No Target Function Classifier
Fast Unsupervised Text Categorization using Semantic Spaces

Tobias Eljasik-Swoboda[1], Michael Kaufmann[2] and Matthias Hemmje[1]
[1]*Faculty of Mathematics and Computer Science, University of Hagen, Universitätsstraße 47, 58084 Hagen, Germany*
[2]*Data Intelligence Research Team, Lucerne School of Information Technology, Suurstoffi 41, 6343 Rotkreuz, Switzerland*

Keywords: Text Categorization, Unsupervised Learning, Classification, Text Analytics.

Abstract: We describe a Text Categorization (TC) classifier that does not require a target function. When performing TC, there is a set of predefined, labeled categories that the documents need to be assigned to. Automated TC can be done by either describing fixed classification rules or by applying machine learning. Machine learning based TC usually occurs in a supervised learning fashion. The learner generally uses example document-to-category assignments (the target function) for training. When TC is introduced for any application or when new topics emerge, such examples are not easy to obtain because they are time-intensive to create and can require domain experts. Unsupervised document classification eliminates the need for such training examples. We describe a method capable of performing unsupervised machine learning-based TC. Our method provides quick, tangible classification results that allow for interactive user feedback and result validation. After uploading a document, the user can agree or correct the category assignment. This allows our system to incrementally create a target function that a regular supervised learning classifier can use to produce better results than the initial unsupervised system. To do so, the classifications need to be performed in a time acceptable for the user uploading documents. We based our method on word embedding semantics with three different implementation approaches; each evaluated using the reuters21578 benchmark (Lewis, 2004), the MAUI citeulike180 benchmark (Medelyan et al., 2009), and a self-compiled corpus of 925 scientific documents taken from the Cornell University Library arXiv.org digital library (Cornell University Library, 2016). Our method has the following advantages: Compared to key word extraction techniques, our system can assign documents to categories that are labeled with words that do not literally occur in the document. Compared to usual supervised learning classifiers, no target function is required. Without the requirement of a target function the system cannot overfit. Compared to document clustering algorithms, our method assigns documents to predefined categories and does not create unlabeled groupings of documents. In our experiments, the system achieves up to 66.73 % precision, 41.8 % recall and 41.09% F1 (all reuters21578) using macroaveraging. Using microaveraging, similar effectiveness is obtained. Even though these results are below those of contemporary supervised classifiers, the system can be adopted in situations where no training data is available or where text needs to be assigned to new categories capturing freshly emerging knowledge. It requires no manually collected resources and works fast enough to gather feedback interactively thereby creating a target function for a regular classifier.

1 INTRODUCTION AND MOTIVATION

Text Categorization (TC) is the assignment of documents to predefined categories based on their content. TC can be done manually or automatically (Sebastiani, 2002). Some examples of automated TC are classification of scientific documents or news articles, spam filtering, and work order routing to appropriate personnel. Automated TC uses two fundamental strategies: Fixed rules and machine learning. Formally, a target function $\Phi‘: D \ x \ C \rightarrow \{T, F\}$ defines whether a document $d \in D$ belongs to a certain category $c \in C$. A classifier $\Phi: D \ x \ C \rightarrow \{T, F\}$ then attempts to achieve results as similar as possible to the target function.

When introducing TC functionality to arbitrary applications, difficulties arise. Besides simple technological integration effort and dealing with

Eljasik-Swoboda, T., Kaufmann, M. and Hemmje, M.
No Target Function Classifier - Fast Unsupervised Text Categorization using Semantic Spaces.
DOI: 10.5220/0006847000350046
In *Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018)*, pages 35-46
ISBN: 978-989-758-318-6

heterogeneous data formats for different TC implementations, the main challenge is curating the necessary data to attempt either TC strategy. *Rule based TC* requires domain experts and information engineers to define rules, whereas *machine learning TC* requires already-labeled example documents to define target functions. Albeit crowd working can be used to mass-produce learning examples, some applications require a higher amount of expertise captured in the examples. In that case, domain experts must define target functions by manually tagging example documents. This is a challenge because experts usually have limited time to spend for this task.

In order to have a target function, an expert committee or a predefined gold-standard must fix the set of categories beforehand. If one has a document corpus D but no predefined set of categories C, unsupervised clustering algorithms are applicable after vectorization of the documents. This problem differs from TC because there is no fixed set of categories. Clustering documents does not always yield human-readable category labels. If one has a predefined C, or wants to identify specific *emerging topics* in large text corpora, neither document clustering or regular supervised learning classifiers can be used.

Our research originates from the Deutsche Forschungsgeselschaft (DFG) sponsored project RecomRatio, which is part of the Robust Argumentation Machines emphasis program (DFG, 2016) currently ramping up activities. The goal is to automatically provide medical professionals with treatment recommendations mined from current medical literature along with the analyzed literature that supports the recommendations. An *Information Retrieval* (IR) component that identifies relevant clinical studies is an integral part of the system. Text categorization is a central part of the IR component. Albeit there are plenty examples and target functions for medicine, emerging topics are not classifiable using legacy target functions. To address this issue, our system suggests a category for a new document upon ingestion within a 30-second limit. A human user can then confirm or correct this category. This way, a target function can be collected and the overall system has classification results that can be rapidly used.

Our research goal is to create a classifier that does not require a target function, that can assign documents to a predefined set of categories, and that spends less than 30 seconds per categorization while being able to recognize new, emerging categories. Striving for this goal, we attempt to answer the following research questions: (1) How can an unsupervised classifier be created that labels documents efficiently? (2) How well can a classifier without target function perform in terms of effectiveness? And (3) what influences classifier performance? The overreaching goal is to design a TC classifier that requires as little manually compiled information as possible while being able to quickly perform TC to predefined categories.

In this paper, we describe three approaches to implementing a *No Target Function Classifier* (NTFC) and evaluate their results using three benchmarks. Even though its effectiveness is below that of contemporary supervised learning classifiers, NTFC offers a number of intrinsic advantages: Compared to regular classifiers, no target function is needed. Without a target function, overfitting is impossible. Compared to key word or key-phrase extraction algorithms, documents can be assigned to categories that have words in their labels, which do not occur in the documents. Compared to clustering algorithms, our system does not create named or unnamed clusters of documents from a document collection but assigns documents to predefined categories. These features make the NTFC uniquely suited for emerging knowledge domains. After thorough experimentation, we conclude with a discussion and analysis of the evaluation results.

2 STATE OF THE ART

2.1 Text Vectorization

Machine learning algorithms usually work with scalars or vectors as input and output for their models (Mohri et al., 2012). If one aims to use these algorithms for any *Natural Language Processing* (NLP) task, then one must first generate vector representations of texts. One approach to capture natural language is to use ontologies (Busse et al., 15). These manually created, machine-readable representations of semantics are used to conceive meaning from natural language texts. We chose not to use ontologies because they must be manually created. This makes them unfit for emerging knowledge scenarios where quick results are needed and useful ontologies might not yet be available.

Fortunately, there is a large collection of mathematical approaches to capture document meaning that do require no other inputs than the documents themselves. In order to capture the meaning of documents, the sense of the terms making up the documents need to be grasped. The

Term Frequency Inverse Document Frequency (TFIDF) measure is an information retrieval method that models how representative term t is for document d (Salton and McGill, 1983). It contains the term frequency $\#(t, d)$ which is defined as the absolute number of how often *t* occurs in *d* and the document frequency $\#D(t)$, which is defined as the number of documents *t* occurs in.

$$TFIDF(t, d) = \#(t, d) * log \frac{|D|}{\#D(t)} \tag{1}$$

Equation 1 models two important intuitions: Firstly, the more often a term occurs in a document, the more it is representative for said document. Secondly, the more documents term *t* occurs in, the less discriminating it is between individual documents. Given that categories have natural language labels consisting of words, TFIDF can be used to generate relationships between documents and categories. A TFIDF vector represents a document as a set of TFIDF values, one for each possible term or for the most relevant terms. The terms making up the category labels have dimensions of the TFIDF vectors associated with them. This can be exploited for vectorization of free text.

For example, the Reuters benchmark contains the category *corn*. Every TFIDF vector representing a document has one dimension associated with the word *corn*. Without optimization, this is computed for every term occurring in either the set of documents or set of terms. This makes document representation by TFIDF vectors high dimensional, leading to the curse of dimensionality (Bellman, 1961). TFIDF vectors are usually sparse. Given the size of the vocabulary of an entire language, many individual terms do not occur in documents and are thus encoded with a 0 in the TFIDF vector. Analogous to TFIDF, TFICF is the same measure but used on category labels instead of documents (Cho and Kim, 1997).

The collection of all TFIDF vectors yields a TFIDF matrix. This matrix, or a simple matrix indicating how often which term occurs in which document, can be used for *topic modeling*. *Topic modeling* techniques rely on the fact, that every matrix can be expressed as the product of three individual matrices (equation 2).

$$A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^{\intercal} \tag{2}$$

The document-to-term matrix *A* is broken up into the document-to-topic matrix *U*, a diagonal topic matrix with positive entries Σ and a term-to-topic matrix *V*. Here, *topics* are abstract statistical entities. Documents are interpreted as probabilities of different topics to occur, while topics are interpreted as probabilities for different terms to occur. This insight is the basis for *Latent Semantic Analysis* (LSA), a method to remove terms from the document-to-term matrix while preserving similarity structures and *Latent Dirichlet Allocation* (LDA), a method to spot hidden structures in the document-to-term matrix (Dumais, 2005); (Blei et al., 2003). LSA and LDA are important techniques for text analysis. They however do not yield unsupervised classifiers, as the recognized topics are abstract statistical entities and not predefined, labeled categories. The resulting topics can be used to assign computable meaning to terms as every term can be expressed as vector of topic probabilities.

Explicit Semantic Analysis (ESA) (Egozi et al., 2011) also assigns meaning to individual terms. To do so, all Wikipedia articles of a given language are analyzed by computing the TFIDF values for each term across all articles. Then the resulting matrix is transposed. This creates term vectors in which each dimension represents the corresponding Wikipedia article’s TFIDF value. Unfortunately, the resulting vectors are sparse and of a very high dimensionality as the English Wikipedia has over 5,500,000 articles. LSA can be used to compress these vectors to lower dimensionality. Using knowledge mined from Wikipedia can be useful in many NLP applications, as it is freely available, frequently updated and provides plenty examples of language usage (Gabrilovich and Markovitch, 2006).

Word embeddings are a new class of approaches to reduce the dimensionality of sparse document vectors by mapping them into lower dimensional vectors with no information loss. Intriguingly, contemporary word embedding algorithms form *semantic spaces* by encoding meaning with the coordinates generated for the words. One such semantic space library is Mikolov’s Word2Vec, which captures syntactic as well as semantic relationships by encoding similar offsets between term vectors (Mikolov et al., 2013). Figure 1 illustrates this in two-dimensions showing offset vectors for gender and age. To the best of our knowledge, such semantic coordinate systems have not been observed in *topic modeling* based term vectors.

Given a large text as input, Word2Vec uses supervised learning by creating text windows of $2m$ words. The two Word2Vec algorithms are based on the assumption that words similar to each other occur in the context of the same surrounding words.

$w_{n-m}, w_{n-(m-1)}, \dots, w_{n-1}, w_n, w_{n+1}, \dots, w_{n+(m-1)}, w_{n+m}$

This is a fundamental difference to TFIDF, LSA, LDA and ESA, where terms are considered similar if they occur in the same document. Word embeddings are much finer grained, which could explain the offset encoding of relationships between terms.

In the *Continuous Bag Of Words* (CBOW) algorithm, the m words before and after w_n are taken as input for a neural network while w_n is the expected output. The *skip-gram* algorithm reverses this pattern and uses the context as expected output and w_n as input for these outputs. The implementations are open source and publicly available (Mikolov, 2013).

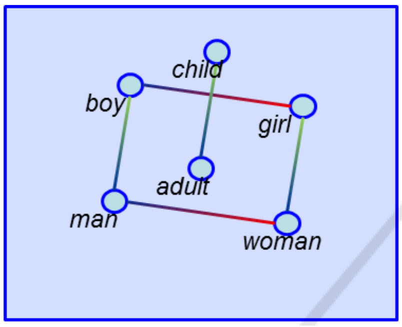

Figure 1: 2D example of a semantic space.

Mikolov et al.'s algorithms optimize the vectors representing each word $v(w)$ in such a way that the cosine similarity (equation 3) for words in their individual context windows is maximized. If for example the terms $t_1 = "buy"$ and $t_2 = "purchase"$ are accompanied by the same surrounding words frequently enough, their representing vectors are optimized for maximum cosine similarity.

$$sim_c = \frac{\sum_{i=1}^{n} v(t_1)_i v(t_2)_i}{\sqrt{\sum_{i=1}^{n} \mathrm{v}(\mathrm{t}_1)_i^{\ 2}} \sqrt{\sum_{i=1}^{n} \mathrm{v}(\mathrm{t}_2)_i^{\ 2}}} \qquad (3)$$

Goldberg and Levy published an in-depth explanation of the math behind Mikolov's approach (Goldberg and Levy, 2014). Pennington et al., developed an additional method called Global Vectors (GloVe) (Pennington et al., 2014). In GloVe, the context windows stretch the entire document but distant words are increasingly lightly weighted.

The fundamental advantage of Word2Vec and GloVe is that they can generate and encode relationships between terms by analyzing a large text file (or the concatenation of many smaller text files). In comparison to LDA term-to-topic vectors, relationships are encoded in offsets while in comparison to ESA, much lower dimensional term-vectors can be produces.

In our scenario, we have text files but no target functions. Additionally, concatenations of text files are easily retrievable from the Internet, for example by downloading Wikipedia.

The aforementioned methods yield vectors representing single words and their semantic relationship to each other. The problem of transforming a sequence of word vectors into a single vector representing this sequence is referred to as *Compositional Distributional Semantics* (CDS). One of many CDS models is the *Basic Additive Model* (BAM, equation 4) (Zanzotto et al., 2010). It sums up and weights the vectors' representing individual terms. In the following equation, the β function yields a scalar weighting for terms t_i, and v provides the term vector for a given term.

$$BAM = \sum_{i=1}^{n} \beta(t_i) v(t_i) \qquad (4)$$

The BAM omits ordering of individual words. Text sequences with a broad vocabulary tend to get higher BAM values if the β function does not compensate for that. In contrast to other CDS models, BAM does not require additional information about the individual terms to work. The easiest implementation for a β function is $\beta = 1/n$, which equally weighs every term occurring in a word sequence. The resulting vectors can then be compared using similarity measures like the cosine similarity (equation 3). The BAM is mathematically equivalent to the centroid of the individual term vectors when using an adequate β function.

Kusner et al. proposed a completely different approach to comparing sequences of word vectors with each other by introducing the *Word Mover's Distance* (WMD) (Kusner et al., 2015). Given two documents d_1 and d_2, the WMD between these documents is defined as the minimum cumulative distance between the words constituting these documents.

If $w_j \in d_1$ also occurs in d_2, it does not add to $WMD(d_1, d_2)$.

If $w_j \notin d_2$, it adds $\min dist_c(v(w_j), v(w_k))$ for $w_k \in d_2$ to $WMD(d_1, d_2)$.

The cosine distance is defined as $dist_c = 1 - sim_c$.

Both the BAM and WMD methods result in a distance measure between two word sequences (i.e., category labels or documents). Of course, word vectors encoding semantic similarity are required.

Mikolov et al. proposed the paragraph2vec method. It builds on the CBOW and skip-gram algorithms, which were extended with an additional document ID that was added to the context windows. Besides generating vectors for each individual word, vectors for documents are generated in the same high dimensional space. When pre-processed vectors are available, BAM and WMD can be used in an online document-by-document fashion. Paragraph2vec can only be computed by analyzing the entire set of documents.

These methods are useful to compute semantic meaning from documents and terms. This allows assessing the similarity between documents or between documents and words without explicitly encoding these or any form of target function. All that is required is a set of texts. None of these methods directly generates a classifier for a predefined set of categories.

In the context Text Categorization in emerging knowledge domains, the work of Nawroth et al., (2018) is highly interesting as it investigates how to recognize emerging Named Entities (Nadeau and Sekine, 2017). Depending on perspective, a named entity can be seen as possible category for TC in any knowledge domain.

2.2 Unsupervised Text Classification

Broadly speaking, one can define unsupervised text classification as methods to assign documents to categories without examples. Three groups of methods can further be distinguished: Methods that yield unlabeled clusters, methods that extract keywords or key phrases from documents, and methods that assign documents to predefined categories. Documents that have the same keywords or key phrases can be regarded as being of the same category. Nevertheless, contemporary approaches usually do not classify documents to predefined categories.

An interesting approach in this direction is Slonim et al.'s work (Slonim et al., 2002) using clustering for unsupervised document classification. They clustered documents so that each one belonged to one cluster. Then the predominant category of the documents in a cluster was used as categorization decision and subsequent performance measurement. That means they knew the correct category per cluster from the beginning and used it as a benchmark for their clustering algorithm. This method requires knowing the target function.

In the context of bootstrapping TC, McCallum and Nigam created a Naïve Bayes classifier that was trained by providing a set of key words for each category (McCallum and Nigam, 1999). They achieved up to 66% accuracy in a 70-leaf taxonomy. Compared to our approach, this actually requires someone to provide key words for each category. Even though this takes vastly less time than manually assigning categories to all documents, it is additional knowledge that might not be available in our bootstrapping scenario.

Ko and Seo build on McCallum and Nigam's results to create a semi-supervised classifier (Ko and Seo, 2009). Their method generates key words for each category by computing which words most often accompanied the category label words within the text. This notion of context is equal to that used in word embedding algorithms. These key words for each category are then used to train a naïve Bayes classifier. The results of the naïve Bayes classifier are subsequently used to train an actual supervised classifier. This multi step approach achieves up to 80% F1 in the Reuters benchmark. Even though this outperforms our approach in F1, it is noteworthy that Ko and Seo's method requires manually compiled external knowledge resources; such as stop word lists and ontologies as for example all adjectives were filtered from the model. In contrast to our approach, this model cannot identify new categories based on a set of documents.

Dai et al. also use a word embedding based approach to construct a classifier (Dai et al., 2017). This classifier discerns whether Twitter tweets are about the flu or not. This way they proposed a method for disease monitoring using social media. The method is related to our NTFC. For every tweet, a random amount of clusters is generated. Then, based on a distance threshold in semantic space, words making up the tweet are assigned to these clusters. Every cluster is then represented using the BAM. In the next step, the distance of every cluster to the term "flu" is measured. If it was under a specified threshold, the cluster is considered flu-related. If one cluster of a tweet is flu-related, the entire tweet is considered flu-related. In their work, they used vectors generated with CBOW based on Google-news articles. The difference in approach is the breaking up of tweets into a random number of clusters. This requires external knowledge in order to correctly parameterize the cluster creation probability and distance between terms and documents. Dai et al., only use very short texts (Twitter Tweets are limited to 140 characters) and one single-word category "flu".

3 DOCUMENT CLASSIFICATION USING WORD EMBEDDING DISTANCES

In the first step, we reduce document content and category labels to lower-case Latin letters. We regard a category as concatenation of individual words constructed from the labels representing the category. This makes categories small documents. Based on these word sequences, we create a word occurrence matrix that maps each term to each document. In order to perform any meaningful categorization, we require a distance measure $dist(c, d)$ between categories and documents.

For single label classification, the classifier is generated by the distance measure and the minimum function: The category with the lowest distance to a document represents the classification for this document.

$$\Phi(c, d) = \min(dist(c, d)) \tag{5}$$

The adopted distance measure always defines the classifier. Multilabel classification can be achieved in two ways: (1) Assigning the n categories with the lowest distance to a document or (2) implementing a distance threshold, which assigns documents to all categories with sufficiently low distances. We decided for the first option because it is an easier to choose parameter when introducing TC. The generic NTFC algorithm can be expressed by the following pseudo-code:

```
NTFC
INPUT: Set of categories $C$
INPUT: Document $d$
INPUT: Assigned categories $n$
INPUT: Distance measure $dist$
OUTPUT: Classifier $\Phi(c,d)$
Start.
01: $\forall c \in C: \Phi(c,d) = F$
02: IF $dist$==TFIDF OR $dist$==BAM
03:  update TFIDF matrix with $d$
04: $minInd[]$ = new $int[n]$
05: $minDist[]$ = new $double[n]$
06: FOR $i = 0; i < n; i++$
07:  FOR $c \in C$
08:       IF $dist(c,d) < minDist[i]$
09:       AND $c \notin minInd[]$
10:            $minDist[i]$= $dist(c,d)$
11:            $minInd[i]$= $c$
12: FOR $i = 0; i < n; i++$
13:  $\Phi(minInd[n], d) = T$
End.
```

The NTFC is a straightforward method that relies on pre-processed word embeddings to generate distance measures between categories and documents. Because these vectors can be generated or downloaded before any classification takes place, high classification efficiency can be achieved. Additionally NTFC can work on individual documents and does not need to analyze all documents to create a classification decision.

We propose three variations of the NTFC, which differ in the utilized distance measures. The first variation is based on TFIDF and omits the usage of information derived from word embeddings. As it is the simplest approach and essentially equal to querying category label terms, we use it as baseline for all other implementations. The TFIDF variation works by limiting the set of possible keywords to terms that occur in category labels. Out of these terms, the top n keywords for a document sorted by TFIDF are computed. A document is subsequently assigned to the category with the most representative label. That means that the highest TFIDF value between a term occurring in a category label and a document is equal to the minimum distance between the category and the document. For example, the MAUI benchmark has the categories *social networks*, *bookmarking*, and *search*. For each document, the TFIDF measure for the individual words making up the labels, (*social*, *networks*, *bookmarking*, and *search*) are computed. If *bookmarking* has the highest TFIDF out of all label words, *bookmarking* is the category assigned to the document. In another example, if *social* has the lowest TFIDF value but *networks* the highest, the document is assigned to the *social networks* category. This approach only considers terms as candidates that literally occur in a document, because the TFIDF value for terms not occurring in a document is 0. Categories with long, multi-word labels have increased chances of representing a specific document. If the documents are classified in a document-by-document fashion, the TFIDF matrix is frequently updated. Alternatively, the TFIDF matrix can be computed for the entire set of documents before categorization allowing skipping steps 2 and 3.

Also utilizing a TFIDF matrix, our second variation is using the BAM model (equation 4) to generate vectors for each document and category. For categories that only have a single term as label, the vector representing this term is loaded from the pre-processed word embeddings. We base our β function on TFIDF using softmax normalization, so that all resulting vectors are of equal length. The

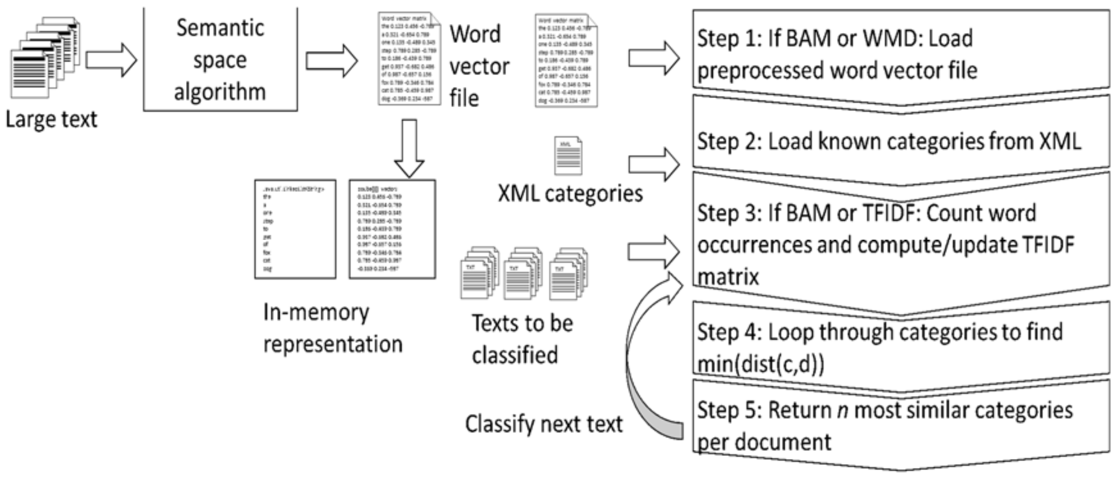
Large text
Semantic space algorithm
Word vector file
In-memory representation
XML categories
Texts to be classified
Classify next text
Step 1: If BAM or WMD: Load preprocessed word vector file
Step 2: Load known categories from XML
Step 3: If BAM or TFIDF: Count word occurrences and compute/update TFIDF matrix
Step 4: Loop through categories to find min(dist(c,d))
Step 5: Return n most similar categories per document

Figure 2: Illustration of the pre-processing process and internal dataflow.

β function depends on the document and term (6).

$$\beta(t_i, d_i) = softmax(TFIDF(t,d)) \quad (6)$$

This variation is inspired by physics as it creates a point similar to the center of mass. The word vector defines the location of the individual mass point. The TFIDF value defines the mass of each point. The same computation is performed for categories, simply substituting d with c. The classifier then uses the cosine distance between documents and categories to find the closest objects and thereby decide which document to assign to which category. For example, the category *graph theory* will have a representing point in the semantic space that is between the vectors for *graph* and *theory*. Each document has a representing point in this semantic vector space. Cosine similarity then detects the minimum distance/maximum similarity documents to the category vector.

The third variation of the NTFC algorithm uses the Word Mover's Distance between categories and documents to detect the closest category. Category labels are much shorter than documents. If all words of a category label occur in a document, the WMD between the category and the document is 0, which ensures that the document is assigned to this category. If they do not all occur in the document, the semantically closest words influence the WMD and overall categorization decision. Reusing the *graph theory* example, a document that contains the words *graph* and *theory* will have WMD = 0. If a document does not contain these words, the cosine distance to the semantically closest words will be added to the WMD between the document and the category. This allows finding the minimum distance/maximum similarity categories for each document.

Besides the utilized distance measure, the NTFC variations depend on the utilized word embedding/ semantic space algorithms, which in turn depend on the texts that are used to compute the word embeddings. Using our model, categories can be added as needed by simply specifying their labels. No examples are needed in order to define a target function. Additionally, the available semantic space can be used to identify new categories by clustering the document vectors and identifying terms central to the document vector clusters.

4 IMPLEMENTATION

We implemented our system in Java. The preprocessing of word vectors is performed with the open source C implementations of the individual algorithms. Their results are stored to a file system and loaded into our system. There, they are stored in a two-dimensional double array where each line corresponds to a specific term and each column to the dimension of the word vectors. We implemented a term list by storing the individual words indices to find the correct vectors quickly.

When not working in the online document-by-document mode, the first step for the BAM and TFIDF baseline implementation is a word occurrence counter. TFIDF values cannot be computed without knowing how often which word occurs in which document. Without TFIDF values, the BAM model cannot be computed. To assess multiple documents at once, we implemented a multi-threaded class, which spawns an arbitrary number of workers. These workers read the text documents from the file system and write the occurrences of individual words to a synchronized, two-dimensional integer array.

The lines of this array correspond to the words the word vector array. The columns correspond to the documents. This information is written to the file system so that it can be re-used by the individual implementations allowing for faster experimentation. We encode categories in simple XML. The TFIDF implementation only computes TFIDF values for terms that actually occur in the category labels. With each ingested document, $\#D(t)$ and the term occurrence file/array is updated. As all information is kept in memory, the individual categorizations can be performed in less than one second for < 2,000 categories on a contemporary Intel i7 CPU. Based on this information and the chosen distance measure, $\min(dist(c, d))$ is computed and used for classification by returning the top n most similar categories per document.

NTFC performs the same operations for each new document. The required time on equal hardware depends on the text used to generate the semantic space. Depending on the complexity of the text examples, the word vector array size can differ. For the setup described in the next chapter, classification time for each category was far below our 30-second limit. Storing a 200-dimensional semantic space as double precision floating point values and combining it with the word occurrence arrays for D and C resulted in memory requirements of about 10 GB for the evaluated scenarios. This is well within the limits of modern computing equipment but requires 64-bit addressing. NTFC can work with arbitrary vectors that encode semantics for words. We decided to use 200-dimensional word embeddings because we lack the computing equipment for 5,500,000-dimensional ESA vectors and LDA based term-to-topic vectors have not been observed to create semantic spaces encoding relationships in offsets.

The WMD approach does not require the TFIDF array and only works with the word occurrence array when accessing all documents at once. To do so, an outer loop runs through the categories comparing their label words with the words making up the document. It performs quicker than the BAM model and stays within our 30-second limit.

5 EVALUATION

We generated 200-dimensional word vectors using CBOW, skip-gram, and GloVe based on a concatenation of Google news articles and the first billion characters of a Wikipedia dump. This resulted in six semantic spaces. We then benchmarked the system's results against the three aforementioned benchmarks. For key-phrase extraction benchmarks, individual key phrases were modeled as categories in our target function Φ'. Even though our classifier doesn't need the target function, we require it for the evaluation. If category c_i is specified to be in document d_j, then $\Phi'(c_i, d_j) = T$. Otherwise $\Phi'(c_i, d_j) = F$.

If our algorithm assigned a document to a correct category according to Φ', then we regarded this as *True Positive* (TP) for the category. If our algorithm assigned a document to an incorrect category, then we regarded this as a *False Positive* (FP). Missing documents for categories were regarded as *False Negatives* (FN). For each category, we computed precision $\pi(c)$, recall $\rho(c)$, and $F1(c)$ using their standard formulae (Sebastiani, 2002).

https://github.com/SirTobiSwobi/NTFCeval contains the raw data of all our experiments. Taking the Reuters single label classification case as an example, our experiments yielded the microaverage results shown in table 1. The bold entries indicate results where an approach outperformed the TFIDF baseline. TFIDF obtains relatively high precision results for unsupervised single label classification. This means that in many cases (MAUI: 62.84%, Reuters: 55.5%, ArXiv: 26.59%), the exact word most representative of a document (out of all category labels according to TFIDF) is part of the correct category. The TFIDF method achieves much higher precision than recall due to only assigning a document to a category if the exact label words of the category have the highest TFIDF for the document. The other methods performed at about half to a third of the precision as the TFIDF approach, whereas almost all experiments produced higher recall than TFIDF because the exact label term does not need to be within the document. BAM performed constantly, relatively independently of the utilized word embedding algorithm and source material yielding F1 results between 11.56% and 18.74%. WMD is more strongly influenced by these parameters, yielding F1 results between 15.63% and 36.35% (the best microaverage results in this experiment).

In the next step, we let our algorithm assign the top two categories to each document. When changing from single label to dual label classification, TFIDF precision is reduced, as many more documents were assigned to incorrect categories. This directly boosted the TFIDF recall to about double that of the single label case. Many

Table 1: Reuters21578 TC benchmark results for one category per document.

Variation	Algorithm	Corpus	Precision	Recall	F1
TFIDF			55.50%	11.59%	19.17%
BAM	CBOW	Google	19.57%	**17.98%**	18.74%
BAM	CBOW	Wiki	19.46%	**17.88%**	18.64%
BAM	Glove	Google	13.80%	**12.69%**	13.22%
BAM	Glove	Wiki	15.00%	**13.79%**	14.37%
BAM	skip-gram	Google	12.07%	11.09%	11.56%
BAM	skip-gram	Wiki	16.85%	**15.48%**	16.14%
WMD	CBOW	Google	35.04%	**32.17%**	**33.54%**
WMD	CBOW	Wiki	33.95%	**31.17%**	**32.50%**
WMD	Glove	Google	35.04%	**32.17%**	**33.54%**
WMD	Glove	Wiki	16.32%	**14.99%**	15.63%
WMD	skip-gram	Google	28.51%	**26.17%**	**27.29%**
WMD	skip-gram	Wiki	37.98%	**34.87%**	**36.35%**

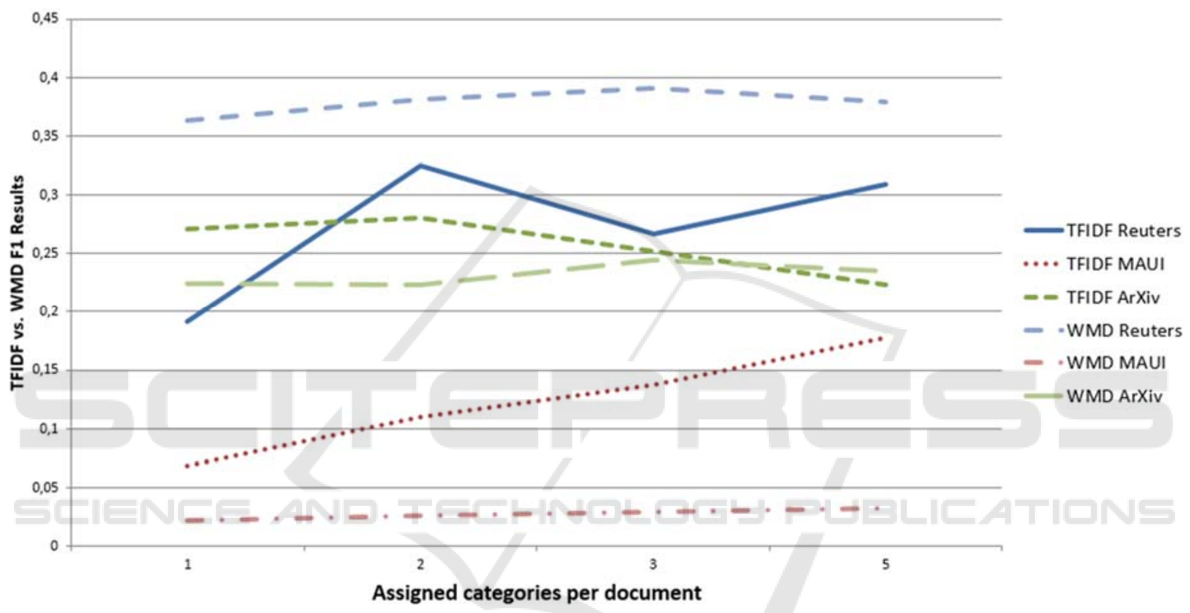

Figure 3: F1 comparison for TFIDF and WMD for different category amounts.

documents that had more than one category in their target function could now get a second correctly assigned category. Overall, TFIDF outperformed all BAM implementations. Interestingly, the WMD implementations now had better precision than TFIDF when no GloVe-based vectors were used. This means that even though TFIDF yielded the best single label microaverage precision, WMD algorithms yielded higher precision and better F1 than TFIDF for non-GloVe-based vectors when two categories per document were assigned. In further experiments, we increased the number of categorizations per documents to three and five.

The previously shown trends of decreased precision for increased recall continued. When moving from two to three categories per document, TFIDF precision drops by about 7% while all WMD implementations have a recall decrease of less than 1%. In the Reuters top 5 experiment, TFIDF had a microaverage precision of 18.8% with a recall of 86.41% resulting in 30.89% F1. In the same experiment, non-GloVe-based WMD implementations yielded higher precision (up to 33.75%) and subsequently F1 values (up to 41,62%). In the Reuters top 3 and top 5 experiments, CBOW/Google Vectors achieved 40.36% and 41.62% F1 with skip-gram/Wikipedia vectors a close second with 39.06% and 37.92% respectively. Over all Reuters and ArXiv experiments, the WMD implementation based on skip-gram/Wikipedia-generated word vectors created the best results. In the MAUI experiments, the BAM implementations worked better than WMD. However, in MAUI TFIDF outperformed WMD and BAM by a large margin. In general, the more categories assigned to each document, the higher the recall. TFIDF tends to reduce in precision with increasing categories while other implementations, especially the WMD, are more stable. The three benchmarks have interesting relationships between the number of documents and categories. Where for the Reuters Benchmark |D|>>|C|, for the MAUI Benchmark, |D|<<|C|. For ArXiv, |D|>|C|. This can explain the results of the NTFC as it seems to perform better the smaller the set of categories is compared to the amount of documents.

6 CONCLUSIONS

We have shown three methods of constructing an unsupervised text classifier. This gives us three answers to our first research question of how an unsupervised classifier can be created that labels documents efficiently? In single label classification, different distance implementations and word embeddings optimize different measures. 55.5% of microaverage precision can be achieved by using the TFIDF approach. However the TFIDF approach yields relatively low recall. Regarding recall and F1, the WMD method performed best. The best implementation of WMD, skip-gram, and Wikipedia source material achieved 36.35% of microaverage F1. Moving to multilabel classification, the TFIDF implementation can achieve higher recall values while simultaneously losing precision. This effect is far less for the WMD implementation. This answers our second research question of how well a classifier without a target function and minimal additional information can perform in terms of effectiveness.

To answer our third research question of what influences classifier performance: The amount of categorizations performed per document strongly influences classifier performance. The closer the amount is to the average amount of categories per document of the target function, the better the results. The relationship in size between the set of categories and the set of documents also influences classifier performance. The less categories there are, the better the performance of the NTFC. The classifier performance also directly depends on the word vectors that are used. We found that using the skip-gram algorithm on Wikipedia produced the best vectors in single label classification. For multi-label categorization, word vectors generated using CBOW on Google news material achieved slightly better precision, recall, and F1 than those in second place which were generated using skip-gram on Wikipedia. The better performance of WMD over BAM can be explained with a stronger dependence on word embeddings, because no TFIDF measures are taken into consideration. Wikipedia based vectors allow to capture more exotic terms that do not occur in the vocabulary extracted from Google news. This leads to the usually better performance of Wikipedia based vectors in the ArXiv benchmark, which consisted out of scientific documents containing a more complex vocabulary than the other benchmarks.

Regarding computation time, all approaches need to count how often which word occurs in which document. Having performed that task, TFIDF requires computation time based on the size of the vocabulary T per category word per document. This, as well as finding the highest TFIDF value per document, costs $O(|C| * |D| * |T|)$. Performed in an online fashion, this usually requires less than a second per document in the utilized benchmarks. For BAM, the TFIDF values are required to create the β function. The results then need to be multiplied with the individual word vectors increasing computational complexity by the factor of their dimension to $O(|C| * |D| * |T| * |v()|)$. This dominates computing the cosine similarity between all documents and categories, as it requires looping through the dimensions. The WMD implementation also loops through all categories and documents. There it computes the cosine similarity between all words making up the document to those making up the category. This computation also costs $O(|C| * |D| * |T| * |v()|)$.

We extracted the possible terms from the word vector file. The size of the vocabulary depends on its source (Wikipedia: 281,317 words; Google news: 71,291 words). When using 200-dimensional word vectors and a Wikipedia-based corpus, WMD and BAM require less than 10 seconds per categorization. In a Google news-based corpus, the required time is less than 3 seconds on a contemporary i7 CPU, which is within our 30-second limit. This distinguishes our approach from available other state-of-the-art approaches: The system can work online and does not require assessing the whole corpus of documents for every document ingestion. Additionally, categories can be added when needed without requiring any target function. Besides the word embeddings, no additional information than the documents and categories is required.

As mentioned in the introduction, our intended information system presents the computed categories after document ingest to the user, who then affirms or corrects the categorizations. This interactive process then builds a target function over time.

When there is a sufficiently large target function, the system can switch to a classic supervised classification algorithm like SVM to mimic the users' document classification behavior. The decision of which system to use can be answered by comparing the supervised learning classifier results to the NTFC results in the background. As soon as the classifier can outperform the NTFC, the system can switch to the regular classifier. Alternatively, the supervised learning classifier and the NTFC can form a classifier committee. Because the NTFC cannot overfit, this can prevent the regular classifier from overfitting. The model can also be used to extract potential new categories from an existing text corpus. Documents can be clustered in semantic space and cluster means in can be computed. These cluster means can be used to find terms most descriptive for the cluster. Clusters can then be regarded as categories while the words closest to the cluster mean can be used as category labels. The nature of semantic spaces allows assessing the relationships between the clusters. For example hyponymy- and hypernymy relationships between the labels of different categories. We intent to investigate this further in future works as well as extending the NTFC to work with multiple clusters for text representation as proposed by Dai et al., (2017). Different to Dai et al.'s work, we will try to minimize the necessity of external knowledge to parameterize the solution.

REFERENCES

Bellman, R. (1961). *Adaptive Control Processes. A Guided Tour*, Princeton University Press, USA.

Blei, D.M., Ng, A.Y., Jordan, M. I. (2003) Latent Dirichlet Allocation, Journal of Machine Learning Research, vol. 3, pp. 993-1022, doi:10.1162/jmlr.2003.3.4-5.993.

Busse J., Humm, B., Lübbert, C., Moelter, F., Reibold, A., Rewald, M., Schlüter, V., Seiler, B., Tegtmeier, E., Zeh, T. (2015). Actually, What Does "Ontology" Mean? A Term Coined by Philosophy in the Light of Different Scientific Disciplines. In: ournal of Computing and Information Technology – CIT 23, pp. 29-41 doi:10.2498/cit.1002508.

Cho, K., Kim, J. (1997). Automatic text categorization on hierarchical category structure by using ICF (inverse category frequency) weighting. In: *Proceedings of KISS conference*. pp. 507-510.

Cornell University Library (2016) *arXiv.org* [online] Available at: https://arxiv.org [Accessed 15 Dec. 2016]

Dai, X., Bikdash, M., Meyer, M. (2017).From social media to public health surveillance: Word embedding based clustering method for twitter classification. In *Proceedings SoutheastCon,* pp. 1-7, doi:10.1109/SECON.2017.7925400.

DFG (2016) *Schwerpunktprogramm "Robust Argumentation Machines" (SPP 1999), 27 June.* [online] Available at: http://www.dfg.de/foerderung/info_wissenschaft/2016/info_wissenschaft_16_38/index.html [Accessed 6 Mar. 2018]

Dumais, S. T. (2005). Latent Semantic Analysis. In: *Annual Review of Information Science and Technology*, vol. 38, pp. 188-230 DOI: 10.1002/aris.1440380105.

Egozi, O, Markovitch, S., Gabrilovich, E. (2011). Concept-Based Information Retrieval using Explicit Semantic Analysis. In *ACM Transactions on Information Systems*, vol. 29, pp. 8:1-8:34. doi: 10.1145/1961209.1961211.

Gabrilovich, E., Markovitch, S., (2006). Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In: *AAAI* Vol. 6, pp. 1301-1306.

Goldberg, Y, Levy, O. (2014). *word2vec Explained, Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method* [online] Available at: https://arxiv.org/pdf/1402.3722v1.pdf [Accessed 25 Jan. 2017]

Ko, Y., Seo, J. (2009). Text classification from unlabeled documents with bootstrapping and feature projection techniques. In *Journal of Information Processing and Management 45,* pp. 70-83.

Kusner, M. J., Sun, Y., Kolkin, N., Weinberger, K. Q. (2015). From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning,* Lille, France.

Lewis, D. (2004) *The Reuters-21578 text categorization benchmark.* [online] Available at: http://www.daviddlewis.com/resources/testcollections/reuters21578/reuters21578.tar.gz [Accessed 02 Aug. 2017]

McCallum, A., Nigam, K. (1999). Text Classification by Bootstrapping with Keywords, EM and Shrinkage. In: *Workshop On Unsupervised Learning In Natural Language Processing,* pp. 52-58.

Medelyan, O., Frank, E., Witten, I. H., (2009). Human-competitive tagging using automatic keyphrase extraction. In *Conference on Empirical Methods in Natural Language Processing EMNLP 09.* Singapore. pp. 1318-1327.

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient Estimation of Word Representation in Vector Space. In: *Proceedings of Workshop at ICLR.* [online] Available at: http://arxiv.org/pdf/1301.3781.pdf [Accessed 29 Dec. 2015]

Mikolov, T. (2013) Word2Vec C Code [online] Available at: https://code.google.com/archive/p/word2vec/source/default/source [accessed 05 Dec. 2015]

Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012). *Foundations of Machine Learning,* MIT Press, Cambridge, Massachusetts, USA.

Nadeau, D., Sekine, S. (2007). A survey of named entity recognition and classification, *Lingvisticae Investigationes*, 30(1), pp. 3-26.

Nawroth, C., Engel, F., Hemmje, M., Eljasik-Swoboda, T. (2018). Emerging Named Entity Recognition for Clinical Argumentation Support. In *Submitted to Proceedings of DATA 2018*.

Pennington, J., Socher, R., Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing*, pp. 1532-1543.

Salton, G., McGill, M. (1983). *Introduction to Modern Information Retrieval*, McGraw-Hill.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. In *ACM Computing Surveys.* Vol 34, pp. 1-47.

Slonim, N., Friedman, N., Tishby, N. (2002). Unsupervised document classification using sequential information maximization..In *Proceedings of the 25th ACM SIGIR conference on Research and development in in-formation retrieval,* doi:10.1145/564376.564401.

Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., Manandhar, S., (2010). Estimating linear models for compositional distributed semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics,* pp. 1263-1271.