0.79*	0.78
tr31	0.78	0.80	*0.80*	**0.81**	0.79	0.79
tr41	0.78	0.79	**0.81**	*0.81*	0.80	*0.81*
re0	0.65	*0.67*	*0.68*	**0.69**	*0.68*	**0.69**
re1	0.66	0.69	*0.69*	*0.69*	**0.69**	*0.69*
k1a	0.65	**0.69**	0.68	*0.69*	0.68	0.68
k1b	*0.85*	**0.87**	**0.87**	0.84	**0.87**	0.85
wap	0.64	**0.70**	*0.69*	**0.70**	**0.70**	**0.70**
HAC with complete linkage						
20 ng	0.13	0.38	*0.47*	*0.46*	**0.48**	*0.47*
Classic	0.45	*0.70*	0.57	*0.67*	*0.71*	**0.77**
fbis	0.60	*0.66*	*0.65*	0.64	**0.68**	*0.65*
Hitech	*0.28*	**0.51**	**0.47**	**0.52**	**0.55**	**0.54**
Reviews	0.38	0.57	0.66	*0.75*	**0.80**	*0.74*
la12	0.32	*0.68*	0.58	*0.65*	**0.72**	*0.69*
tr31	*0.78*	0.78	0.70	0.74	**0.83**	**0.83**
tr41	0.72	*0.80*	*0.79*	*0.80*	*0.81*	**0.83**
re0	*0.47*	**0.65**	**0.66**	**0.66**	**0.66**	**0.66**
re1	0.41	*0.68*	*0.66*	*0.66*	*0.68*	**0.69**
k1a	0.52	0.63	0.65	0.65	*0.67*	**0.70**
k1b	0.63	*0.84*	0.80	*0.83*	**0.89**	*0.84*
wap	0.58	0.64	0.67	0.67	*0.69*	**0.70**
HAC with average linkage						
20 ng	0.07	0.16	0.46	**0.50**	0.42	*0.49*
Classic	0.45	0.62	0.51	**0.76**	0.63	*0.69*
fbis	0.62	0.67	*0.69*	*0.70*	**0.71**	*0.70*
Hitech	0.27	*0.52*	0.48	0.51	**0.56**	0.50
Reviews	0.35	0.57	0.51	**0.73**	*0.58*	0.55
la12	0.30	0.58	0.73	*0.75*	**0.77**	**0.76**
tr31	0.76	*0.85*	*0.85*	**0.88**	**0.88**	*0.85*
tr41	0.72	0.72	**0.83**	**0.83**	0.79	*0.81*
re0	0.48	0.61	*0.66*	**0.67**	*0.65*	0.64
re1	0.62	0.66	**0.72**	*0.70*	**0.72**	**0.72**
k1a	0.53	0.54	*0.65*	**0.67**	0.64	*0.66*
k1b	0.85	0.86	**0.90**	**0.90**	*0.89*	0.89
wap	0.54	0.55	**0.68**	**0.68**	0.66	*0.67*

The best group of models for each data set and algorithm is highlighted in bold, the second best group is italicized

Table 8 The average improvements in quality measures and run-times relative to VSM (baseline) for models of semantic similarity with approximate matrix multiplication

Measures\models	COV-10%	COV-20%	COV-100%
Spherical k-means			
F-measure(Improv. %) ↑	**03.48 ± 06.44**	**04.99 ± 08.43**	**05.75 ± 08.46**
Entropy (Improv. %) ↓	–03.34 ± 06.39	**–06.01 ± 07.37**	**–07.35 ± 07.96**
Purity (Improv. %)↑	**03.09 ± 05.86**	**04.46 ± 07.62**	**04.94 ± 07.85**
VD^n(Improv. %)↓	–03.30 ± 06.36	**–05.35 ± 07.25**	**–05.82 ± 06.84**
VI^n(Improv. %)↓	–03.16 ± 06.15	**–05.62 ± 07.49**	**–06.89 ± 08.44**
Run-time ($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	01.25 ± 01.61	01.84 ± 02.62	03.21 ± 03.83
HAC with complete-link			
F-measure (Improv. %) ↑	**36.54 ± 34.38**	**39.28 ± 39.11**	**47.65 ± 45.73**
Entropy (Improv. %) ↓	**–25.06 ± 17.90**	**–27.78 ± 17.06**	**–33.53 ± 14.88**
Purity (Improv. %)↑	**43.54 ± 41.21**	**45.96 ± 45.48**	**52.15 ± 51.98**
VD^n(Improv. %)↓	**–26.08 ± 18.24**	**–27.78 ± 17.49**	**–33.29 ± 15.30**
VI^n(Improv. %)↓	**–22.44 ± 15.32**	**–24.93 ± 14.66**	**–30.71 ± 12.41**
Run-time ($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	01.17 ± 00.41	01.37 ± 00.53	02.00 ± 01.47
HAC with average-link			
F-measure (Improv. %) ↑	**25.21 ± 37.84**	**21.25 ± 29.71**	**31.70 ± 38.69**
Entropy (Improv. %) ↓	**–15.36 ± 13.81**	**–14.10 ± 14.32**	**–22.74 ± 14.57**
Purity (Improv. %)↑	**29.80 ± 42.29**	**23.77 ± 31.21**	**35.43 ± 41.24**
VD^n(Improv. %)↓	**–10.83 ± 26.54**	**–12.18 ± 24.21**	**–20.70 ± 22.76**
VI^n(Improv. %)↓	**–14.38 ± 18.69**	**–15.00 ± 18.13**	**-22.85 ± 16.37**
Run-time ($t_{\mathrm{Model}}/t_{\mathrm{VSM}}$)	01.44 ± 00.23	01.69 ± 00.37	02.38 ± 01.33

Quality measures that are significantly superior to the VSM (using t-test) are highlighted in bold

kernel $\tilde{K}$. We compare the output clusters to ground-truth partitioning and calculate the quality measures for each data set. We repeat this experiment with 20 different subsets of sampled documents and calculate the average of all quality measures for different experiments. We then calculate the improvement in average quality measures relative to VSM for each data set using Eq. (16), and then calculate the average and standard deviation of improvements in quality measures (using the same approach in Sect. 6.4.) We also conducted t-test with 95% confidence interval to test the statistical significance of the improvements.

In this set experiments, we use semantic similarity models based on covariance matrix (COV) as they achieve superior performance to other semantic similarity models based on term–term correlations.

Table 8 shows the improvements in average quality measures as well as the run-times when COV model is used with only 10 and 20% of documents selected. We can observe from table that using similarity models based a small subset of documents outperforms VSM, and the achieved improvements are large percentages of the improvement achieved by the COV model with the whole set of documents. We can also observe that the computational complexity is reduced to less than 1.5 times when 10% of documents are used. We could conclude that the choice of the number of documents used for calculating term–term correlations is a trade-off between clustering performance and computational complexity.

7 Conclusions and future work

In this work, the effectiveness of document clustering algorithms has been improved by using similarity models based on statistical correlations between terms. We first define different representation models that explicitly estimate semantic similarity based on term–term correlations. These models are theoretically analyzed, and their performance with different clustering algorithms is evaluated. Results show that similarity models based on covariance matrix between terms achieve the best performance.

In addition, the paper proposes hybrid models for document representation that capture statistical similarity by applying dimension reduction techniques in a semantic space. The paper studies the effectiveness of hybrid models in enhancing document clustering and compares them to well-known models for document representation. Results show that hybrid models are either statistically significant or equivalent to other representation models that capture semantic similarity between documents. Clustering algorithms that are based on making local decisions, such as hierarchical algorithms, are more sensitive to errors in estimating document similarity, and accordingly benefit more from the proposed models.

We finally propose the use of approximate matrix multiplication to reduce computational complexity. A low-dimension representation for documents is calculated based on random sampling with replacement. Experiments show that using this random sampling considerably reduces the run-time while maintaining much of the improvement achieved by the semantic similarity models in the document clustering task.

Future work in semantic analysis for enhancing document clustering includes the study of the problem of determining the intrinsic dimensionality for hybrid models, and other approaches to reduce the computational complexity of semantic mapping and dimension reduction. A distributed implementation of the proposed models to run on extremely large data sets is also a subject of future work.

References

1. Cai D, He X, Han J (2005) Document clustering using locality preserving indexing. IEEE Trans Knowl Data Eng 17(12):1624–1637
2. Carbonell J, Yang Y, Frederking R, Brown R, Geng Y, Lee D (1997) Translingual information retrieval: A comparative evaluation. In: Proceedings of the fifteenth international joint conference on artificial intelligence. Morgan Kaufmann, San Mateo, pp 708–715
3. Cristianini N, Shawe-Taylor J, Lodhi H (2002) Latent semantic kernels. J Intell Inf Syst 18(2):127–152
4. Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci Technol 41(6):391–407
5. Dhillon I (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 269–274
6. Dhillon I, Kogan J, Nicholas C (2003) Feature selection and document clustering. In: Berry M (ed) Survey of Text Mining. Springer, New York, pp 73–100
7. Dhillon IS, Modha DS (2001) Concept decompositions for large sparse text data using clustering. Mach Learn 42(1/2):143–175
8. Ding C, Li T, Jordan MI (2010) Convex and semi-nonnegative matrix factorizations. IEEE Trans Pattern Anal Mach Intell 32:45–55
9. Dongen S (2000) Performance criteria for graph clustering and Markov cluster experiments. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands
10. Drineas P, Kannan R, Mahoney M (2007) Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication. SIAM J Comput 36(1):132–157
11. Farahat AK, Kamel MS (2009) Document clustering using semantic kernels based on term–term correlations. In: Proceedings of the 2009 IEEE international conference on data mining workshops. IEEE Computer Society, Washington, DC, pp 459–464

12. Farahat AK, Kamel MS (2010) Enhancing document clustering using hybrid models for semantic similarity. In: Proceedings of the eighth workshop on text mining at the tenth SIAM international conference on data mining. SIAM, Philadelphia, pp 83–92
13. Fung B, Wang K, Ester M (2003) Hierarchical document clustering using frequent itemsets. In: Proceedings of the third SIAM international conference on data mining. SIAM, Philadelphia, pp 59–70
14. Furnas G, Landauer T, Gomez L, Dumais S (1983) Statistical semantics: analysis of the potential performance of keyword information systems. Bell Syst Tech J 62(6):1753–1806
15. Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of the twentieth international joint conference on artificial intelligence. Morgan Kaufmann, San Mateo, pp 6–12
16. Han E, Boley D, Gini M, Gross R, Hastings K, Karypis G, Kumar V, Mobasher B, Moore J (1998) Web-ACE: a web agent for document categorization and exploration. In: Proceedings of the second international conference on autonomous agents. ACM, New York, pp 408–415
17. He X, Zha H, Ding C, Simon H (2002) Web document clustering using hyperlink structures. Comput Stat Data Anal 41(1):19–45
18. Hotho A, Staab S, Stumme G (2003) WordNet improves text document clustering. In: Proceedings of the SIGIR 2003 semantic web workshop. ACM, New York, pp 541–544
19. Hu X, Zhang X, Lu C, Park EK, Zhou X (2009) Exploiting wikipedia as external knowledge for document clustering. In: Proceedings of the fifteenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 389–396
20. Huang A, Milne D, Frank E, Witten I (2009) Clustering documents using a Wikipedia-based concept representation. In: Proceedings of the thirteenth Pacific-Asia conference on advances in knowledge discovery and data mining. Springer, Berlin, pp 628–636
21. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
22. Jing L, Ng M, Huang J (2010) Knowledge-based vector space model for text clustering. Knowl Inf Syst 25:35–55
23. Jolliffe I (2002) Principal component analysis. Springer, New York
24. Karypis G (2003) CLUTO—a clustering toolkit. Technical Report #02-017, University of Minnesota, Department of Computer Science, Minnesota, MN, USA
25. Lee D, Seung H (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401:788–791
26. Lewis D (1999) Reuters-21578 text categorization test collection distribution 1.0
27. Meila M (2003) Comparing clusterings by the variation of information. In: Learning theory and Kernel Machines. Springer, Berlin, pp 173–187
28. Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41
29. Pessiot J-F, Kim Y-M, Amini MR, Gallinari P (2010) Improving document clustering in a learned concept space. Inf Process Manage 46(2):180–192
30. Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620
31. Scholkopf B, Smola A, Muller K (1997) Kernel principal component analysis. Lect Notes Comput Sci 1327:583–588
32. Schütze H, Silverstein C (1997) Projections for efficient document clustering. In: Proceedings of the twentieth annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '97. ACM, New York, pp 74–81
33. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge
34. Slonim N, Tishby N (2000) Document clustering using word clusters via the information bottleneck method. In: Proceedings of the twenty-third annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 208–215
35. Luxburg U von (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
36. Wang P, Hu J, Zeng H, Chen Z (2009) Using wikipedia knowledge to improve text classification. Knowl Inf Syst 19(3):265–281
37. Wong SKM, Ziarko W, Wong PCN (1985) Generalized vector spaces model in information retrieval. In: Proceedings of the eighth annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 18–25
38. Wu J, Xiong H, Chen J (2009) Adapting the right measures for k-means clustering. In: Proceedings of the fifteenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 877–886

39. Xu W, Liu X, Gong Y (2003) Document clustering based on non-negative matrix factorization. In: Proceedings of the twenty-sixth annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 267–273
40. Zhao Y, Karypis G (2004) Empirical and theoretical comparisons of selected criterion functions for document clustering. Mach Learn 55(3):311–331
41. Zhao Y, Karypis G (2005) Hierarchical clustering algorithms for document datasets. Data Min Knowl Discov 10(2):141–168

Author Biographies

Ahmed K. Farahat received the B.Sc. (with honors) and M.Sc. degrees in Computer Engineering from Cairo University in 2003 and 2007, respectively. He worked as a Research and Teaching Assistant at the Computer Engineering Department at Cairo University from September 2003 to August 2007. He also worked as a Research Engineer at the Human Language Technologies (HLT) group at IBM Cairo Technology Development Center (C-TDC) from January 2005 to August 2007. Since September 2007, Ahmed has been a Ph.D. candidate and Research Assistant at the Pattern Analysis and Machine Intelligence Laboratory at the Department of Electrical and Computer Engineering at University of Waterloo. His research interests include data mining, machine learning and computer vision.

Mohamed S. Kamel Ph.D. (University of Toronto) is University Research Chair Professor and Director of the Pattern Analysis and Machine Intelligence Laboratory at the Department of Electrical and Computer Engineering at the University of Waterloo. Dr. Kamel's research interests are in Computational Intelligence, Pattern Recognition and Machine Learning. He has authored and co-authored over 400 papers in journals and conference proceedings, 11 edited volumes, 16 chapters in books, 2 patents and numerous industrial project reports. Under his supervision, 84 Ph.D. and M.A.SC. students have completed their degrees. He is the Editor-in-Chief of the International Journal of Robotics and Automation, Associate Editor of the IEEE SMC, Part A, Pattern Recognition Letters, Cognitive Neurodynamics journal and Pattern Recognition J. Dr. Kamel is member of ACM, PEO, Fellow of IEEE, the Engineering Institute of Canada (EIC), the Canadian Academy of Engineering (CAE) and the International Association of Pattern Recognition (IAPR).