# Term Based Semantic Clusters for Very Short Text Classification

**Jasper Paalman**
Jheronimus Academy of Data Science
j.v.paalman@tilburguniversity.edu

**Shantanu Mullick**
School of Industrial Engineering
Eindhoven University of Technology
s.mullick@tue.nl

**Kalliopi Zervanou**
School of Industrial Engineering
Eindhoven University of Technology
k.zervanou@tue.nl

**Yingqian Zhang**
School of Industrial Engineering
Eindhoven University of Technology
yqzhang@tue.nl

## Abstract

Very short texts, such as tweets and invoices, present challenges in classification. Although term occurrences are strong indicators of content, in very short texts, the sparsity of these texts makes it difficult to capture important semantic relationships. A solution calls for a method that not only considers term occurrence, but also handles sparseness well. In this work, we introduce such an approach, the **T**erm **B**ased **Se**mantic **C**lusters (TBSeC) that employs terms to create distinctive semantic concept clusters. These clusters are ranked using a semantic similarity function which in turn defines a semantic feature space that can be used for text classification. Our method is evaluated in an invoice classification task. Compared to well-known content representation methods the proposed method performs competitively.

## 1 Introduction

Bag-of-words approaches (Harris, 1954) to text classification rely on measures of term occurrence, or co-occurrence, such as $tf \cdot idf$ (Salton and Buckley, 1988), log-likelihood (Dunning, 1993), and mutual information (Church and Hanks, 1990). Such methods, despite their enduring popularity, are well known for their shortcomings in dealing with numerous natural language issues, such as morphological, semantic, and other types of variation and ambiguity (Augenstein et al., 2017). These issues become more critical in very short texts, such as microblogs, chat logs, reviews and invoices, because of the lack of data and context that could provide more reliable measures and a source for semantic disambiguation.

Distributional semantic models (Baroni and Lenci, 2010) such as pre-trained word embeddings (Pennington et al., 2014; Grave et al., 2018) encode words as fixed sized real-valued vectors. Embeddings may address the issues of data sparseness and lack of extensive context, by providing a semantic representation model that is not as sensitive to literal word occurrence in the data, thus providing semantic information coverage of out-of-vocabulary words (Bojanowski et al., 2017). This is because embeddings may allow for any given word to be mapped to a real-valued vector (e.g. by using character n-grams), even if it hasn't been observed during training. Additionally, embeddings implicitly capture semantic, syntactic and lexical properties, thereby representing implicit relationships. In this way, embeddings provide a rich representation that is otherwise difficult to attain. In a short text scenario, despite early findings that a small corpus size may not be sufficient for representing the relationships between words and documents (Song et al., 2014), availability of pre-trained embeddings makes this issue less of a concern. Despite these advantages and growing popularity, embeddings trained on general language data usually do not perform well in (i) specialised domains (Liu et al., 2018; Kameswara Sarma, 2018) and in (ii) languages with richer morphological variation than English (Zervanou et al., 2014). In this work, we attempt to address these issues while exploiting the advantages of pre-trained word embeddings in a text classification task for very short, domain specific texts in a morphologically rich language, i.e., invoices in Dutch.

Our aim is to classify short invoice descriptions, in such a way that each class reflects a different group of products or services, as illustrated in the examples in Table 1. When considering such texts, the augmented information offered by embeddings is crucial, because such texts abound in ellipsis, grammatical errors, misspellings, and


*Proceedings of Recent Advances in Natural Language Processing*, pages 878–887,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_102

| Text | Class |
|---|---|
| Vervoer Almere-Lille<br>*Transport Almere-Lille* | Travel expenses |
| 600GB SAS interne harde schijf<br>*600GB SAS internal hard drive* | Automation hardware |

Table 1: Example documents and classes

semantic variation. The inherent advantage of embeddings in dealing with out-of-vocabulary words presents, at the same time, the disadvantage of providing a text representation that does not focus on the importance of individual terms for the classification. Conversely, measures of term occurrence focus heavily on individual term importance but are very sensitive to variation. In very short text and domain-specific applications, where occurring terms are both strong indicators of the respective text class, as well as abound in variation, the preferred solution would combine embeddings to extracted terms. For example, for invoices, each occurring term in an invoice description is highly informative of the respective invoice text class. Hence a method is required that not only focuses on such terms, but also leverages the flexibility of embeddings. Our proposed method **T**erm **B**ased **Se**mantic **C**lusters (TBSeC) attempts to provide such a solution. The contribution of this paper lies in (i) combining the advantages of word embeddings with conventional term extraction techniques (ii) apply our method in an application domain not previously investigated, namely invoice text, which is characterised by specialised terminology and very short, elliptical and/or ungrammatical text, in a language that is morphologically richer than English and therefore posing an additional challenge in statistical approaches.

TBSeC proposes a two-stage methodology. In the first stage we use class-specific textual information to build semantic concept clusters. Concept clusters are vector representations of strongly related terms that are distinctive for a certain class. In the second stage, we compute cluster similarity scores on generated concept clusters for a given description. This serves as a ranking function that can be used in both unsupervised and supervised learning tasks.

The remainder of this paper is organized as follows: section 2 discusses related work; section 3 elaborates on the TBSeC method; section 4 outlines the experimental setup and section 5 discusses the results. We conclude with our main observations and suggestions for further work.

## 2 Related work

Text or document classification is defined as the assignment of text sections or entire documents to a predefined set of categories (Feldman and Sanger, 2007). For this purpose, algorithms process various types of text representations which are used as features for describing content. To our knowledge, invoice text classification has not been investigated previously[1]. Work related to text classification of short texts has been applied for microblogs (Singh et al., 2016; Ren et al., 2016; Missier et al., 2016), email subject classification (Alsmadi and Alhami, 2015) and spam detection (Bahgat et al., 2016).

Initial approaches to document content representation used counts of term frequency and inverse document frequency, $tf \cdot idf$ (Salton and Buckley, 1988), whereby frequently occurring terms are assumed to represent document content and inverse document term frequency scores select the most distinctive terms for a given document within a collection. Various subsequent approaches use variants of term occurrence measures with probabilities, such as $\chi$2-test, log likelihood (Dunning, 1993) and mutual information (Church and Hanks, 1990), or attempt to combine statistical measures with various types of linguistic and stop-word filters, so as to refine the keyword results. Considerations regarding term ambiguity and variation also led to rule-based approaches (Jacquemin, 2001) and resource-based approaches exploiting existing thesauri and lexica, such as UMLS (Hliaoutakis et al., 2009), or WordNet (Aggarwal et al., 2018). Knowledge poor statistical approaches, such as Latent Semantic Analysis (Deerwester et al., 1990) and Latent Dirichlet Allocation (Blei et al., 2003) attempt to detect document content in an unsupervised manner while reducing the dimensionality of the feature space of other bag-of-word approaches, but are also sensitive to sparse data and variation in short texts.

The advent of large semantic resources in the form of pre-trained word embeddings (Pennington et al., 2014; Grave et al., 2018) gave rise to a new line of approaches employing word embeddings for document content representation, such as Word2Vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017), GloVe (Pennington et al.,

[1] An approach reported by Bartoli et al. (2010) focuses on image features of scanned invoices rather than the invoice text.

2014), and ELMo (Peters et al., 2018). Within this line of approaches, methods have also been developed using word embeddings specifically for document classification, such as task-oriented word embeddings (Liu et al., 2018) and word-sense based embeddings (Jin et al., 2016). Embedding models encoding words in documents or document sections have also been developed, such as Doc2vec (Le and Mikolov, 2014), Infersent (Conneau et al., 2017), Skip-thought (Kiros et al., 2015) and FastSent (Hill et al., 2016). Such type of embeddings can be employed to calculate the semantic similarity between texts, but the risk is that intricate word-specific semantic relationships are lost. Methods originating from text similarity research using word rather than document embeddings, such as Kusner et al. (2015); Kenter and De Rijke (2015); De Boom et al. (2016) attempt to address this issue. Embeddings have been also used for keyphrase extraction via supervised (Mahata et al., 2018) and unsupervised (Bennani-Smires et al., 2018) approaches.

Finally, the problem of variation and data sparsity with very short texts has been addressed in the past with query expansion approaches (Vechtomova, 2009). For text similarity purposes query expansion techniques have been used for document term augmentation, exploiting relevant search results and a matrix representation (Sahami and Heilman, 2006; Abhishek and Hosanagar, 2007) or a combination of search results page count difference and lexico-syntactic patterns derived from text snippets (Bollegala et al., 2007).

## 3 The TBSeC methodology

Our proposed method, TBSeC, consists of two stages: In the first stage we use class-specific textual information to build semantic concept clusters. Concept clusters are vector representations of strongly related terms that are distinctive for a certain class. In the second stage, we compute cluster similarity scores on generated concept clusters for a given description, thereby forming a semantic feature space. This serves as a ranking function that can be used in both unsupervised and supervised learning tasks.

### 3.1 Concept clustering

Concept clustering starts by extracting distinctive terms, particular to a class[2]. Distinctive terms are extracted based on normalized term frequency. For a given class, word embeddings belonging to found terms are used to form numerous clusters. Hence, for each class multiple clusters are created and each cluster can be seen as a group of terms that are closely related. Each cluster is transformed into a concept cluster vector by taking the respective word embeddings of terms and computing the mean over each dimension. This process is illustrated in Figure 1, with actual examples included and word embeddings indicated as vectors with dots.

Specifically, the method works as shown in Algorithm 1 for $n$ distinct classes, $k$ most frequent terms to be incorporated as cluster and normalized term frequency (i.e., Bag-of-words, $B_{ij}$) threshold $t$. A more detailed description is given underneath the algorithm. Algorithm line numbers refer to steps as denoted in Figure 1.

Terms that provide a clear distinctive value to a class are retrieved using term frequency and L1 normalization over rows and columns. Terms not appearing in the vocabulary of the embedding model, or having a score below the normalized threshold $t$ are filtered out. Word embeddings for these selected terms are employed to create concept clusters for each class using the DBSCAN clustering model (Ester et al., 1996). Terms can either be included in a multi-term cluster through DBSCAN clustering or can be added as a single-term cluster when occurring frequently enough based on $k$. Ultimately, for each class $i$, concept clusters are created as a single vector equal to the averaged embedding of included terms.

### 3.2 Semantic cluster similarity

We adapt the similarity measure by Kenter and De Rijke (2015), which is based on the BM25 framework (Robertson et al., 2009), to propose our semantic cluster similarity measure. We combine idf scoring with word by word cosine similarity to calculate a weighted similarity score.

The Kenter and De Rijke (2015) function for calculating semantic text similarity between two sentences $s_l$ and $s_s$ is as defined as follows:

[2]In the case of invoice classification, each class refers to a specific expense type.

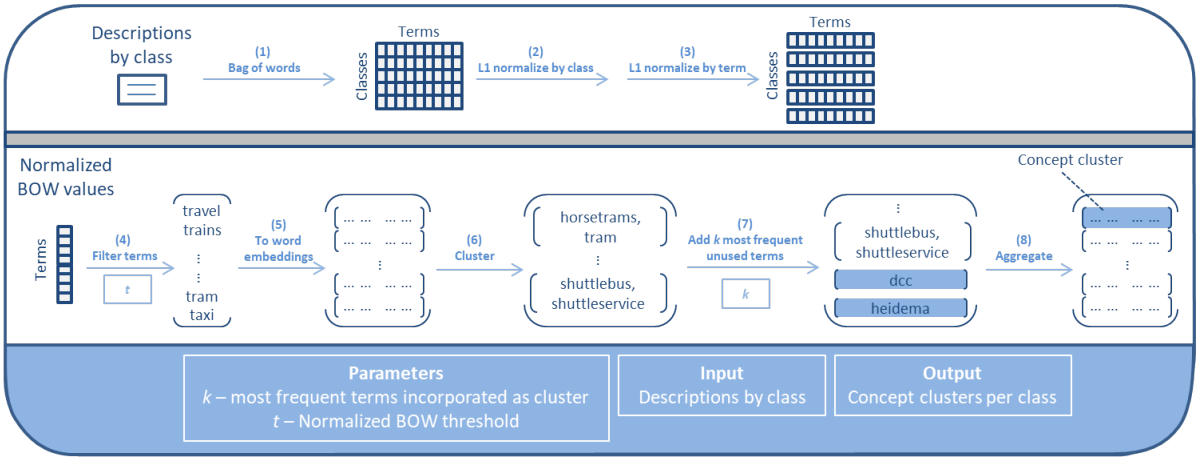


Figure 1: Concept clustering process diagram. *Upper panel*: Preparing normalized bag of words using descriptions by class. *Lower panel*: Employing found values to create concept clusters for a given class

$$f_{sts}(s_l, s_s) = \sum_{w \in s_l} IDF(w) \cdot \frac{sem(w, s_s) \cdot (k_1 + 1)}{sem(w, s_s) + k_1 \cdot (1 - b + b \cdot \frac{|s_s|}{avgsl})} \quad (1)$$

Parameters $k_1$ and $b$ are inherited from the BM25 framework and serve as smoothing parameters. Parameter $k_1$ influences what semantic text similarity value is approached asymptotically, thereby limiting the influence that semantic term similarity can have. Parameter $b$ provides a degree of importance to sentence length ratio $\frac{|s_s|}{avgsl}$, comparing sentence length to the average sentence length $avgsl$ of sentences being ranked. The function $sem$ returns the semantic term similarity of term $w$ with respect to text $s$, as follows:

$$sem(w, s) = \max_{w' \in s} f_{sem}(w, w') \quad (2)$$

where $f_{sem}$ is a vector similarity function that is typically computed as cosine similarity.

In TBSeC, the Kenter and De Rijke (2015) function is adapted to compare each sentence to all composed concept clusters. In our implementation, parameter $b$ is redundant, because each concept cluster is represented as a single embedding and the measure is computed by a single cosine similarity score. Moreover, we normalize our similarity score with the number of terms appearing in the sentence to allow for use in a supervised learning task. Finally, we remove term-specific weighting, for four reasons: First, idf scoring imposes a hefty constraint on the terms that can be used because of the predefined vocabulary. Second, we argue that the limited amount of terms appearing in a description justifies the exclusion of term-specific weighting. Each term in the description holds an important piece of information and differentiating is not essential. Third, terms that don't hold considerable semantic importance are not likely to steer the score towards an incorrect class. Each concept cluster is created to serve as a distinctive concept, thus making it unrealistic that unimportant terms will relate to it well. Fourth, terms in descriptions are subject to frequent misspellings and personal abbreviations, making idf scores inherently unreliable in this setting.

Based on the changes to Equation 1 discussed above, our function for calculating semantic cluster similarity $f_{scs}$ between text $s$ and concept cluster $c$ is defined as follows:

$$f_{scs}(s, c) = \frac{1}{|s|} \cdot \sum_{w \in s} \frac{(sem(w, c) \cdot t(w)) \cdot (k_1 + 1)}{(sem(w, c) \cdot t(w)) + k_1} \quad (3)$$

where $t(w)$ is the term frequency of term $w$ in the text. The score is normalized by the number of terms $|s|$ in text $s$. In addition to smoothing parameter $k_1$, two other hyper parameters $sem_{th}$ and $sem_{sq}$ are added to influence scoring. These hyper parameters affect the result of $sem$ as follows.

$$sem(w, c) = \begin{cases} f_{sem}(w, c) & \text{if } f_{sem}(w, c) \geq sem_{th} \\ & \text{and } sem_{sq} = \text{false} \\ f_{sem}(w, c)^2 & \text{if } f_{sem}(w, c) \geq sem_{th} \\ & \text{and } sem_{sq} = \text{true} \\ 0 & \text{if } f_{sem}(w, c) < sem_{th} \end{cases} \quad (4)$$

**Algorithm 1** Concept Clustering

**Parameters:**
$k$ - No. of most frequent terms to be incorporated as cluster
$t$ - Normalized bag of words threshold
**Input:**
$D_i$ , $i \in \{1, \ldots, n\}$ - Merged descriptions for each class
**Output:**
$C_i$ , $i \in \{1, \ldots, n\}$ - Concept clusters for each class

① $B_{ij} \leftarrow$ Calculate bag of words using $D_i$, with found vocabulary set $V$, $i \in \{1, \ldots, n\}, j \in \{1, \ldots |V|\}$
• $C_i \leftarrow$ Instantiate empty concept cluster array, with $i \in \{1, \ldots, n\}$

• **for** $i \in \{1, \ldots, n\}$ **do**
② $B_{i*} \leftarrow$ L1 normalize $B_{i*}$ ▷ Normalize by class
• **end for**
• **for** $j \in \{1, \ldots |V|\}$ **do**
③ $B_{*j} \leftarrow$ L1 normalize $B_{*j}$ ▷ Normalize by term
• **end for**
• **for** $i \in \{1, \ldots, n\}$ **do** ▷ For each class
④ Retrieve terms in vocabulary with a score $> t$ and occur more than once
• $terms \leftarrow$ Array of found terms that appear in word embedding vocabulary
⑤ $emb \leftarrow$ Respective word embeddings of $terms$
• $C_i \leftarrow$ CREATE CLUSTERS($emb, k, terms$)
• **end for**
• **return** $C$

• **function** CREATE CLUSTERS($emb, k, terms$)
• Instantiate $DBSCAN$ clustering model with $eps$, $min\_samples$ and $metric = cosine\,similarity$
• Fit $emb$ on $DBSCAN$ model
⑥ $clusters \leftarrow$ formed clusters as collections of word embeddings
• **for** $term$ in $k$ most frequent $terms$ **do**
• **if** $term$ not used in $clusters$ **then**
• $e \leftarrow$ word embedding of $term$
⑦ $cluster$.append($[e]$)
• **end if**
• **end for**
• $concepts_c \leftarrow$ empty concept cluster array with $c \in \{1, \ldots, |clusters|\}$
• **for** $c \in \{1, \ldots, |clusters|\}$ **do**
⑧ $concepts_c \leftarrow$ coordinate mean over each dimension
• **end for**
• **return** $concepts$
• **end function**

The semantic threshold $sem_{th}$ serves as a way to add a semantic similarity bound above which it will be presumed to hold importance. Parameter $sem_{th}$ achieves that when $f_{sem}(w, c)$ is under the set threshold value, that $sem(w, c)$ equals 0. Squaring $f_{sem}(w, c)$ through $sem_{sq}$ increases term importance of terms that are a near match and lowers importance of terms that match to a lesser extend. Squaring $f_{sem}(w, c)$ therefore promotes the divergence of semantic similarity scores.

Semantic cluster similarity $f_{scs}$ produces features for use in supervised learning applications, but the initial performance of TBSeC is measured without a predictive model. For this reason, a similarity score for each class is required in order to rank the classes. This is calculated as $scs$ for each class $i$, by extracting the maximum score over all concept clusters $c \in C_i$ (see $C_i$ in Algorithm 1):

$$scs(s, C_i) = \max_{c \in C_i} f_{scs}(s, c) \tag{5}$$

## 4 Experimental setup

This section covers data description, data processing and the experimental set-up for our method.

### 4.1 Data

Our invoice data originate from an auditing company. In the data, as illustrated in the examples in Table 1, each class refers to a particular type of expenses. Available data is accessed from a data directory, where each file is specific to a client. For our purposes, only the *invoice description* and *class assignment* are relevant. The volume of the entire data directory amounts to approximately 1.5 million instances. There is a total of 111 unique classes to which assignments are made. The five classes that are least represented have 24, 106, 178, 418 and 452 entries respectively. Invoice descriptions on average contain 2.80 terms, with a standard deviation of 1.55.

### 4.2 Word embeddings

Pre-trained Dutch FastText word embeddings[3] are used for sentence embedding construction and for use in semantic similarity computations (Bojanowski et al., 2017). The FastText embedding model was trained on Dutch Wikipedia.

[3] https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md

### 4.3 Data processing

Descriptions are processed using a procedure similar to the one used in training the FastText model. Special characters are replaced with a whitespace, stopwords in both the English and Dutch language are dropped, digits are removed and finally terms are retrieved by splitting on any sequence of unicode whitespace characters. When creating validation sets special care is taken to remove duplicates and to include data from all individual clients and all classes in a randomized manner. As a result, largely balanced validation sets are formed with data from various sources.

### 4.4 Learning algorithm

During supervised learning a Support Vector Machine[4] is used with regularization parameter $C = 0.1$ and a linear kernel. This classifier performed best when compared to other feasible classifiers (e.g. random forest), given a local working memory bounded set-up, and allows for the use of sparse matrices. Regularization parameter $C$ regulates the importance of focusing on correctly classifying training samples in favor of realizing a hyperplane with a large minimum margin. A high $C$ can lead to overfitting, a low $C$ can lead to the inability to learn meaningful decision boundaries. We set $C$ to $0.1$ since it appears to offer a good balance on the basis of the main validation set in terms of limited running time and general performance.

### 4.5 Parameter tuning

Prior to including our framework in a supervised learning task, we optimize the parameters (see section 5.1 for results). We construct an initial set of concept clusters using preset values $k = 5$ and $t = 0.8$. Parameters are set such that the model offers a well-performing baseline with low chances of overfitting. Initial concept clusters are used to tune semantic cluster similarity parameters. This order of parameter tuning is chosen, because it is relatively straightforward to pick sensible values for $k$ and $t$, as opposed to $f_{scs}$ hyperparameters. Table 2 lists the attempted combinations of parameter settings for $f_{scs}$.

After parameter tuning for semantic cluster similarity, employed concept clusters are reconsidered. Values in the range from 5 to 50 with a step size of 5 are attempted for the $k$ most frequent terms incorporated as cluster.

| Parameter | Distinct values |
|---|---|
| $k_1$ | [1.2, 1.6, 2.0] |
| $sem_{sq}$ | [true, false] |
| $sem_{th}$ | [0, 0.1, 0.2, 0.3, 0.4] |

Table 2: $f_{scs}$ parameter settings

Validation is performed on a dataset with approximately 5,000 entries. Performance differences for $f_{scs}$ are evaluated using three performance measures: (1) accuracy, (2) ranking loss and (3) standardized score. Accuracy is calculated by picking the class with the highest semantic similarity score as predicted class and finding the percentage of correctly classified instances. Ranking loss is calculated by obtaining the rank of the true class. The standardized score is calculated by standardizing all scores for a given instance and retrieving the score of the true class. By standardizing it can be observed how the score for the true class is positioned against all other scores. Ultimately, the objective is to maximize the accuracy and standardized score and to minimize ranking loss. Afterwards, we investigate the influence of parameter $k$ for concept cluster construction on the basis of accuracy and dimension size. Accuracy is calculated as an unsupervised score as well as a 5-fold cross validated supervised score. Both scoring methods use best values for $k_1$, $sem_{th}$ and $sem_{sq}$ which are found in the previous step. Results are compared to determine an appropriate value for parameter $k$ for use of features in a predictive model.

### 4.6 Invoice classification

We use the proposed semantic cluster similarity matching method, to measure performance in a classification task. We compare the performance to existing methods and we test whether combining methods leads to an increase in performance.

The parameters for generation of semantic cluster similarity scores are set in accordance with values obtained during parameter tuning.

For validation, a dataset containing approximately 20,000 entries is used. The data is used to perform 5-fold cross validation to determine overall performance. For all tests, we chose to use accuracy as performance quality measurement. Only one out of 111 classes shows significant imbalance, which for testing overall method performance is deemed negligible. Consequently, no balancing is performed and no alternative quality measure, such

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

| Method | Parameters | Dimension size |
|---|---|---|
| $tf \cdot idf$ | - | 117,766 |
| $tf$ | - | 117,766 |
| LSA | # components: 100 | 100 |
| LDA | # topics: 200 | 200 |
| FastText | - | 300 |
| P-mean | P-values: $\{-\infty, 1, 2, 3, \infty\}$ | 1500 |

Table 3: Benchmark methods

| $k_1$ | $sem_{sq}$ | $sem_{th}$ | acc | loss | std score |
|---|---|---|---|---|---|
| 2.0 | true | 0.0 | **21.46** | **28.08** | **1.5773** |
| 2.0 | true | 0.1 | **21.46** | 28.09 | 1.5759 |
| 1.6 | true | 0.0 | 21.26 | 28.12 | 1.5412 |
| 1.6 | true | 0.1 | 21.24 | 28.13 | 1.5397 |
| 2.0 | true | 0.2 | 21.24 | 28.35 | 1.5127 |
| 1.6 | true | 0.2 | 21.26 | 28.40 | 1.4739 |

Table 4: Best $f_{scs}$ configurations

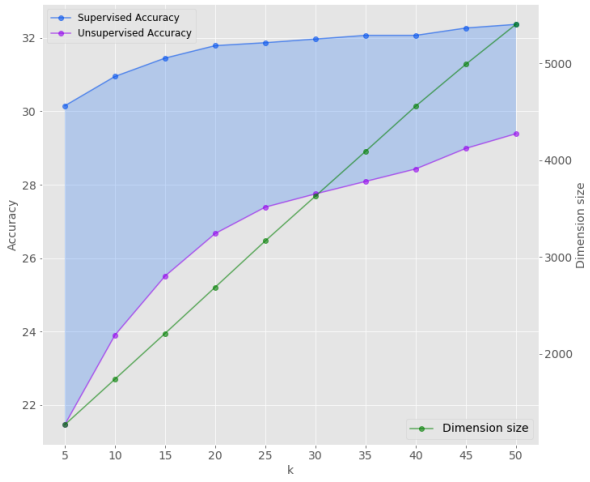


Figure 2: Concept clustering parameter tuning

as macro-averaged recall or F1-score is used.

Our benchmarks consist of other state-of-the-art content feature generation methods: term frequency ($tf$), $tf \cdot idf$, Latent Semantic Analysis (*LSA*, Deerwester et al. (1990)), Latent Dirichtlet Allocation (*LDA*, Blei et al. (2003)), concatenated power mean word embeddings (*P-mean*, Rücklé et al. (2018)), and FastText sentence embeddings (Bojanowski et al., 2017). All benchmark methods, parameter settings and dimension sizes are shown in Table 3.

Feature transformation models (*tf*, *tf·idf*, LSA, LDA) are trained on the entire training directory. Additionally, when using such transformation models, words are stemmed to reduce inflectional word forms to their word stem.

## 5 Results

This section discusses our results from parameter tuning and the subsequent supervised classification task.

### 5.1 Parameter tuning

#### 5.1.1 Semantic cluster similarity

Parameter combinations have been attempted and results have been retrieved for the used dataset. For each quality measure, the 3 best scores have been retrieved, returning all instances that conform to that score. The results are shown in Table 4.

Table 4 tells us that squaring the semantic similarity score works well. High accuracy scores are retrieved, accounting for the fact that evaluations are made based on maximum similarity scoring instead of a predictive model. As we are able to retrieve six configurations, we note that quality measures tend to score parameter settings similarly. The configuration with $k_1 = 2.0$, $sem_{sq} = true$ and $sem_{th} = 0.0$ across quality measures is the best performing.

#### 5.1.2 Concept clustering

Next, we study the influence of parameter $k$ on concept cluster construction. For each setting, we present the accuracy scores and dimension sizes in Figure 2.

With increasing $k$, more concept clusters of unused single terms are added. As a result, the dimension size steadily increases. We can see that accuracy also shows an increasing trend with the value of $k$. When more unused single terms are added, we run the risk of overfitting. The ratio of single terms to broader concept clusters increases with $k$, thereby relatively shifting the focus from broader concepts to frequently occurring distinctive individual terms. This behaviour is also reflected in the graph, in the relative performance difference between unsupervised and supervised scoring. Although supervised accuracy is undoubtedly higher with lower number of concept clusters, both accuracy scores converge with increasing $k$. By adding unused single terms, the unsupervised ranking method is able to capture an increasing number of edge cases, leading to a convergence in performance. This behaviour is an indication of overfitting.

After carefully considering the points above, we proceed with value $k = 20$. This choice appears to offer a good compromise between generalizing ability and direct performance gains. The primary

| Methods | test acc | test sd | train acc | train sd |
|---|---|---|---|---|
| **Benchmark** | | | | |
| $tf \cdot idf$ | 43.93 | 0.89 | 82.79 | 0.07 |
| $tf$ | **44.85** | 0.67 | 84.84 | 0.12 |
| LSA | 7.89 | 0.35 | 9.16 | 0.20 |
| LDA | 11.78 | 0.37 | 14.15 | 0.15 |
| TBSeC | 36.03 | 0.60 | 46.96 | 0.26 |
| FastText | 32.62 | 0.79 | 41.26 | 0.24 |
| P-mean | 39.78 | 0.63 | 90.33 | 0.1 |
| **Feature combinations** | | | | |
| $tf$ & TBSeC | 47.83 | 0.54 | 86.17 | 0.13 |
| $tf$ & FastText | 47.07 | 0.48 | 85.81 | 0.14 |
| $tf$ & TBSeC & FastText | **47.86** | 0.42 | 86.38 | 0.10 |

Table 5: Invoice classification results

goal of TBSeC is relating input to broader concepts, which is why a relatively moderate value for $k$ is preferred.

### 5.2 Invoice classification

In this section, we discuss the results of our method against state-of-the art benchmark methods in a supervised invoice classification task. The results, consisting of cross validated accuracy scores with standard deviation ($sd$), are shown in Table 5 under header 'Benchmark'. The fact that term frequency has comparable performance to $tf \cdot idf$ reinforces the notion that all terms within an invoice description are important to take into account and that term-specific weighting has limited value. Moreover, techniques that are concerned with dimensionality reduction (LSA, LDA) perform worse, arguably because they truncate a large amount of information, most of which should have been retained. Sentence embeddings and our method TBSeC perform relatively well, with accuracy scores nearing performance levels of $tf \cdot idf$ and term frequency. Furthermore, a large feature space (1500D) that is achieved with P-mean embeddings appears to have a positive influence on the amount of information that is contained. It is also found that methods $tf \cdot idf$, term frequency and P-mean have a tendency to overfit on the data, having cross-validation training accuracy scores of over 80%. In comparison, TBSeC and FastText have training accuracy scores closer to test accuracy scores.

Combinations of techniques are attempted next to improve performance. The feature combinations are formed by concatenating the feature spaces of each method. The most successful combinations are highlighted in Table 5 under header 'Feature combinations'. Combining feature generation techniques leads to surprising results. P-mean performs well as sole feature, but doesn't yield better results when combined with other techniques. In contrast, FastText does pair well with other techniques and improves performance levels. Moreover, when TBSeC is used performance is even better. Term frequency in combination with both TBSeC and FastText does also improve performance, although slightly. TBSeC and FastText are more likely to be complementary as soon as TBSeC encompasses lower number of concept clusters, but this doesn't guarantee better overall performance. In the current configuration it manages to perform better, but the difference is not major. The tendency to overfit for $tf \cdot idf$, term frequency and P-mean is a likely cause for some of the other unsuccessful feature combinations.

## 6 Conclusion

A new feature generation framework TBSeC was presented that is suited to the prediction of well defined classes on the basis of very short texts (2.8 words on average). Generated features were proven to be able to function well independently and jointly with traditional feature generation techniques. Performance and reliability was improved by pairing multiple disjoint feature generation techniques, including TBSeC. A combination of highly specific features with more flexible ones was found to lead to the best results. Combinations of features were found to reach a bound in effectiveness, highlighting that methods ultimately start impeding each other. Businesses can use our method to derive actionable insights from online user generated content such as firm-specific tweets, online reviews and customer chats logs. Future work could test TBSeC on larger texts, since it offers more room to differentiate from sentence embeddings.

## References

Vibhanshu Abhishek and Kartik Hosanagar. 2007. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the ninth international conference on Electronic commerce*. ACM, pages 89–94.

Ayush Aggarwal, Chhavi Sharma, Minni Jain, and Amita Jain. 2018. Semi supervised graph based keyword extraction using lexical chains and centrality measures. *Computación y Sistemas* 22(4).

Izzat Alsmadi and Ikdam Alhami. 2015. Clustering and classification of email contents. *Journal of King*

*Saud University-Computer and Information Sciences* 27(1):46–57.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 546–555.

Eman M Bahgat, Sherine Rady, and Walaa Gad. 2016. An e-mail filtering approach using classification techniques. In *The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Beni Suef, Egypt*. Springer, pages 321–331.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Alberto Bartoli, Giorgio Davanzo, Eric Medvet, and Enrico Sorio. 2010. Improving features extraction for supervised invoice classification. In *Artificial Intelligence and Applications*. MH Hamza.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Brussels, Belgium, pages 221–229. https://www.aclweb.org/anthology/K18-1022.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. *www* 7:757–766.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters* 80:150–156.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391–407.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19(1):61–74.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*. volume 96, pages 226–231.

Ronen Feldman and James Sanger. 2007. *The text mining handbook : advanced approaches in analyzing unstructured data*. Cambridge University Press.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483* .

Angelos Hliaoutakis, Kalliopi Zervanou, and Euripides G. M. Petrakis. 2009. The AMTEx approach in the medical document indexing and retrieval application. *Data and Knowledge Engineering* 68(3):380–392.

Christian Jacquemin. 2001. *Spotting and Discovering Terms Through Natural Language Processing*. MIT Press.

Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for text classification. In *IJCAI*. volume 16, pages 2824–2830.

Prathusha Kameswara Sarma. 2018. Learning word embeddings for data sparse and sentiment rich data sets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. Association for Computational Linguistics, New Orleans, Louisiana, USA, pages 46–53.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, pages 1411–1420.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*. pages 957–966.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. pages 1188–1196.

Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian, and Luyang Liu. 2018. Task-oriented word embedding for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 2023–2032.

Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. pages 634–639.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Paolo Missier, Alexander Romanovsky, Tudor Miu, Atinder Pal, Michael Daniilakis, Alessandro Garcia, Diego Cedrim, and Leonardo da Silva Sousa. 2016. Tracking dengue epidemics using twitter content classification and topic modelling. In *International Conference on Web Engineering*. Springer, pages 80–92.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Yafeng Ren, Ruimin Wang, and Donghong Ji. 2016. A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences* 369:188–198.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4):333–389.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated $p$-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400* .

Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*. AcM, pages 377–386.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

Monika Singh, Divya Bansal, and Sanjeev Sofat. 2016. Behavioral analysis and classification of spammers distributing pornographic content in social media. *Social Network Analysis and Mining* 6(1):41.

Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short text classification: A survey. *Journal of multimedia* 9(5):635–644.

Olga Vechtomova. 2009. *Query Expansion for Information Retrieval*, Springer US, Boston, MA, pages 2254–2257.

Kalliopi Zervanou, Elias Iosif, and Alexandros Potamianos. 2014. Word semantic similarity for morphologically rich languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*. pages 1642–1648.