Knowledge and Information Systems, 31:3, pp. 455−474, 2012
(This is recompilation of the published material.)

Text Document Clustering Using Global Term Context Vectors

Argyris Kalogeratos · Aristidis Likas

Abstract Despite the advantages of the traditional Vector Space Model (VSM) representation, there are known deficiencies concerning the *term independence* assumption. The high dimensionality and sparsity of the text feature space, and phenomena such as polysemy and synonymy can only be handled if a way is provided to measure term similarity. Many approaches have been proposed that map document vectors onto a new feature space where learning algorithms can achieve better solutions. This paper presents the Global Term Context Vector-VSM (GTCV-VSM) method for text document representation. It is an extension to VSM that: i) it captures local contextual information for each term occurrence in the term sequences of documents; ii) the local contexts for the occurrences of a term are combined to define the global context of that term; iii) using the global context of all terms a proper semantic matrix is constructed; iv) this matrix is further used to linearly map traditional VSM (Bag of Words - BOW) document vectors onto a '*semantically smoothed*' feature space where problems such as text document clustering can be solved more efficiently. We present an experimental study demonstrating the improvement of clustering results when the proposed GTCV-VSM representation is used compared to traditional VSM-based approaches.

Keywords Text Mining · Document Clustering · Semantic Matrix · Data Projection

1 Introduction

The text document clustering procedure aims towards automatically partitioning a given collection of unlabeled text documents into a (usually predefined) number

Argyris Kalogeratos
Department of Computer Science, University of Ioannina, GR 45110, Ioannina, Greece
E-mail: akaloger@cs.uoi.gr

Aristidis Likas
Department of Computer Science, University of Ioannina, GR 45110, Ioannina, Greece
E-mail: arly@cs.uoi.gr
Tel.: +30 26510 08810
Fax: +30 26510 08882

of groups, called *clusters*, such that similar documents are assigned to the same cluster while dissimilar documents are assigned to different clusters. This is a task that discovers the underlying structure in a set of data objects and enables the efficient organization and navigation in large text collections.

The challenging characteristics of the text document clustering problem are related to the complexity of the natural language. Text documents are represented in high dimensional and sparse (*HDS*) feature spaces, due to their large term vocabularies (the number of different terms of a document collection, or text features in general). In an HDS feature space, the difference between the distance of two similar objects and the distance of two dissimilar objects is relatively small [4]. This phenomenon prevents clustering methods from achieving good data partitions. Moreover, the text semantics, e.g. term correlations, are mostly implicit and non-trivial, hence difficult to extract without prior knowledge for a specific problem.

The traditional document representation is the *Vector Space Model* (*VSM*) [28] where each document is represented by a vector of weights corresponding to text features. Many variations of VSM have been proposed [17] that differ in what they consider as a feature or '*term*'. The most common approach is to consider different words as distinct terms, which is the widely known *Bag Of Words* (*BOW*) model. An extension is the *Bag Of Phrases* model (*BOP*) [23] that extracts a set of informative phrases or *word n-grams* (n consecutive words). Especially for noisy document collections, e.g. containing many spelling errors, or collections whose language is not known in advance, it is often better to use VSM to model the distribution of *character n-grams* in documents. Herein, we consider word features and we refer to them as *terms*, however, the procedures we describe can be directly extended to more complex features.

Despite the simplicity of the popular word-based VSM version, there are common language phenomena that it cannot handle. More specifically, it cannot distinguish the different senses of a polysemous word in different contexts, or realize the common sense between synonyms. It also fails to recognize multi-word expressions (e.g. '*Olympic Games*'). These deficiencies are in part due to the over-simplistic assumption of *term independence*, where each dimension of the HDS feature space is considered to be vertical to the others, and makes the classic VSM model incapable of capturing the complex language semantics. The VSM representations of documents can be improved by examining the relations between terms either at a low level, such as *terms co-occurrence frequency*, or at a higher *semantic similarity* level.

Among the popular approaches is the *Latent Semantic Indexing* (*LSI*) [7] that solves an eigenproblem using *Singular Value Decomposition* (*SVD*) to determine a proper feature space to project data. *Concept Indexing* [16] computes a k-partition by clustering the documents, and then uses the centroid vectors of the clusters as the axes of the reduced space. Similarly, *Concept Decomposition* [8] approximates in a least-squares fashion the term-by-document data matrix using centroid vectors. A more simple but quite efficient method is the *Generalized Vector Space Model* (*GVSM*) [31]. GSVM represents documents in the document similarity space, i.e. each document is represented as a vector containing its similarities to the rest of the documents in the collection. The *Context Vector Model* (*CVM-VSM*) [5] is a VSM-extension that describes the semantics of each term by introducing a *term context vector* that stores its similarities to the other terms. The similarity between terms is based on a document-wise term co-occurrence frequency. The term context vectors are then used to map document vectors into a feature space of equal size to the original, but less sparse. The *ontology-based VSM* approaches [13, 15] map the terms of the original space onto a feature space defined by a hierarchically structured thesaurus, called *ontology*. Ontologies provide information about the words of a language and their possible semantic relations, thus an efficient mapping can disambiguate the word senses in the context of each document. The main disadvantage is that, in most cases, the ontologies are static and rather generic knowledge bases which may cause heavy semantic smoothing of the data. A special text representation problem is related to very short texts [14, 25].

In this work, we present the *Global Term Context Vector-VSM* (*GTCV-VSM*) representation which is an entirely corpus-based extension to the traditional VSM that incorporates *contextual information* for each vocabulary term. First, the *local context* for each term occurrence in the term sequences of documents is captured and represented in vector space by exploiting the idea of the *Locally Weighted Bag of Words* [18]. Then all the local contexts of a term are combined to form its *global context vector*. Global context vectors constitute a *semantic matrix* which efficiently maps the traditional VSM document vectors onto a semantically richer feature space of same dimensionality to the original. As indicated by our experimental study, in the new space, superior clustering solutions are achieved using well-known clustering algorithms such as the spherical k-means [8] or spectral clustering [24].

The rest of this paper is organized as follows. Sect 2 provides some background on document representation using the Vector Space Model. In Sect 3, we describe recent approaches for representing a text document using histograms that describe the local context at each location of the document term sequence. In Sect 4, we present our proposed approach for document representation. The experimental results are presented in Sect 5, and finally in Sect 6, we provide conclusions and directions for future work.

2 Document Representation in Vector Space

In order to apply any clustering algorithm, the raw collection of N text documents must be first preprocessed and represented in a suitable feature space. A standard approach is to eliminate trivial words (e.g. stopwords) and words that appear in a small number of documents. Then, *stemming* [26] is applied, which aims to replace each word by its corresponding word *stem*. The V derived word stems constitute the collection's *term vocabulary*, denoted as $\mathcal{V}=\{\nu_1,\ldots,\nu_V\}$. Thus, a text document, which is a finite term sequence of T vocabulary terms, is denoted as $d^{seq}=\langle d^{seq}(1),\ldots,d^{seq}(T)\rangle$, with $d^{seq}(i) \in \mathcal{V}$. For example, the phrase '*The dog ate a cat and a mouse!*' is a sequence $d^{seq}=\langle dog,\ ate,\ cat,\ mouse\rangle$.

2.1 The Bag of Words Model

According to the typical VSM approach, the *Bag of Words* (*BOW*) model, a document is represented by a vector $d \in \mathbb{R}^V$, where each word term ν_i of the vocabulary is associated with a single vector dimension. The most popular weighting scheme is the *normalized* $tf \times idf$ that introduces the *inverse document frequency* as an external weight to enforce the terms that have *discrimination power* and appear in a small number of documents. For the ν_i vocabulary term, it is computed as $idf_i = \log(N/df_i)$, where N denotes the total number of documents and df_i denotes the *document-frequency*, i.e. the number of documents that contain term ν_i. Thus, the normalized $tf \times idf$ BOW vector is a mapping of the term sequence d^{seq} defined as follows

$$\Phi_{bow} : d^{seq} \rightarrow d = h \cdot (tf_1 \; idf_1, \ldots, tf_V \; idf_V)^\top \in \mathbb{R}^V, \tag{1}$$

where normalization is performed with respect to the Euclidean norm using the coefficient h. The document collection can then be represented using the N document vectors as rows in the *Document-Term matrix* D, which is a $N \times V$ matrix whose rows and columns are indexed by the documents and the vocabulary terms, respectively.

In the VSM there are several alternatives to quantify the semantic similarity between document pairs. Among them, *Cosine similarity* has shown to be an effective measure [11] and for a pair of document vectors d_i and d_j is given by

$$sim_{cos}(d_i, d_j) = \frac{d_i^\top d_j}{\|d_i\|_2 \|d_j\|_2} \; \in \; [0, 1]. \tag{2}$$

Unit similarity value implies the two documents are described by identical distributions of term frequencies. Note that this is equal to the dot product $d_i^\top d_j$ if document vectors are normalized in the unit positive V-dimensional hypersphere.

2.2 Extensions to VSM

The BOW model, despite having a series of advantages, such as generality, and simplicity, it cannot model efficiently the rich semantic content of text. The Bag Of Phrases model uses phrases of two or three consecutive words as features. Its disadvantage is the fact that it has been observed that as phrases become longer they obtain superior semantic value, but at the same time, they become statistically inferior with respect to single-word representations [19]. A category of methods developed aiming on tackling this difficulty recognize the frequent *wordsets* (unordered itemsets) in a document collection [3, 10], while the method proposed in [20] exploits the frequent word subsequences (ordered) that are stored in a *Generalized Suffix Tree* (*GST*) for each document.

Modern variations of VSM are used to tackle the difficulties occurring due to HDS spaces, by projecting the document vectors onto a new feature space called *concept space*. Each *concept* is represented as a *concept vector* of relations between the concept and the vocabulary terms. Generally, this approach of document mapping can be expressed as

$$\Phi_{VSM} : d \rightarrow d' = Sd \in \mathbb{R}^{V'}, \; V' \leq V, \tag{3}$$

where the $V' \times V$ matrix S stores the concept vectors as rows. This projection matrix is also known as *semantic matrix*. The Cosine similarity between two normalized document images in the concept space can be computed as a dot-product

$$sim_{sem}^{(cos)}(d_i', d_j') = (\overline{Sd_i})^\top (\overline{Sd_j}) = (h_i^S Sd_i)^\top (h_j^S Sd_j) = h_i^S h_j^S (d_i^\top S^\top Sd_j), \tag{4}$$

where the scalar normalization coefficient for each document is $h_i^S{=}1/\|Sd_i\|_2$. The similarity defined in Eq. 4 can be interpreted in two ways: i) as a dot product of the document images $(\overline{Sd_i})^\top(\overline{Sd_j})$ that both belong to the new space $\mathbb{R}^{V'}$ and ii) as a composite measure that takes into account the pairwise correlations between the original features expressed by the matrix $S^\top S$.

There is a variety of methods proposing alternative ways to define the semantic matrix though many of them are based on the above linear mapping. The widely used *Latent Semantic Indexing* (*LSI*) [7] projects the document vectors onto a space spanned by the eigenvectors corresponding to the V' largest eigenvalues of the matrix $D^\top D$. The eigenvectors are extracted by the means of *Singular Value Decomposition* (*SVD*) on matrix $D^\top$ and they capture the latent semantic information of the feature space. In this case, each eigenvector is a different concept vector and V' is a user parameter much smaller than V, while there is also a considerable computational cost to perform the SVD. In *Concept Indexing* [16], the concept vectors are the centroids of a V'-partition obtained by applying document clustering. In [9], statistical information such as the covariance matrix is combined with traditional mapping approaches in to latent space (LSI, PCA) to compose a hybrid vector mapping.

A computationally simpler alternative that utilizes the Document-Term Matrix D as a semantic matrix is the *Generalized Vector Space Model* (*GVSM*) [31], i.e. $S_{gvsm}{=}D$ and the image of a document is given by $d'{=}Dd$. By examining the product $Dd \in \mathbb{R}^{N\times 1}$, we can conclude that a GVSM projected document vector d' has lower dimensionality if $N{\le}V$. Moreover, if both d and D are properly normalized, then image vector d' consists of the N Cosine similarities between the document vector d and the rest of the $N{-}1$ documents in the collection. This observation implies that the GVSM works in the *document similarity space* by considering each document as a different concept. On the other hand, the respective product $S_{gvsm}^\top S_{gvsm}{=}D^\top D$ (used in Eq. 4) is a $V{\times}V$ *Term Similarity Matrix* whose r-th row has the dot-product similarities between term ν_r and the rest of the $V{-}1$ of vocabulary terms. Note that terms become more similar as their corresponding normalized frequency distributions into the N documents are more alike. Based on the GVSM model, it is proposed in [1] to build local semantic matrices for each cluster during document clustering.

A rather different approach proposed in [5] for information retrieval is the *Context Vector Model* (*CVM-VSM*) where, instead of a few concise concept vectors, it computes the context in which each of the V vocabulary terms appears in the dataset, called *term context vector* (*tcv*). This model computes a $V{\times}V$ matrix S_{cvm} containing the term context vectors as rows. Each tcv_i vector aims to capture the V pairwise similarities of term ν_i to the rest of the vocabulary terms. Such similarity is computed using a *co-occurrence frequency measure*. Each matrix element $[S_{cvm}]_{ij}$ stores the similarity between terms ν_i and ν_j computed as

$$[S_{cvm}]_{ij} = \begin{cases} 1 & , i = j \\ \dfrac{\sum_{r=1}^{N} tf_{ri} tf_{rj}}{\sum_{r=1}^{N}(tf_{ri} \cdot \sum_{q=1,\ q\neq i}^{V} tf_{rq})} & , i \neq j. \end{cases} \tag{5}$$

Note that this measure is not symmetric, generally $[S_{cvm}]_{ij} \neq [S_{cvm}]_{ji}$, due to the denominator that normalizes the pairwise similarity to [0, 1] with respect to the 'total amount' of similarity between term ν_i and the other vocabulary terms. The rows of matrix S_{cvm} can be normalized with respect to the Euclidean norm and each document image is then computed as the centroid of the normalized context vectors of all terms appearing in that document

$$\Phi_{cvm} : d \to d' = \sum_{i=1}^{V} tf_i \cdot tcv_i, \tag{6}$$

where tf_i is the frequency of term ν_i. The motivation for using term context vectors is to capture the semantic content of a document based on the co-occurrence frequency of terms in the same document, averaged over the whole corpus. The CVM-VSM representation is less sparse than BOW. Moreover, weights such as *idf* can be incorporated to the transformed document vectors computed using Eq. 6. In [5] several more complicated weighting alternatives have been tested in the context of information retrieval that in our text document clustering experiments did not perform better than the standard *idf* weights.

In a higher semantic level than term co-occurrences, additional information for vocabulary terms provided by ontologies has also been exploited to compute the term similarities and to construct a proper semantic matrix. *WordNet* [22] and *Wikipedia* [30] have been used for this purpose in [6, 15, 29], respectively.

2.3 Discussion

Summarizing the properties of the above mentioned vector-based document representations, in the traditional BOW approach, the dimensions of the term feature space are considered to be independent to each other. Such an assumption is very simplistic, since there exist semantic relations among terms that are ignored. The VSM-extensions aim to achieve *semantic smoothing*, a process that redistributes the term weights of a vector model, or map data in a new feature space, by taking into account the correlations between terms. For instance, if the term '*child*' appears in a document, then it could be assumed that the term '*kid*' is also related to the specific document, or even terms like '*boy*', '*girl*', '*toy*'. The resulting representation model is also a VSM, but the document vectors become less sparse and the independence of features is mitigated in an indirect way. The smoothing is usually achieved by a linear mapping of data vectors to a new feature space using a semantic matrix S. It is convenient to think that the new document vector $d'=Sd$ contains the dot product similarities between the original BOW vector d and the rows of the semantic matrix S.

A basic difference between the various semantic smoothing methods is related to the dimension of the new feature space which is determined by the number V' of row vectors of matrix S. In case their number is less than the size V of the vocabulary, such vectors are called as *concept vectors* and are usually produced using the LSI method. Each concept vector has a distribution of weights associated to the V original terms that define their contribution of to the corresponding *concept*. Of course the resulting representation of the smoothed vector d' is less interpretable than the original and there is always a problem of determining the proper number of concept vectors.

An alternative approach for semantic smoothing assumes that each row vector of matrix S is associated with one vocabulary term. Unlike a concept vector that describes abstract semantics of higher level, here, the elements of each vector describe the relation of this term to the other terms. Those relations constitute the so called *term context*, thus the respective vector is called *term context vector*. Each element of the mapped vector d' will contain the dot product similarity between document d and the corresponding term context vector, i.e. for each term v_i the element d'_i provides the degree to which the original document d contains the term v_i and its context, instead of just computing its frequency as happens in the BOW representation. Note also that in BOW representation, a dot product would give zero similarity for two documents that do not have common terms. On the contrary, the dot product between a document vector and a term context vector of a term v_i that does not appear in that document may give a non-zero similarity. This happens if the document contains at least one term v_j with non-zero weight in the context of term v_i. For this reason, the smoothed representation d' is usually less sparse that d and retains their interpretability of dimensions. Moreover, concept-based methods may be applied on the new representations.

The motivation of our work is to establish the importance of *term context vectors* and to define an efficient way to compute them. The CVM-VSM method considers that the term context is computed based on term co-occurrence frequency at the document-level. It does not take into account the sequential nature of text and thus ignore the local distance of terms when computing term context. On the other hand, the GTCV-VSM proposed in this work extends the previous approach by considering term context at three levels: i) It uses the notion of *local term context vector* (*ltcv*) to model the context around the location in the text sequence where a term appears. These vectors are computed using a local smoothing kernel as suggested in the *LoWBOW* approach [18] which is described in the next section. The kernel *takes into account the distance* in which other terms appear around the sequence location under consideration. ii) It computes the *document term context vector* (*dtcv*) for each term that summarizes the term context at the document-level and iii) it computes the final *global term context vector* (*gtcv*) for each term representing the overall term context at corpus-level. The *gtcv* vectors constitute the rows of the semantic matrix S. Thus the intuition behind GTCV-VSM approach is to capture the local term context from term sequences and then to construct a representation for global term context by averaging *ltcvs* at the document and corpus-level.

3 Utilizing Local Contextual Information

A text document can be considered as a finite term sequence of its T consecutive terms denoted as $d^{seq}{=}\langle d^{seq}(1), \ldots, d^{seq}(T)\rangle$ but, except for Bag of Phrases, so far in this paper the previously mentioned VSM-extensions ignore this property. A category of methods have been proposed aiming to capture local information directly from the term sequence of a document. The representation proposed in [27], first considers a segmentation of the sequence that is done by dragging a window of n terms along the sequence and computing the local BOW vectors for each of the overlapping segments. All these local BOW vectors constitute the document representation called *Local Word Bag* (*LWB*). To compute the similarity between a pair of documents, the authors introduce a variant of the *VG-Pyramid*

Matching Kernel [12] that maps the two sets of local BOW vectors to a multi-resolution histogram and computes a weighted histogram intersection.

Another approach for text representation presented in [18], is the *Locally Weighted Bag of Words* (*LoWBOW*) that preserves local contextual information of text documents by the effective modeling of the text sequential structure. At first, a number of L equally distant locations are defined in the term sequence. Each sequence location ℓ_i, i=1, …, L, is then associated with a local histogram which is a point in the multinomial simplex $\mathbb{P}_{V-1}$, where V is the number of vocabulary terms. More specifically, for $(V-1)\geq 0$, the $\mathbb{P}_{V-1}$ space is the $(V-1)$-dimensional subset of $\mathbb{R}^V$ that contains all probability vectors (histograms) over V objects (for a discussion on the multinomial simplex see the Appendix of [18])

$$\mathbb{P}_{V-1} = \left\{ H \in \mathbb{R}^V : \ H_i \geq 0, \ \forall i = 1, \ldots, V \text{ and } \sum_{i=1}^{V} H_i = 1 \right\}. \tag{7}$$

Contrary to LWB, in LoWBOW the local histogram is computed using a *smoothing kernel* to weight the contribution of terms appearing around the referenced location in the term sequence, and to assign more importance to closely neighboring terms. Denoting as $H_{\delta(d^{seq}(t))}$ the *trivial term histogram* of V terms whose probability mass is concentrated only at the term that occurs at the location t in d^{seq}

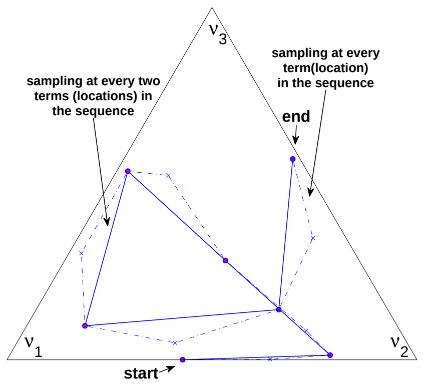
$$\left[H_{\delta(d^{seq}(t))}\right]_i = \begin{cases} 1, & \nu_i = d^{seq}(t) \\ 0, & \nu_i \neq d^{seq}(t) \end{cases}, \ i = 1, \ldots, V, \tag{8}$$

then the locally smoothed histogram at a location ℓ in the d^{seq} term sequence is computed as in [18]

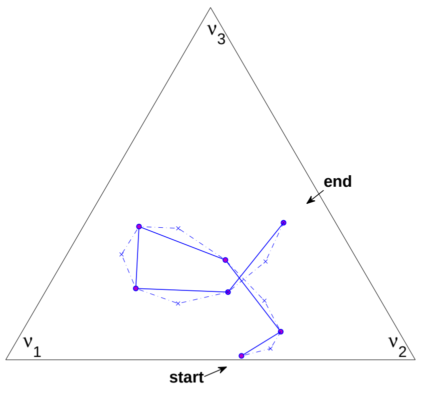
$$lowbow(d^{seq}, \ell) = \sum_{t=1}^{T} H_{\delta(d^{seq}(t))} K_{\ell,\sigma}(t), \tag{9}$$

where T is the length of d^{seq}. $K_{\ell,\sigma}(t)$ denotes the weight for location t in sequence given by a discrete Gaussian weighting kernel function of mean value ℓ and standard deviation σ. Specifically, the weighting function is a Gaussian probability density function restricted in $[1, T]$ and renormalized so that $\sum_{t=1}^{T} K_{\ell,\sigma}(t) = 1$. It is easy to verify that the result of the histogram smoothing of Eq. 9 is also a histogram.

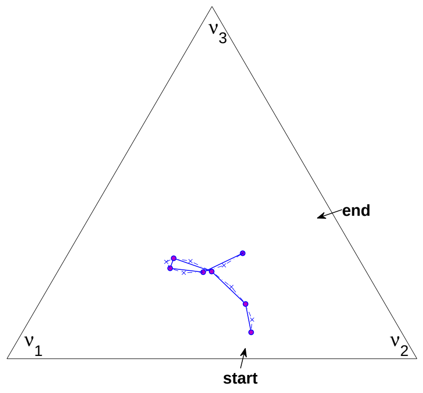
It must be noted that for σ=0 the *lowbow* histogram (Eq. 9) coincides with the trivial histogram $H_{\delta(d^{seq}(\ell))}$, where all the probability mass is concentrated at the term at location ℓ. As σ grows, part of the probability mass is transfered to the terms occurring near location ℓ. In this way, the *lowbow* histogram at location ℓ is enriched with information about the terms occurring in the neighborhood of ℓ. The smoothing parameter σ adjusts the 'locality' of term semantics that is taken into account by the model. Thus, instead of mining unordered local vectors as in [27], the LoWBOW approach embeds the term sequence of a document in the $\mathbb{P}_{V-1}$ simplex. The sequence of the L locally smoothed histograms (denoted as *lowbow histograms*) form a curve in the (V-1)-dimensional simplex (denoted as *LoWBOW curve*). Fig. 1 illustrates the LoWBOW curves generated for a toy example and describes the role of parameter σ. In this figure we aim to illustrate i) the LoWBOW curve representation, i.e. the curve that corresponds to a sequence



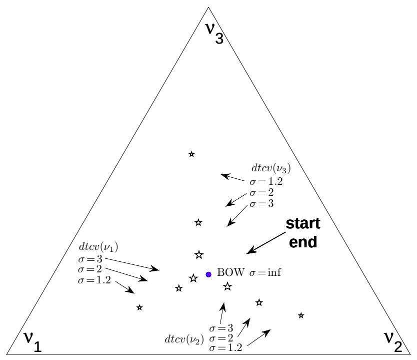
(a) $\sigma = 1.2$



(b) $\sigma = 2$



(c) $\sigma = 3$



(d)

Fig. 1 A toy example where the sequence $\langle \nu_1,\ \nu_2,\ \nu_2,\ \nu_2,\ \nu_1,\ \nu_3,\ \nu_3,\ \nu_1\ \nu_1,\ \nu_1,\ \nu_2,\ \nu_2,\ \nu_3 \rangle$ is considered that uses three different terms ν_1, ν_2, ν_3 (vocabulary size: V=3). The subfigures present LoWBOW curves in the ($V-1$)-dimensional simplex for increasing values of the parameter σ that induce more smoothing to the curve. Each point of the curve corresponds to a local histogram computed at a sequence location. The more a term affects the local context at a location in the sequence, the more the curve point (the *lowbow* histogram related to that location) moves towards the respective corner of the simplex. For σ=0 local histograms correspond to simplex corners, thus the curve moves from corner to corner of the simplex. Two different sampling rates for LoWBOW representation are illustrated: sampling at every term location in the sequence (dashed line) which is the our strategy to collect contextual information for each term, and sampling every two terms (solid line). d) For σ=∞, the LoWBOW curve reduces to a single point that coincides with the BOW histogram of the sequence. In (d) we present as 'stars' the average *ltcv* histograms for each term (*dtcv* histograms) for the three different values of σ and α=0.6 for all terms. As the value of σ increases, the *dtcv* histograms of all terms become more similar tending to coincide with the BOW representation.

of histograms (local context vectors), where each local context vector is computed at a specific location of the sequence and corresponds to a point in the (V-1)-dimensional simplex; ii) the impact of the smoothing coefficient σ on the computed local context vectors. This figure illustrates that the increase of smoothing makes the *lowbow* histograms (points of the curve) more similar. This can also be verified by observing that as smoothing increases, the curve becomes more concentrated around a central location of the simplex. For σ=∞ all histograms become similar to the BOW representation and the curve reduces to a single point. On the contrary, for σ=0 the histograms correspond to simplex corners.

A similarity measure between LoWBOW curves has been proposed in [18] that assumes a sequential correspondence between two documents and computes the sum of the similarities between the L pairs of LoWBOW histograms. Obviously, it is expected for this similarity measure to underestimate the thematic similarity between documents that follow different order in the presentation of similar semantic content.

4 A Semantic Matrix based on Global Term Context Vectors

In this section we present the Global Term Context Vector-VSM (GTCV-VSM) approach for capturing the semantics of the original term feature space of a document collection. The method computes the contextual information of each vocabulary term, that is subsequently utilized in order to create a semantic matrix. In analogy with CVM-VSM, our approach reduces data sparsity but not dimensionality. The interpretability of the derived vector dimensions remains as strong as in the BOW model as the value of each dimension of the mapped vector corresponds to one vocabulary term. Methods that reduce data dimensionality could also be applied on the new representations at a subsequent phase. Compared to CVM-VSM, GTCV-VSM generalizes the way the term context is computed by taking into account the distance between terms in the term sequence of each document. This is achieved by exploiting the idea of LoWBOW to describe the local contextual information at a certain location in a term sequence. It must be noted that our method borrows from the LoWBOW approach only way the local histogram is computed at each location of the term sequence and does not make use of the LoWBOW curve representation.

More specifically, we define the *local term context vector* (*ltcv*) as a histogram associated with the exact occurrence of term $d^{seq}(\ell)$ at location ℓ in a sequence d^{seq}. Hence, one *ltcv* vector is computed at every location in the term sequence, i.e. ℓ=1, . . . , T. Note that GTCV-VSM does not preserve any curve representation. This means that we are not interested in the temporal order of the local term context vectors. The $ltcv(d^{seq}, \ell)$ is a modified $lowbow(d^{seq}, \ell)$ probability vector that represents contextual information around location ℓ, while adjusting explicitly the *self-weight* $\alpha_{d^{seq}(\ell)}$ of the reference term appearing at location ℓ

$$[ltcv(d^{seq}, \ell)]_i = \begin{cases} \alpha_{d^{seq}(\ell)} & , \nu_i = d^{seq}(\ell), \\ (1-\alpha_{d^{seq}(\ell)}) \cdot \frac{idf_i \, \cdot \, [lowbow(d^{seq}, \ell)]_i}{\sum_{j=1, j\neq i}^{V} idf_j \, \cdot \, [lowbow(d^{seq}, \ell)]_j} & , \nu_i \neq d^{seq}(\ell). \end{cases} \tag{10}$$

The self-weight ($0 \leq \alpha_{d^{seq}(\ell)} \leq 1$) adjusts the relative importance between contextual information (computed using the *lowbow* histogram) and the self-representation of each term. Fig. 2 illustrates an example of how the value of parameter α affects the local term weighting around a reference term in a sequence. When the parameter σ of the Gaussian smoothing kernel is set to zero, or α=1, the $ltcv(d^{seq}, \ell)$ reduces to a trivial histogram $H_{\delta(d^{(seq)}(\ell))}$ (see Eq. 8). The other extreme is the infinite σ value, where for small α values all the *ltcv* computed in a document d become similar to the tf histogram for that document.

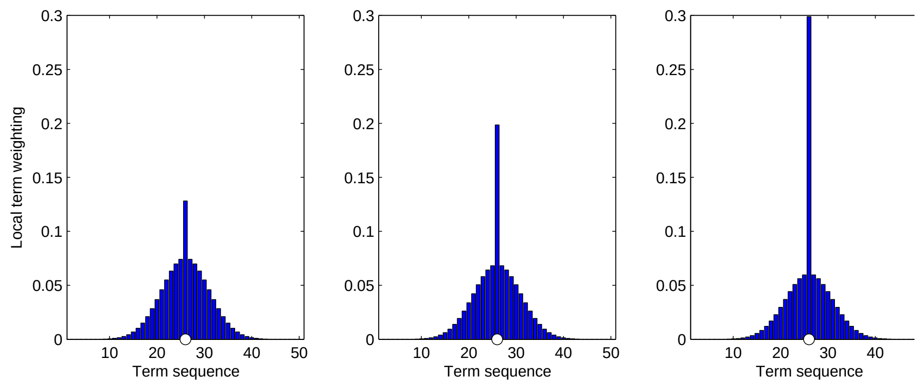


Fig. 2 Various weight distributions for the neighboring terms around a reference term occurring in the middle of a term sequence of length 50. The distributions are obtained by varying the value of parameter α in Eq. 10. This distribution defines the contribution of each term to the context of the specific reference term. The scale value of the local kernel is set to σ=5, while self-weight α is set to 0.05 (left), 0.10 (middle), and 0.2 (right).

The latter observation is the reason for considering an explicit self-weight in Eq. 10, because a flat smoothing kernel obtained for large σ value can make a *lowbow* vector to have improperly low self-weight for the reference term. For example, if a term appears once in a document, then the *lowbow* vector with $\sigma=\infty$ at that location would contain very low weight for that term. Generally, the value of α_ν determines how much the context vector of term ν should be dominated by the self-weight of term ν. In our method we set this parameter independently for each individual term as a function of its idf_ν component

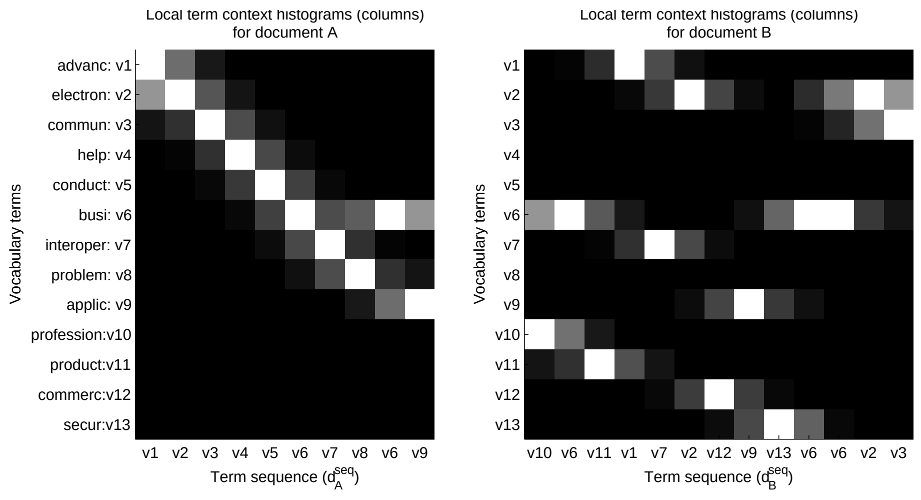
$$\alpha_\nu = \lambda + (1-\lambda)\cdot\left(1-\frac{idf_\nu}{logN}\right), \quad \lambda \in [0,1], \tag{11}$$

where λ is a lower bound for all a_ν, ν=1, ..., V (in our experiments we used λ=0.2). The rationale for the above equation is that for terms with high document frequency (i.e. low idf_ν), we assign high α_ν values that suppress the local context in the respective context vectors. In other words, the context is considered more important for terms that occur in fewer documents. In Fig. 3a, we present an example illustrating the *ltcv* vectors of two term sequences presented in Fig. 3c.

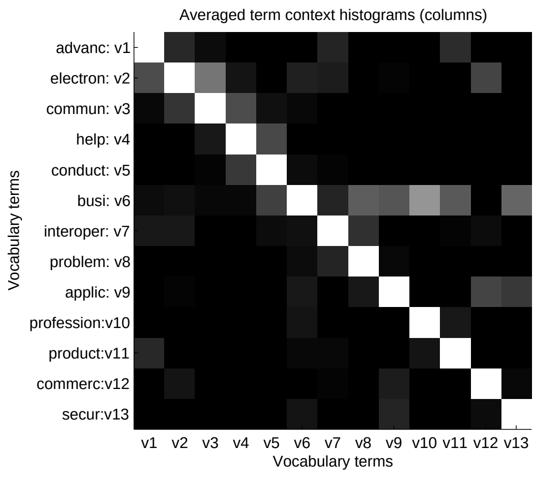
We further define the *document term context vector* (*dtcv*) as a probability vector that summarizes the context of a specific term at the document-level by averaging the *ltcv* histograms corresponding to the occurrences of this term in the document. More specifically, suppose that a term ν appears $no_{i,\nu} > 0$ times in the term sequence d_i^{seq} (i.e. in the i-th document) which is of length T_i. Then the *dtcv* of this term ν for document i is computed as:

$$dtcv(d_i^{seq},\, \nu) = \frac{1}{no_{\nu,i}} \sum_{j=1}^{no_{i,\nu}} ltcv(d_i^{seq},\, \ell_{i,\nu}(j)), \tag{12}$$

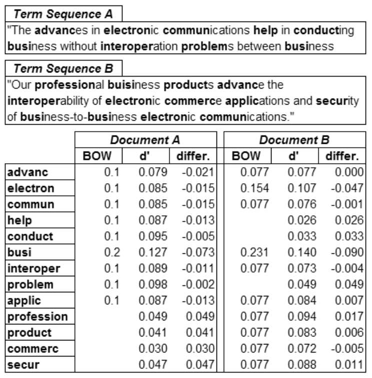
where $\ell_{i,\nu}(j)$ is an integer value in $[1,\ldots,T_i]$ denoting the location of the j-th occurrence of ν in d_i^{seq}.



(a)



(b)



Term Sequence A

"The **advanc**es in **electron**ic **commun**ications **help** in **conduct**ing **busi**ness without **interoper**ation **problem**s between **busi**ness

Term Sequence B

"Our **profession**al **buisi**ness **product**s **advanc**e the **interoper**ability of **electron**ic **commerc**e **applic**ations and **secur**ity of **busi**ness-to-**busi**ness **electron**ic **commun**ications."

	Document A			***Document B***		
	BOW	**d'**	**differ.**	**BOW**	**d'**	**differ.**
advanc	0.1	0.079	-0.021	0.077	0.077	0.000
electron	0.1	0.085	-0.015	0.154	0.107	-0.047
commun	0.1	0.085	-0.015	0.077	0.076	-0.001
help	0.1	0.087	-0.013		0.026	0.026
conduct	0.1	0.095	-0.005		0.033	0.033
busi	0.2	0.127	-0.073	0.231	0.140	-0.090
interoper	0.1	0.089	-0.011	0.077	0.073	-0.004
problem	0.1	0.098	-0.002		0.049	0.049
applic	0.1	0.087	-0.013	0.077	0.084	0.007
profession		0.049	0.049	0.077	0.094	0.017
product		0.041	0.041	0.077	0.083	0.006
commerc		0.030	0.030	0.077	0.072	-0.005
secur		0.047	0.047	0.077	0.088	0.011

(c)

Fig. 3 An example of how *ltcv* histograms are used to summarize the overall context in which a term appears in the two term sequences of (c) using Eq. 14. a) The term sequences (x-axis) of documents A, B are presented and the corresponding local term context vectors are illustrated as grey-scaled *columns*. Those vectors are computed at every location in the sequence using a Gaussian smoothing kernel with σ=1 and α=0.6 for all terms. Brighter intensity at cell i, j indicates higher contribution of the term ν_i to the local context of the term appearing at location j in the sequence. b) The resulting transposed semantic matrix ($S^\top$), where the grey-scaled columns illustrate the global contextual information for each vocabulary term computed by averaging the respective local context histograms (Eq. 13). c) The two initial term sequences (the stem of each non-trivial term is emphasized). Assuming the same *idf* weight for each vocabulary term, the table presents the BOW vector, the transformed vector d' using Eq. 14 as well as the effect of semantic smoothing ($diff = BOW - d'$) on document vectors. The redistribution of term weights that results by the proposed mapping reveals is done in such a way that low frequency terms are gaining weight against the more frequent ones. Note also that the similarity between the two documents is 0.756 for the BOW model and 0.896 for the GTCV-VSM.

Next, the *global term context vector* (*gtcv*), is defined for a vocabulary term ν so as to represent the overall contextual information for all appearances of ν in the corpus of all N term sequences (documents).

$$gtcv(\nu) = h_{gtcv(\nu)} \left(\sum_{i=1}^{N} tf_{i,v} \; dtcv(d_i^{seq},\, \nu) \right). \tag{13}$$

The coefficient $h_{gtcv(\nu)}$ normalizes the vector $gtcv(\nu)$ with respect to the Euclidean norm, and $tf_{i,\nu}$ is the frequency of the term ν in the i-th document. Thus, the $gtcv(\nu)$ of term ν is computed using a weighted average of the document context vectors $dtcv(d_i^{seq},\, \nu)$ obtained for each document i in which term ν appears. Thus, in contrast to LoWBOW curve approach which focuses on the sequence of local histograms that describe the writing structure of a document, our method focuses on the extraction of the global semantic context of a term by averaging the local contextual information at all the corpus locations where this term appears.

Finally, the extracted global contextual information is used to construct the $V \times V$ semantic matrix S_{gtcv} where each row ν is the $gtcv(\nu)$ vector of the corresponding vocabulary term ν. Fig. 1d provides an example of illustrating the $dtcv(d_i^{seq},\, \nu)$ vectors for each document (the points denoted as 'stars'). Fig. 3b illustrates the final *gtcv* vectors obtained by averaging the document-level contexts for each vocabulary term.

To map a document using the proposed Global Term Context Vector-VSM approach, we compute the vector d' where each element ν is Cosine similarity between the BOW representation d of the document and the global term context vector $gtcv(\nu)$:

$$\Phi_{gtcv} : d \rightarrow d' = S_{gtcv}\, d, \;\; d' \in \mathbb{R}^V. \tag{14}$$

Note that the transformed document vector d' is V-dimensional that retains the interpretability, since each dimension still corresponds to a unique vocabulary term. Moreover, if $\sigma{=}0$ and $\alpha{>}0$, then $S_{gtcv}\, d{=}d$. Looking at Eq. 4, the product $S_{gtcv}^{\top} S_{gtcv}$ essentially computes a Term Similarity Matrix where the similarity between two terms is based on the distribution of term weights in their respective global term context vectors, i.e., on the similarity of their global context histograms. The table of Fig. 3c illustrates the effect of redistribution (compared to BOW) of the term weights (semantic smoothing) in the transformed document vectors achieved by the proposed mapping.

The procedure of representing the input documents using GTCV-VSM takes place in the preprocessing phase. Let T_i the length of the i-th document and V_i its vocabulary. Let also V the size of the whole corpus vocabulary. Then the cost to compute one *ltcv* vector at a location of the term sequence using Eq. 10, and to add its V_i non-zero dimensions to the respective *dtcv*, is $\mathrm{O}(T_i{+}V_i)$. This is done T_i times and the final *dtcv* of each different term of the document is added to the respective the *gtcv* rows. Thus, using proper notation for the average length $\overline{T_i}$ and vocabulary size $\overline{V_i}$ of the documents in a corpus, the cost of constructing the semantic matrix can be expressed as $\mathrm{O}(N{\cdot}\overline{T_i}{\cdot}(\overline{T_i}{+}2{\cdot}\overline{V_i}))$. However, since $\overline{V_i}{\leq}\overline{T_i}{\ll}V$, the overall computational cost of the GTCV-VSM is determined by the $\mathrm{O}(N{\cdot}V^2)$ cost of the matrix multiplication of the mapping of Eq. 14.

Table 1 Characteristics of text document collections. N denotes the number of documents, V is the size of the global vocabulary and $\overline{V_i}$ the average document vocabulary, *Balance* is the ratio of the smallest to the largest class and $\overline{T_i}$ is the average length of the term sequences of documents.

Name	**Topics**	**Classes**	N	$Balance$	V	$\overline{V_i}$	$\overline{T_i}$
D$_1$	20–NGs: graphics, windows.x, motor, baseball, space, mideast	6	2000	200/400	4343	48.8	110
D$_2$	20–NGs: atheism, autos, baseball, electronics, med, mac, motor, politics.misc	7	3500	500/500	6442	52.6	108
D$_3$	20–NGs: atheism, christian, guns, mideast	4	1600	400/400	4080	62	131
D$_4$	20–NGs: forsale, autos, baseball, motor, hockey	5	1250	250/250	4762	44.1	104
D$_5$	Reuters–21578: acq, corn, crude, earn, grain, interest, money-fx, ship, trade, wheat	10	9979	237/3964	5613	39.1	76

5 Clustering Experiments

Our experimental setup was based on five different datasets: D_1-D_4 are subsets of the 20-Newsgroups[1], while D_5 is the Mod Apte split [2] version of the Reuters-21578[2] benchmark document collection where the 10 classes with larger number of training examples are kept. The characteristics of these datasets are presented in Table 1. The preprocessing of datasets included the removal of all tags, headers and metadata from the documents, while applied word stemming and discarded terms appearing in less than five documents. It is worth mentioning how we preprocessed the term sequences of documents. We considered a *dummy term* that replaced in the sequences all the low-frequency terms that were discarded so as to maintain the relative distance between the terms that remained in each sequence. For similar reasons, two dummy terms were considered at the end of every sentence denoted by characters as (e.g. '.', '?', '!'). The dummy term is ignored when constructing the final data vectors.

For each dataset, we have considered several data mappings Φ and after each mapping the *spherical k-means* (*spk-means*) [8] and spectral clustering (*spectral-c*) [24] algorithms were applied to cluster the mapped documents vectors into the k predefined number of clusters corresponding to the different topics (classes) in a collection. In contrast to *k-means* that is based on the Euclidean distance [21], spk-means uses the Cosine similarity and maximizes the *Cohesion* of the clusters $C=\{c_1,\dots,c_k\}$

$$Cohesion(C) = \sum_{j=1}^{k} \sum_{d_i \in c_j} u_j^\top d_i, \tag{15}$$

where u_j is the normalized centroid of cluster c_j with respect to the Euclidean norm.

Spectral clustering projects the document vectors in a subspace which is spanned by the k largest eigenvectors of the Laplacian matrix L computed from the similarity matrix $A^{(N\times N)}$ of pairwise Cosine similarities between documents. More specifically, the Laplacian matrix is computed as $L=D^{-1/2}AD^{-1/2}$, where D is a

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.tar.gz.

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/reuters21578.tar.gz

Table 2 NMI values of the clustering solution for VSM (BOW), GVSM, CVM-VSM and the proposed GTCV-VSM (for several values of σ) document representations using the spk-means algorithm.

		D$_1$			**D$_2$**			**D$_3$**			**D$_4$**			**D$_5$**		
Method	σ	***avg***	***best***	***avg$_{10\%}$***	***avg***	***best***	***avg$_{10\%}$***	***avg***	***best***	***avg$_{10\%}$***	***avg***	***best***	***avg$_{10\%}$***	***avg***	***best***	***avg$_{10\%}$***
BOW	–	.722	.821	.594	.748	.829	.638	.537	.548	.379	.625	.779	.505	.552	.562	.535
GTCV	1	.749	.854	.601	.767	.845	.638	.544	.564	.372	.667	.793	.515	.570	.578	.561
	2	.756	.871	.631	.765	.852	.657	.563	.574	.396	.670	.832	.539	.572	.580	.561
	5	.773	.881	.687	.777	.864	.662	.577	.602	.400	**.688**	**.851**	.539	.589	**.633**	.578
	10	**.777**	**.886**	.685	**.781**	**.873**	.672	**.590**	**.621**	.424	.684	.849	.540	**.590**	.630	.580
	30	.761	.879	.659	.776	.863	.653	.579	.590	.369	.683	.842	.518	.576	.612	.568
	inf	.760	.862	.631	.772	.862	.639	.574	.586	.366	.681	.840	.521	.576	.610	.566
GVSM	–	.752	.832	.611	.747	.822	.637	.556	.576	.419	.670	.827	.547	.575	.580	.573
CVM	–	.750	.841	.612	.754	.851	.659	.547	.604	.400	.672	.824	.541	.578	.581	.575

diagonal matrix. Each diagonal element contains the sum of the i-th row of similarities $D_{ii}=\sum_{j=1}^{N}A_{ij}$. The next step is the construction of a matrix $X^{(N\times k)}=\{x_i : i=1,\dots,k\}$ whose columns correspond to the k largest eigenvectors of L. The standard k-means algorithm is then used to cluster the rows of matrix X after being normalized to unit length in Euclidean space, where the i-th row is the vector representation of the i-th document in the new feature space.

Clustering evaluation was based on the supervised measure *Normalized Mutual Information* (*NMI*) and the F_1*-measure*. We denote as n_i^{gt} the number of documents of class i, n_j the size of cluster j, n_{ij} the number of documents belonging to class i that are clustered in cluster j, C^{gt} the grouping based on ground truth labels of documents $c_1^{gt},\dots,c_k^{gt}$ (true classes). Let us further denote $p(c_i^{gt})=n_i^{gt}/N$ and $p(c_j)=n_j/N$ the probability of selecting arbitrarily a document from the dataset and that belongs to class c_i^{gt} and cluster c_j, respectively, and $p(c_i^{gt},c_j)=n_{ij}/N$ the joint of arbitrarily selecting a document from dataset and that belongs to cluster c_j and is of class c_i^{gt}. Then, the [0,1]–Normalized MI measure is computed by dividing the Mutual Information by the maximum between the cluster and class entropy:

$$NMI(C^{gt},C)=\left(\sum_{\substack{c_i^{gt}\in C^{gt}\\ c_j\in C}} p(c_i^{gt},c_j)\log\frac{p(c_i^{gt},c_j)}{p(c_i^{gt})p(c_j)}\right)\Big/\max\{H(C^{gt}),\,H(C)\}. \tag{16}$$

When C and C^{gt} are independent, the value of NMI equals to zero, while it equals to one if these partitions contain identical clusters.

The F_1*-measure* is the harmonic mean of the *precision* and *recall* measures of the clustering solution:

$$F_1=\frac{precision\cdot recall}{precision+recall}. \tag{17}$$

Higher values of F_1 in [0,1] indicate better clustering solutions.

Tables 2, 3, 5, and 6 present the results from the experiments conducted for each collection. Specifically, we compared the classic BOW representation, the GVSM, the proposed GTCV-VSM method (with λ=0.2 in Eq. 11), that represents the documents as described in Eq. 14 and the CVM-VSM as proposed in [5], where document vectors are computed based on Eq. 6 with *idf* weights. More specifically,

Table 3 F_1-measure values of the spk-means clustering solution for the different representation methods.

Method	σ	D1 *avg*	D1 *best*	D1 $avg_{10\%}$	D2 *avg*	D2 *best*	D2 $avg_{10\%}$	D3 *avg*	D3 *best*	D3 $avg_{10\%}$	D4 *avg*	D4 *best*	D4 $avg_{10\%}$	D5 *avg*	D5 *best*	D5 $avg_{10\%}$
BOW	–	.779	.920	.685	.780	.901	.645	.703	.706	.570	.735	.918	.558	.675	.697	.646
GTCV	1	.806	.940	.688	.790	.921	.650	.709	.713	.576	.755	.920	.561	.691	.695	.677
	2	.814	.946	.688	.792	.924	.674	.721	.728	.580	.764	.938	.598	.698	.714	.672
	5	.828	.953	.722	.817	.929	.665	.736	.737	.597	**.773**	**.948**	.611	**.712**	**.751**	.681
	10	**.832**	**.954**	.733	**.820**	**.936**	.603	**.737**	**.739**	.603	.773	.947	.581	.712	.749	.681
	30	.814	.950	.747	.794	.929	.657	.725	.727	.576	.766	.944	.579	.698	.746	.666
	inf	.813	.942	.689	.792	.926	.651	.722	.728	.576	.765	.944	.581	.698	.744	.666
GVSM	–	.790	.923	.705	.783	.903	.640	.706	.71	.576	.750	.943	.591	.687	.720	.672
CVM	–	.765	.941	.672	.790	.930	.672	.708	.725	.576	.751	.934	.604	.685	.716	.669

Table 4 The p and t values of the statistical significance t-test of the difference in k-means performance using GTCV-VSM (σ=10) and the compared representation methods, with respect to the two evaluation measures. Values of p smaller than the significance level of 0.05 (5%) indicate significant superiority of GTCV-VSM.

GTCV (σ=10) **vs**	D1 *p-val*	D1 *t-val*	D2 *p-val*	D2 *t-val*	D3 *p-val*	D3 *t-val*	D4 *p-val*	D4 *t-val*	D5 *p-val*	D5 *t-val*
BOW_{NMI}	$.011 \cdot 10^{-6}$	5.98	$.075 \cdot 10^{-3}$	4.05	$.025 \cdot 10^{-6}$	5.81	$.080 \cdot 10^{-8}$	6.45	.0000	12.8
$GVSM_{NMI}$	.0008	2.68	$.081 \cdot 10^{-3}$	4.02	$.050 \cdot 10^{-3}$	4.15	.085	1.73	$.056 \cdot 10^{-5}$	5.17
CVM_{NMI}	.0051	2.83	.0010	3.33	$.052 \cdot 10^{-4}$	4.65	.1659	1.39	$.077 \cdot 10^{-3}$	4.04
BOW_{F_1}	$.020 \cdot 10^{-5}$	5.39	$.050 \cdot 10^{-2}$	3.54	$.046 \cdot 10^{-2}$	3.56	.0010	3.32	.0000	12.8
$GVSM_{F_1}$	$.037 \cdot 10^{-3}$	4.22	.0021	3.11	$.067 \cdot 10^{-2}$	3.45	.0329	2.15	.0000	9.06
CVM_{F_1}	$.081 \cdot 10^{-3}$	4.02	$.06 \cdot 10^{-8}$	6.50	.0027	3.04	.0314	2.18	.0000	9.31

Table 5 NMI values of the clustering solution for VSM (BOW), GVSM, CVM-VSM and the proposed GTCV-VSM (for several values of σ) document representations using the spectral clustering algorithm.

Method	σ	D1 *avg*	D1 *best*	D1 $avg_{10\%}$	D2 *avg*	D2 *best*	D2 $avg_{10\%}$	D3 *avg*	D3 *best*	D3 $avg_{10\%}$	D4 *avg*	D4 *best*	D4 $avg_{10\%}$	D5 *avg*	D5 *best*	D5 $avg_{10\%}$
BOW	–	.753	.761	.750	.781	.788	.737	.569	.585	.555	.718	.780	.631	.558	.559	.506
GTCV	1	.770	.774	.769	.790	.795	.750	.614	.626	.600	.735	.779	.642	.560	.561	.516
	2	.781	.785	.760	.790	.794	.757	.625	.632	.601	.752	.789	.649	.562	.564	.523
	5	.794	.804	.790	**.833**	**.853**	.763	.639	.640	.619	**.768**	**.827**	.669	.579	**.600**	.557
	10	**.807**	**.814**	.801	.833	.853	.761	**.645**	**.648**	.620	.758	.819	.661	**.581**	.589	.558
	30	.791	.796	.769	.807	.832	.743	.613	.613	.609	.755	.797	.647	.567	.582	.535
	inf	.774	.782	.767	.794	.794	.722	.619	.619	.610	.749	.793	.637	.560	.568	.530
GVSM	–	.756	.770	.702	.794	.830	.747	.593	.595	.586	.722	.780	.637	.548	.554	.513
CVM	–	.761	.768	.751	.801	.823	.760	.605	.606	.590	.728	.794	.642	.557	.566	.519

for each collection, each representation method was tested for 100 runs of spk-means (Tables 2, 3) and spectral-c (Tables 5, 6). To provide fair comparative results, for each document collection all methods were initialized using the same random document seeds. The average of all runs (*avg*), the average of the worst 10% of the clustering solutions ($avg_{10\%}$), and the best values are reported for each performance measure. The worst 10% concerns the 10% of the solutions with the lowest *Cohesion*, while the best clustering solution is that having the maximum *Cohesion* in the 100 runs (for spectral-c the sum of squared distances is considered for this purpose). Moreover, in Fig. 4 we present the average clustering performance of spk-means with respect to the value of λ parameter of Eq. 11 where, although

Table 6 F_1-measure values of the spectral clustering solution for the different representation methods.

Method	σ	**D$_1$**			**D$_2$**			**D$_3$**			**D$_4$**			**D$_5$**		
		avg	*best*	*avg$_{10\%}$*	*avg*	*best*	*avg$_{10\%}$*	*avg*	*best*	*avg$_{10\%}$*	*avg*	*best*	*avg$_{10\%}$*	*avg*	*best*	*avg$_{10\%}$*
BOW	–	.801	.811	.780	.819	.822	.767	.710	.723	.701	.808	.911	.697	.666	.669	.654
GTCV	1	.811	.819	.809	.822	.832	.772	.729	.741	.728	.834	.915	.722	.694	.703	.663
	2	.818	.823	.806	.837	.841	.779	.733	.746	.732	.865	.922	.725	.689	.703	.652
	5	.837	.840	.818	.887	**.927**	.792	.744	.756	.737	**870**	**.930**	.740	**.716**	**.727**	.647
	10	**.840**	**.842**	.826	**.890**	.925	.788	**.754**	**.759**	.742	.865	.929	.736	.710	.725	.654
	30	.823	.826	.809	.856	.886	.769	.726	.735	.725	.864	.925	.705	.704	.701	.642
	inf	.814	.817	.806	.826	.832	.734	.728	.735	.729	.859	.922	.703	.692	.686	.653
GVSM	–	.756	.770	.702	.826	.901	.780	.709	.714	.724	.823	.916	.705	.642	.657	.654
CVM	–	.761	.768	.779	.831	.897	.791	.725	.725	.723	.825	.916	.713	.673	.678	.654

Table 7 The p and t values of the statistical significance t-test of the difference in spectral clustering performance using GTCV-VSM (σ=10) and the compared representation methods, with respect to the two evaluation measures. Values of p smaller than the significance level of 0.05 (5%) indicate significant superiority of GTCV-VSM.

GTCV (σ=10) vs	**D$_1$**		**D$_2$**		**D$_3$**		**D$_4$**		**D$_5$**	
	p-val	*t-val*	*p-val*	*t-val*	*p-val*	*t-val*	*p-val*	*t-val*	*p-val*	*t-val*
BOW$_{NMI}$	.0000	27.3	.0000	13.8	.0000	620.	$.026\cdot10^{-4}$	4.85	.0000	8.03
GVSM$_{NMI}$	.0000	16.7	.0000	7.51	.0000	130.	$.129\cdot10^{-5}$	4.99	.0000	12.1
CVM$_{NMI}$	.0000	19.3	$.150\cdot10^{-8}$	6.35	.0000	138.	$.316\cdot10^{-3}$	3.67	.0000	8.83
BOW$_{F_1}$	.0000	24.1	.0000	11.4	.0000	875.	$.123\cdot10^{-4}$	4.48	.0000	19.1
GVSM$_{F_1}$	.0000	15.1	.0000	7.53	.0000	410.	$.113\cdot10^{-2}$	3.31	.0000	30.7
CVM$_{F_1}$	.0000	18.7	.0000	7.11	.0000	268.	$.115\cdot10^{-3}$	3.94	.0000	14.1

not best for all cases, the value 0.2 we used seems to be a reasonable choice for all the datasets we have considered. Note that similar effect was observed for spectral-c method.

In order to illustrate the statistical significance of the obtained results, the well-known *t-test* was applied for each dataset to determine the significance of the performance difference between our methods and the compared methods. We have considered the case where σ=10 for the Gaussian kernel for all datasets. Within a confidence interval of 95% and for the value of degrees of freedom equal to 198 (for two sets of 100 experiments each), the critical value for t is t_c=1.972 (p_c=5% for p value). This means that if the computed $t \geq t_c$, then the null hypothesis is rejected ($p \geq 5\%$, respectively), i.e. our method is superior, otherwise the *null hypothesis* is accepted. As it can be observed from the results of the statistical tests for spk-means presented in Table 4, the performance superiority of GTCV-VSM is clearly significant in four out of five datasets with respect to all other methods. For dataset D_4 the tests indicate that GTCV-VSM, although still better than BOW, has less significant difference in performance compared to GVSM and CVM-VSM. Table 4 provides the respective t-test results for the spectral-c method where, also due to the lower standard deviation of the results using all document representation methods, the GTCV-VSM demonstrates significantly better results than the compared representations.

The experimental results indicate that our method outperforms the traditional BOW approach in all cases, even for small values of smoothing parameter σ (e.g. σ=1 or 2). This substantiates our rationale that the clustering procedure is assisted

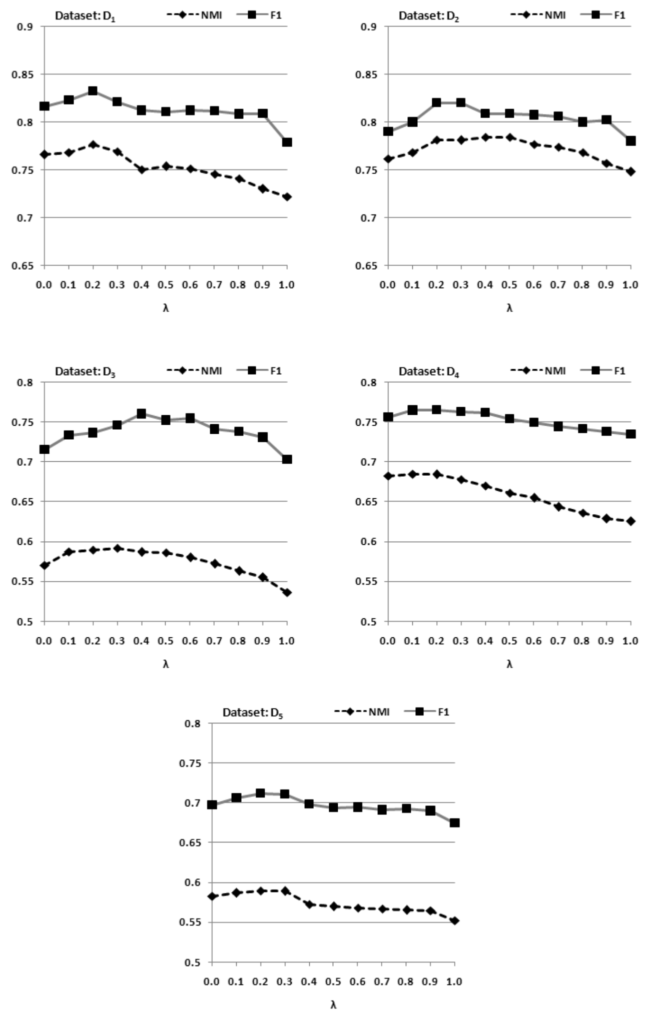


Fig. 4 The effect of varying the parameter λ on the spk-means clustering performance for each dataset. Eq. 11 is used to determine the term self-weight α_ν when computing the *ltcv* histograms.

by the proposed semantic smoothing which takes into account the local contextual information associated with a term occurrence. GTCV-VSM requires moderate values for the parameter σ to achieve better performance. The same is observed for the quality (in terms of NMI or F_1) of the best solution (i.e. the one with maximum *Cohesion*) found in the 100 runs, where moderate values of σ (i.e. σ=5 or 10) result in better GTCV-VSM performance. Moreover, the clustering results for a wide range of values of the smoothing parameter σ indicate that the method is quite robust to the specification of this parameter. GTCV-VSM behaves similarly to BOW when a low value is set for σ, while when this value becomes very high the discriminative information of the global term context vectors is reduced. This was demonstrated using spk-means and spectral clustering methods. Among them, the latter in all cases except from D_5 presented better average clustering solutions in terms of both evaluation measures NMI and F_1, while interestingly, spk-means was superior in terms of the best clustering solutions in most cases (with the exception of D_3) despite operating in a feature space of a much larger size.

6 Conclusions

We have presented the Global Term Context Vector-VSM (GTCV-VSM) document representation, an extension to the Vector Space Model that determines a proper feature space to project the typical VSM document vector representations. Our approach is entirely corpus-based and operates in the preprocessing in a sequence of four steps: i) captures local contextual information associated with each term occurrence in the term sequences of documents; ii) summarizes the local context vectors of each term into the respective global term context vectors; iii) constructs the semantic matrix for a problem using the global term context vectors; and finally iv) projects documents using the semantic matrix. The proposed approach achieves semantic smoothing by reducing data sparsity, while retaining the original dimensionality. The derived representation maintains the initial interpretability since each dimension is associated with a single vocabulary term. In the experimental document clustering study, we compared the proposed representation with the typical VSM, the Generalized-VSM and CVM-VSM, using Cosine similarity. The statistical analysis of the obtained results indicates that GTCV-VSM assists well-known clustering algorithms, such as spherical k-means and spectral clustering, to achieve better clustering solutions compared to other representation methods.

Our plans for future work are to investigate the potential of combining the local and global contextual information associated with terms to explore ways of building compact concept vectors, to efficiently project the transformed document vectors in feature spaces of lower dimensionality and to perform a systematic study for procedures that could efficiently compute α_ν parameters (Eq. 13) for each vocabulary term, which could improve the global term context vectors. Finally, we aim at examining the proposed representation for document classification.

References

1. AlSumait L, Domeniconi C (2008) Text clustering with local semantic kernels. In: Berry M, Castellanos M (eds) Survey of Text Mining II, Springer London, pp 219–232
2. Apté C, Damerau F, Weiss SM (1994) Towards language independent automated learning of text categorization models. In: SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Springer-Verlag New York, Inc., New York, NY, USA, pp 23–30
3. Beil F, Ester M, Xu X (2002) Frequent term-based text clustering. In: KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge discovery and data mining, ACM, New York, NY, USA, pp 436–442, DOI http://doi.acm.org/10.1145/775047.775110

4. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is "nearest neighbor" meaningful? In: ICDT '99: Proceedings of the 7th International Conference on Database Theory, Springer-Verlag, London, UK, pp 217–235
5. Billhardt H, Borrajo D, Maojo V (2002) A context vector model for information retrieval. Journal of the American Society for Information Science and Technology 53(3):236–249, DOI http://dx.doi.org/10.1002/asi.10032
6. Chen C, Tseng F, Liang T (2010) An integration of fuzzy association rules and wordnet for document clustering. Knowledge and Information Systems (available online):(available online), DOI http://dx.doi.org/10.1007/s10115-010-0364-2
7. Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. Journal of the American Society for Information Science 41:391–407
8. Dhillon I, Modha D (2001) Concept decompositions for large sparse text data using clustering. Machine Learning 42(1):143–175, DOI http://dx.doi.org/10.1023/A:1007612920971
9. Farahat A, Kamel M (2010) Statistical semantics for enchancing document clustering. Knowledge and Information Systems (available online):(available online), DOI http://dx.doi.org/10.1007/s10115-010-0367-z
10. Fung B, Wang K, Ester M (2003) Hierarchical document clustering using frequent itemsets. In: Proceedings of SIAM International Conference on Data Mining
11. Ghosh J, Strehl A (2006) Similarity-based text clustering: A comparative study. In: Kogan J, Nicholas C, Teboulle M (eds) Grouping Multidimensional Data, Springer Berlin Heidelberg, pp 73–97
12. Grauman K, Darrell T (2007) The pyramid match kernel: Efficient learning with sets of features. The Journal of Machine Learning Research 8:725–760, DOI http://doi.acm.org/10.1145/361219.361220
13. Hotho A, Maedche E, Staab S (2001) Ontology-based text document clustering. Knstliche Intelligenz 4:48–54
14. Hu X, Sun N, Zhang C, Chua T (2009) Exploiting internal and external semantics for the clustering of short texts using world knowledge. In: Proceeding of the 18th ACM conference on Information and knowledge management, ACM, New York, NY, USA, CIKM '09, pp 919–928, DOI http://doi.acm.org/10.1145/1645953.1646071
15. Jing J, Zhou L, Ng M, Huang Z (2006) Ontology-based distance measure for text clustering. In: Proceedings SIAM SDM Workshop on Text Mining
16. Karypis G, Han E (2000) Concept indexing: a fast dimensionality reduction algorithm with applications to document retrieval and categorization. In: Technical Report TR-00-0016, University of Minnesota
17. Keikha M, Razavian N, Oroumchian F, Razi H (2008) Document representation and quality of text: An analysis. In: Berry M, Castellanos M (eds) Survey of Text Mining II, Springer London, pp 219–232
18. Lebanon G, Mao Y, Dillon J (2007) The locally weighted bag of words framework for document representation. Journal of Machine Learning Research 8:2405–2441
19. Lewis D (1992) An evaluation of phrasal and clustered representations on a text categorization task. In: SIGIR '92: Proceedings of the 15th annual International ACM SIGIR Conference on Research and Development

in Information Retrieval, ACM, New York, NY, USA, pp 37–50, DOI http://doi.acm.org/10.1145/133160.133172

20. Li Y, Chung S, Holt J (2008) Text document clustering based on frequent word meaning sequences. Data & Knowledge Engineering 64(1):381–404, DOI http://dx.doi.org/10.1016/j.datak.2007.08.001
21. McQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability, pp 281–297
22. Miller G, Beckwith R, Fellbaum C, Gross D, Miller K (1990) Wordnet: An on-line lexical database. International Journal of Lexicography 3:235–244
23. Mladenic D (1998) Machine learning on non-homogeneous, distributed text data. PhD thesis, University of Ljubljana, Faculty of Computer and Information Science
24. Ng A, Jordan M, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems 14:849–864
25. Ni X, Quan X, Lu Z, Wenyin L, Hua B (2010) Short text clustering by finding core terms. Knowledge and Information Systems pp 1–21, DOI http://dx.doi.org/10.1007/s10115-010-0299-7
26. Porter M (1997) An algorithm for suffix stripping. In: Jones K, Willett P (eds) Readings in information retrieval, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 313–316
27. Pu W, Liu N, Yan S, Yan J, Xie K, Chen Z (2007) Local word bag model for text categorization. In: ICDM '07: Proceedings of the 2007 7th IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, pp 625–630, DOI http://dx.doi.org/10.1109/ICDM.2007.69
28. Salton G, Wong A, Yang C (1975) A vector space model for automatic indexing. Communications of the ACM 18(11):613–620, DOI http://doi.acm.org/10.1145/361219.361220
29. Wang P, Domeniconi C (2008) Building semantic kernels for text classification using wikipedia. In: KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge discovery and data mining, ACM, New York, NY, USA, pp 713–721, DOI http://doi.acm.org/10.1145/1401890.1401976
30. Wikipedia (2004) Wikipedia, the free encyclopedia. URL http://en.wikipedia.org/
31. Wong S, Ziarko W, Wong P (1985) Generalized vector spaces model in information retrieval. In: SIGIR '85: Proceedings of the 8th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, pp 18–25, DOI http://doi.acm.org/10.1145/253495.253506