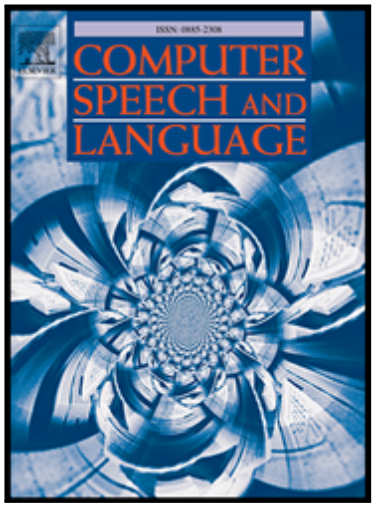
Journal Pre-proof



tax2vec: Constructing Interpretable Features from Taxonomies for Short Text Classification

Blaž Škrlj, Matej Martinc, Jan Kralj, Nada Lavrač, Senja Pollak

PII: S0885-2308(20)30037-1
DOI: https://doi.org/10.1016/j.csl.2020.101104
Reference: YCSLA 101104

To appear in: *Computer Speech & Language*

Received date: 31 January 2019
Revised date: 19 April 2020
Accepted date: 21 April 2020

Please cite this article as: Blaž Škrlj, Matej Martinc, Jan Kralj, Nada Lavrač, Senja Pollak, tax2vec: Constructing Interpretable Features from Taxonomies for Short Text Classification, *Computer Speech & Language* (2020), doi: https://doi.org/10.1016/j.csl.2020.101104

tax2vec: Constructing Interpretable Features from Taxonomies for Short Text Classification

Blaž Škrlj[b,a], Matej Martinc[b,a], Jan Kralj[a], Nada Lavrač[a,c], *Senja Pollak[a]

[a] *Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
[b] *Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia*
[c] *University of Nova Gorica, Glavni trg 8, 5271 Vipava, Slovenia*

Abstract

The use of background knowledge is largely unexploited in text classification tasks. This paper explores word taxonomies as means for constructing new semantic features, which may improve the performance and robustness of the learned classifiers. We propose tax2vec, a parallel algorithm for constructing taxonomy-based features, and demonstrate its use on six short text classification problems: prediction of gender, personality type, age, news topics, drug side effects and drug effectiveness. The constructed semantic features, in combination with fast linear classifiers, tested against strong baselines such as hierarchical attention neural networks, achieves comparable classification results on short text documents. The algorithm's performance is also tested in a few-shot learning setting, indicating that the inclusion of semantic features can improve the performance in data-scarce situations. The tax2vec capability to extract corpus-specific semantic keywords is also demonstrated. Finally, we investigate the semantic space of potential features, where we observe a similarity with the well known Zipf's law.

Keywords: taxonomies, vectorization, text classification, short documents, feature construction, semantic enrichment

*Corresponding author

1. Introduction

In text mining, document classification refers to the task of classifying a given text document into one or more categories based on its content [1]. Given an input set of labeled text documents, a text classifier is expected to learn to associate the patterns appearing in the documents to the document labels. Deep learning approaches [2] have recently become a standard in natural language-related learning tasks, demonstrating good performance on a variety of different classification tasks, including sentiment analysis of tweets [3] and news categorization [4]. Despite achieving state-of-the-art performance on many tasks, deep learning is not yet optimized for situations, where the number of documents in the training set is low or when the documents contain very little text [5].

Semantic data mining denotes a data mining approach where domain ontologies are used as background knowledge in the data mining process [6]. Semantic data mining approaches have been successfully applied to association rule learning [7], semantic subgroup discovery [8, 9], data visualization [10] and text classification [11]. Provision of semantic information allows the learner to use features on a higher semantic level, possibly enabling better data generalizations. The semantic information is commonly represented as relational data in the form of taxonomies or 3 ontologies. Development of approaches that leverage such information remains a lively research topic in several fields, including biology [12, 13], sociology [14] and natural language processing [15].

This paper contributes to semantic data mining by using word taxonomies as means for semantic enrichment by constructing new features, with the goal to improve the performance and robustness of the learned classifiers. In particular, it addresses classification of short or incomplete documents, which is useful in a large variety of text classification tasks. Short text is characterized by shortness in the text length, and sparsity in the terms presented, which results in the difficulty in managing and analyzing them based on the bag-of-words representation only. Short texts can be found everywhere, such as search snippets, product reviews and similar [16]. For example, in author profiling, the task is

to recognize the author's characteristics such as age or gender [17], based on a collection of author's text samples. Here, the effect of data size is known to be an important factor, influencing classification performance [18]. A frequent text type for this task are tweets, where a collection of tweets from the same author is considered a single document, to which a label must be assigned. The fewer instances (tweets) per user we need, the more powerful and useful the approach. Learning from only a handful of tweets can lead to preliminary detection of bots in social networks, and is hence of practical importance [19, 20]. In a similar way, this holds true for nearly any kind of text classification task. For example, for classifying news into a specific topic, using only snippets or titles may be preferred due to non-availability of entire news texts or for increasing the processing speed. Moreover, in biomedical applications, Grässer et al. [21] tried to predict drug's side effects and effectiveness from patients' short commentaries, while Boyce et al. [22] investigated the use of short user comments to assess drug-drug interactions.

It has been demonstrated that deep neural networks in general need a large amount of information in order to learn complex classifiers, i.e. they require a large training set of documents. For example, the recently introduced BERT neural network architecture [2] consisting of many hidden layers was trained on the whole Wikipedia. It was also shown that the state-of-the-art models do not perform well when incomplete (or scarce) information is used as input [23]. On the other hand, promising results regarding zero-shot [24] and few-shot [25] learning were recently achieved.

This paper proposes a novel approach named *tax2vec*, where semantic information available in taxonomies is used to construct semantic features that can improve classification performance on short texts. In the proposed approach, features are constructed automatically and remain *interpretable*. We believe that tax2vec could help explore and understand how external semantic information can be incorporated into existing (black-box) machine learning models, as well as help to explain what is being learned.

This work is structured as follows. Following the theoretical preliminaries

and the related work necessary to understand how semantic background knowledge can be used in learning, presented in Section 2, we continue with the description of the proposed tax2vec methodology in Section 3. In Section 4, we describe the experimental setting used to test the methodology. In Section 5, we present the results of experiments, including the evaluation of the qualitative properties of features constructed using tax2vec, and extensive classification benchmark tests. Section 6 discusses the properties of the resulting semantic space and the explainability of the proposed tax2vec algorithm. Implementation and availability of tax2vec is addressed in Section 7. The paper concludes with a summary and prospects for further work in Section 8. For completeness, Appendix A includes a detailed description of the Personalized PageRank algorithm, while Appendix B presents an example segmentation of news articles into paragraphs, forming short documents of interest for this study. Finally, Appendix C contains an additional ablation study regarding the impact of feature numbers on the classifier performance.

2. Background and related work

In this section we present the theoretical preliminaries and some related work, which served as the basis for the proposed tax2vec approach. We begin by explaining different levels of semantic context and the rationale behind the proposed approach.

2.1. Semantic context

Document classification is highly dependent on *document representation*. In simple bag-of-words representations, the frequency (or a similar weight such as term frequency-inverse document frequency—tf-idf) of each word or n-gram is considered as a separate feature. More advanced representations group words with similar meaning together. Such approaches include Latent Semantic Analysis [26], Latent Dirichlet Allocation [27], and more recently word embeddings [28]. It has been previously demonstrated that context-aware algorithms signifi-

cantly outperform the naive learning approaches [29]. We refer to such semantic context as the *first-level context*.

Second-level context can be introduced by incorporating *background knowledge* (e.g., ontologies) into a learning task, which can lead to improved interpretability and performance of classifiers, learned e.g., by rule learning [8] or random forests [30]. In text mining, Elhadad *et al.* [31] present an ontology-based web document classifier, while Kaur *et al.* [32] propose a clustering-based algorithm for document classification that also benefits from knowledge stored in the underlying ontologies. Cagliero and Garza [29] present a custom classification algorithm that can leverage taxonomies and demonstrate on a case study of geospatial data that such information can be used to improve the learner's classification performance. Use of hypernym-based features for classification tasks has been considered previously. For example, hypernym-based features were used in rule learning by the Ripper rule learning algorithm [11]. Moreover, it was also demonstrated that the use of hypernym-based features constructed from WordNet significantly impacts the classifier performance [33].

2.2. Feature construction and selection

When unstructured data is used as input, it is common to explore the options of feature construction. Even though recently introduced deep neural network based approaches operate on simple word indices (or byte-pair encoded tokens) and thus eliminate the need for manual construction of features, such alternatives are not necessarily the optimal approach when vectorizing the background knowledge in the form of taxonomies or ontologies. Features obtained by training a neural network are inherently non-symbolic and as such do not present any added value to the developer's understanding of the (possible) causal mechanisms underlying the learned classifier [34, 35]. In contrast, understanding the semantic background of a classifier's decision can shed light on previously not observed second-level context vital to the success of learning, rendering otherwise incomprehensible models easier to understand.

Definition 1 (Feature construction)**.** *Given an unstructured input consisting*

of n documents, a feature construction algorithm outputs a matrix $F \in \mathbb{R}^{n \times \alpha}$, where α denotes the predefined number of features to be constructed.

In practical applications, features are constructed from various data sources, including texts [36], graphs [37, 38], audio recordings and similar data [39]. With the increasing computational power at one's disposal, automated feature construction methods are becoming prevalent. Here, the idea is that given some criterion, the feature constructor outputs a set of features selected according to the criterion. For example, the tf-idf feature construction algorithm, applied to a given document corpus, can automatically construct hundreds of thousands of n-gram features in a matter of minutes on an average of-the-shelf laptop.

Many approaches can thus output too many features to be processed in a reasonable time, and can introduce additional noise, which renders the task of learning even harder. To solve this problem, one of the known solutions is *feature selection*.

Definition 2 (Feature selection)**.** *Let $F \in \mathbb{R}^{n \times \alpha}$ represent the feature matrix (as defined above), obtained during automated feature construction. A feature selection algorithm transforms matrix F to a matrix $F' \in \mathbb{R}^{n \times d}$, where d represents the number of desired features after feature selection.*

Feature selection thus filters out the (unnecessary) features, with the aim of yielding a compact, information-rich representation of the unstructured input. There exist many approaches to feature selection. They can be based on the individual feature's information content, correlation, significance etc. [40]. Feature selection is, for example, relevant in biological data sets, where only a handful of the key gene markers are of interest, and can be identified by assessing the impact of individual features on the target space [41].

2.3. Learning from graphs and relational information

In this section we briefly discuss the works that influenced the development of the proposed approach. One of the most elegant ways to learn from graphs is by transforming them into propositional tables, which are a suitable input

for many down-stream learning algorithms. Recent attempts to vectorization of graphs include the node2vec [42] algorithm for constructing features from homogeneous networks; its extension metapath2vec [43] for heterogeneous networks; its symbolic version SGE [38]; the mol2vec [44] vectorization algorithm for molecular data; the struc2vec [45] graph vectorization algorithm based on homophily relations between nodes, and more. All these approaches (apart from SGE) are sub-symbolic, as the obtained vectorized information (embeddings) are not interpretable. Similarly, recently introduced graph-convolutional neural networks also yield local node embeddings, which take node feature vectors into account [46, 47].

In parallel to graph-based vectorization, approaches which tackle the problem of learning from relational databases have also been developed. Symbolic (interpretable) approaches for this vectorization task, known under the term propositionalization, include RSD [48], a rule-based algorithm which constructs relational features; and wordification [49], an approach for unfolding relational databases into bag-of-words representations. The approach, described in the following sections, relies on some of the key ideas initially introduced in the mentioned works on propositionalization, as taxonomies are inherently relational data structures.

3. The tax2vec approach

In this section we outline the proposed tax2vec approach. We begin with a general description of classification from short texts, followed by the key features of tax2vec, which offer solutions to some of the currently not well explored issues in text mining.

3.1. The rationale behind tax2vec

In general text classification tasks, deep learning approaches have outperformed other classifiers [2]. However, in classification tasks involving short documents (tweets, opinions, etc.), particularly where the number of instances is

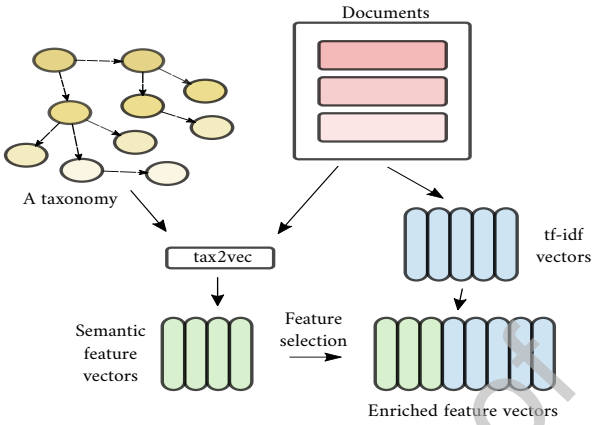


Figure 1: Schematic representation of tax2vec, combined with standard tf-idf representation of documents. Note that darker nodes in the taxonomy represent more general terms.

low, deep learners are still outperformed by simpler classifiers, such as SVMs [50]. This observation was a motivation for the development of the tax2vec algorithm, proposed in this paper. Compared to non-symbolic node vectorization algorithms discussed in the previous section, tax2vec uses hypernyms as potential features directly and thus makes the process of feature construction and selection possible without the loss of classifier's *interpretability*.

We present the proposed tax2vec algorithm for semantic feature vector construction that can be used to enrich the feature vectors constructed by the established text processing methods such as tf-idf. The tax2vec algorithm takes as input a labeled or unlabeled corpus of n documents and a word taxonomy. It outputs a matrix of *semantic feature vectors* in which each row represents a semantics-based vector representation of one input document. Example use of tax2vec in a common language processing pipeline is shown in Figure 1. Note that the obtained semantic feature vectors serve as additional features in the final, vectorized representation of a given corpus.

Let us first explore how parts of the WordNet taxonomy [51] related to the training corpus can be used for the construction of novel features, as such background knowledge can be applied in virtually every English text-based learning

setting, as well as for many other languages [52].

3.2. Deriving semantic features

The tax2vec approach implements a two-step semantic feature construction process. First, a document-specific taxonomy is constructed, then a term-weighting scheme is used for feature construction.

3.2.1. Document-based and corpus-based taxonomy construction

In the first step of the tax2vec algorithm, a corpus-based taxonomy is constructed from the input document corpus. In this section we describe how the words from individual documents of a corpus are mapped to terms of the WordNet taxonomy to construct a *document-based taxonomy* by focusing on semantic structures, derived exclusively from the *hypernymy* relation between words. Individual document-based taxonomies are then merged into a joint *corpus-based taxonomy*.

When constructing a document-based taxonomy, each word is mapped to the hypernym WordNet taxonomy. This results in a tree-like structure, which spans from individual words to higher-order semantic concepts. For example, given the word monkey, one of its mappings in the WordNet hypernym taxonomy is the term *mammal*, which can be further mapped to e.g., *animal* etc., eventually reaching the most general term, i.e. *entity*.

In order to construct the mapping, the first problem to be solved is *word-sense disambiguation*. For example, the word *bank* has two different meanings, when considered in the following two sentences:

River bank was enforced.	National bank was robbed.

There are many approaches to word-sense disambiguation (WSD). We refer the reader to [53] for a detailed overview of the WSD methodology.

In tax2vec, we use Lesk [54], the gold standard WSD algorithm, to map each disambiguated word to the corresponding term in the WordNet taxonomy. The identified term is then associated with a path in the WordNet taxonomy

leading from the given term to the root of the taxonomy. Example hypernym path (with WordNet-style notation), extracted for word "astatine", is shown in Figure 2.

$Synset('entity.n.01')$
$\rightarrow Synset('abstraction.n.06')$
$\rightarrow Synset('relation.n.01')$
$\rightarrow Synset('part.n.01')$
$\rightarrow Synset('substance.n.01')$
$\rightarrow Synset('chemical\_element.n.01')$
$\rightarrow Synset('astatine.n.01')$

Figure 2: Example hypernym path extracted for word "astatine", where the $\rightarrow$ corresponds to the "hypernym of" relation (the majority of hypernym paths end with the "entity" term, as it represents one of the most general objects in the taxonomy).

By finding a hypernym path to the root of the taxonomy for all words in the input document, a *document-based taxonomy* is constructed, which consists of all hypernyms of all words in the document. After constructing the document-based taxonomy for all the documents in the corpus, the taxonomies are joined into a *corpus-based taxonomy*.

Note that processing each document and constructing the document-based taxonomy is entirely independent from other documents, allowing us to process the documents in parallel and join the results only when constructing the joint corpus-based taxonomy.

3.2.2. Semantic feature construction

During the construction of a document-based taxonomy, document-level term counts are calculated for each term. For each word t and document D, we count the number $f_{t,D}$ of times the word or one of its hypernyms appeared in a given document D.

The obtained counts can be used for feature construction directly: each term t from the corpus-based taxonomy is associated with a feature, and a document-

level term count is used as the feature value. The current implementation of tax2vec weights the feature values using the double normalization tf-idf metric. For term t, document D and user-selected normalization factor K, feature value tf-idf(t,D,K) is calculated as follows [55]:

$$\text{tf-idf}(t, D, K) = \underbrace{\left(K + (1-K)\frac{f_{t,D}}{\max_{\{t' \in D\}} f_{t',D}}\right)}_{\text{Weighted term frequency}} \cdot \underbrace{\log\left(\frac{N}{n_t}\right)}_{\substack{\text{Inverse} \\ \text{document frequency}}} \tag{1}$$

where $f_{t,D}$ is the term frequency, normalized by $\max_{\{t' \in D\}} f(t', D)$, which corresponds to the raw count of the most common hypernym of words in the document; value N represents the total number of documents in the corpus, n_t denotes the number of document-based taxonomies the hypernym appears in (i.e. the number of documents that contain a hyponym of t). Note that the term frequencies are normalized with respect to the most frequently occurring term to prevent a bias towards longer documents. In the experiments the normalization constant K was set to 0.5.

3.3. Feature selection

The problem with the above presented approach is that all hypernyms from the corpus-based taxonomy are considered, and therefore, the number of columns in the feature matrix can grow to tens of thousands of terms. Including all these terms in the learning process introduces unnecessary noise, and unnecessarily increases the spatial complexity. This leads to the need of feature selection (see Definition 2 in Section 2.2) to reduce the number of features to a user-defined number (a free parameter specified as part of the input). We next describe the scoring functions of feature selection approaches considered in this work.

As part of tax2vec, we implemented both supervised (Mutual Information - MI and Personalized PageRank - PPR), as well as unsupervised (Betweenness centrality - BC and term count-based selection) feature selection methods, discussed below. Note that the feature selection process is conducted *exclusively* on the semantic space (i.e. on the mapped WordNet terms).

Feature selection by term counts. Intuitively, the rarest terms are the most document-specific and could provide additional information to the classifier. This is addressed in tax2vec by the simplest heuristic, used in the algorithm: a term-count based heuristic that simply takes overall counts of all hypernyms in the corpus-based taxonomy, sorts them in ascending order according to their frequency of occurrence and takes the top d.

Feature selection using term betweenness centrality. As the constructed corpus-specific taxonomy is not necessarily the same as the WordNet taxonomy, the graph-theoretic properties of individual terms within the corpus-based taxonomy could provide a reasonable estimate of a term's importance. The proposed tax2vec implements the betweenness centrality (BC) [56] measure of individual terms as the scoring measure. The betweenness centrality is defined as:

$$BC(t) = \sum_{u \neq v \neq t} \frac{\sigma_{uv}(t)}{\sigma_{uv}}; \tag{2}$$

where σ_{uv} corresponds to the number of shortest paths (see Figure 3) between nodes u and v, and $\sigma_{uv}(t)$ corresponds to the number of paths that pass through term (node) t. Intuitively, betweenness measures the t's importance in the corpus-based taxonomy. Here, the terms are sorted in a descending order according to their betweenness centrality, and again, the top d terms are used for learning.

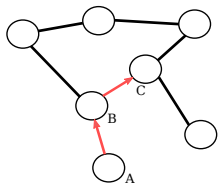


Figure 3: An example shortest path. The path colored red represents the smallest number of edges needed to reach node C from node A.

Feature selection using mutual information. The third heuristic, mutual information (MI) [57], aims to exploit the information from the labels,

assigned to the documents used for training. The MI between two random discrete variables represented as vectors F_i and Y (i.e. the i-th hypernym feature and a target binary class) is defined as:

$$MI(F_i, Y) = \sum_{x,y \in \{0,1\}} p(F_i = x, Y = y) \cdot \log_2 \left(\frac{p(F_i = x, Y = y)}{p(F_i = x) \cdot p(Y = y)} \right) \quad (3)$$

where $p(F_i = x)$ and $p(Y = y)$ correspond to marginal distributions of the joint probability distribution of F_i and Y. Note that for this step, tax2vec uses the binary feature representation, where the tf-idf features are rounded to the closest integer value (either 0 or 1). This way, only well represented features are taken into account. Further, tax2vec uses one-hot encodings of target classes, meaning that each target class vector consists exclusively of zeros and ones. For *each* of the target classes, tax2vec computes the mutual information (MI) between *all* hypernym features (i.e. matrix X) and a given class. Hence, for each target class, a vector of mutual information scores is obtained, corresponding to MI between individual hypernym features and a given target class.

Finally, tax2vec sums the MI scores obtained for each target class to obtain the final vector, which is then sorted in descending order. The first d hypernym features are used for learning. At this point tax2vec yields the selected features as a sparse matrix, maintaining the spatial complexity amounting to the number of float-valued non-zero entries.

Personalized PageRank-based hypernym ranking. Advances by Kralj *et al.* [58, 59] in learning using extensive background knowledge for rule induction explored the use of Personalized PageRank (PPR) algorithm for node subset selection in semantic search space exploration. In tax2vec, we use the same idea to prioritize (score) hypernyms in the corpus-based taxonomy. In this section, we first briefly describe the Personalized PageRank algorithm and then describe how it is applied in tax2vec.

The PPR algorithm takes as an input a network and a set of starting nodes in the network and returns a vector assigning a score to each node

in the input network. The scores of nodes are calculated as the stationary distribution of the positions of a random walker that starts its walk on one of the starting nodes and, in each step, either randomly jumps from a node to one of its neighbors (with probability p, set to 0.85 in our experiments) or jumps back to one of the starting nodes (with probability $1-p$). Detailed description of the PPR used in tax2vec is given in Appendix A. The PPR algorithm is used in tax2vec as follows:

1. Identify a set of hypernyms in the corpus-based taxonomy, to which the words in the input corpus map to in the first step of tax2vec (described in Section 3.2.1).
2. Run the PPR algorithm on the corpus-based taxonomy, using the hypernyms identified in step 1 as the starting set.
3. Use the top d best ranked hypernyms as candidate features.

Note that this heuristics offers *global* node ranks with respect to the corpus used.

3.4. The tax2vec algorithm

All the aforementioned steps form the basis of tax2vec, outlined in Algorithm 1. First, tax2vec iterates through the given labeled document corpus in parallel (lines 3–7). For each document, *MaptoTaxonomy* method identifies a set of disambiguated words and determines their corresponding terms in taxonomy $\mathfrak{T}$ (i.e. WordNet) using method m (i.e. Lesk). Term counts are stored for later use (*storeTermCounts*), and the taxonomy, derived from a given document (*doc*) is added to the corpus taxonomy $\mathfrak{T}_{\text{CORPUS}}$. Once traversed, the terms present in $\mathfrak{T}_{\text{CORPUS}}$ represent potential *features*. Term counts, stored for each document are aggregated into vectors of size n, where n is the number of documents in the corpus. The result of this step is a real-valued, sparse matrix (vecSpace), where columns represent all possible terms from $\mathfrak{T}_{\text{CORPUS}}$. In the following step, feature selection is conducted. Here, graph-based methods (e.g., BC and PPR) identify top d terms based on $\mathfrak{T}_{\text{CORPUS}}$'s properties (lines 9–12),

Algorithm 1: tax2vec

Data: Training set documents D, training document labels Y_{tr}, WordNet taxonomy $\mathfrak{T}$, word-to-taxonomy mapping m, feature selection heuristic h, number of selected features d

1 $\mathfrak{T}_{\text{CORPUS}} \leftarrow$ empty structure;
2 termCounts $\leftarrow$ empty structure;
3 **for** *doc* $\in D$ *(in parallel)* **do**
4 $\quad \mathfrak{T}_{\text{DOCUMENT}} \leftarrow \text{MaptoTaxonomy}(doc, \mathfrak{T}, m)$;
5 $\quad$ Add storeTermCounts($\mathfrak{T}_{\text{DOCUMENT}}$) to termCounts;
6 $\quad$ Add $\mathfrak{T}_{\text{DOCUMENT}}$ to $\mathfrak{T}_{\text{CORPUS}}$;
7 **end**
8 vecSpace $\leftarrow$ tf-idf(constructTfVectors(D, $\mathfrak{T}_{\text{CORPUS}}$,termCounts));
9 **if** *h is graph-based* **then**
10 $\quad$ topTerms $\leftarrow$ selectFeatures(h, $\mathfrak{T}_{\text{CORPUS}}$, d, optional Y_{tr});
11 $\quad$ selectedFeatures $\leftarrow$ select topTerms from vecSpace;
12 **end**
13 **else**
14 $\quad$ selectedFeatures$\leftarrow$ selectFeaturesDirectly(h, vecSpace,d ,Y_{tr});
15 **end**
16 **return** selectedFeatures;

Result: d new feature vectors in sparse vector format.

and non-graph methods (e.g., MI) is used directly on the sparse matrix to select which d features are the most relevant (lines 13–15). Finally, $selectedFeatures$, a matrix of selected semantic features is returned.

Note that in practice, tax2vec must also store the inverse document frequencies in order to generate features for unseen documents. We omit the description of this step for readability purposes.

3.5. Handling noise

Numerous data sets, including contemporary social media data sets, can be noisy and as such hard to handle by a learning system. We next discuss how distinct parts of tax2vec potentially handle noise in the data, including typos, incomplete and missing words and uncommon characters.

During the initial step of the semantic space construction, tax2vec conducts document-level word disambiguation in order to semantically characterize a given token (word). During this step, any tokens that are not present in the taxonomy will be ignored. Further, as word disambiguation requires a certain word window to operate, this hyperparameter can be used to control the size of context considered by tax2vec. In this work, however, we did not explicitly address the problem of invalid tokens in a given token's neighborhood, yet observed that small window sizes (two and three) offered reasonably robust performance.

Even though disambiguation with Lesk offers the initial *semantic pruning* capabilities, the tax2vec algorithm can further address potential noise as follows. As the user can determine the depth in the WordNet taxonomy that will be considered as the starting point for semantic space construction, potentially too specific terms can be avoided if necessary.

Finally, in the third step, tax2vec conducts *feature selection*. This part of the algorithm is responsible for *filtering* redundant and non-informative terms that could be considered as noise. We tested both supervised, as well as unsupervised feature selection methods, exploring whether additional information about class labels helps with term pruning. Apart from the semantic pruning and selection strategies discussed above, links, mentions and hashtags can be removed to further reduce the noise in social media texts (as mentioned in the description of the SVM implementation by Martinc et al. [60] in Section 4.2).

We believe all three steps to some extent address how noise is being handled. However, it is expected that additional grammar correction and text normalization could serve as a complementary step to offer improved performance on social media texts.

4. Experimental setting

This section presents the experimental setting used in testing the performance of tax2vec in document classification tasks. We begin by describing the data sets on which the method was tested. Next, we describe the classifiers used to assess the use of features constructed using tax2vec, along with the baseline approaches. We continue by describing the metrics used to assess classification performance, and the description of the experiments.

4.1. Data sets

We tested the effects of features produced with tax2vec on six different class labeled text data sets summarized in Table 1, intentionally chosen from different domains.

Table 1: Data sets used for experimental evaluation of tax2vec's impact on learning. Note that MNS corresponds to the maximum number of text segments (max. number of tweets or comments per user or number of news paragraphs as presented in Appendix B).

Data set (target)	Classes	Words	Unique words	Documents	MNS	Average tokens per segment
PAN 2017 (Gender)	2	5169966	607474	3600	102	14.23
MBTI (Personality)	16	11832937	372811	8676	89	27.98
PAN 2016 (Age)	5	943880	178450	402	202	13.17
BBC news	5	902036	58128	2225	76	70.39
Drugs (Side effects)	4	385746	27257	3107	3	41.47
Drugs (Overall effect)	4	385746	27257	3107	3	41.47

The first three data sets are composed of short documents from social media, where we consider classification of tweets.

PAN 2017 (Gender) data set. Given a set of tweets per user, the task is to predict the user's gender[1] [5].

MBTI (Meyers-Briggs personality type) data set. Given a set of tweets per user, the task is to predict to which personality class a user belongs[2], first discussed in [61].

[1] `https://pan.webis.de/clef17/pan17-web`

[2] `https://www.kaggle.com/datasnaek/mbti-type/kernels`

PAN 2016 (Age) data set. Given a set of tweets per user, the classifier should predict the users's age range[3] [18].

Next, we consider a news articles data set by which we test the potential of the method also on longer documents, while for few shot learning experiments (Section 5.3), we transform the setting to short text documents by using only few paragraphs per article and test whether competitive performance to full-text-based classification can be obtained.

BBC news data set. Given a news article (composed of a number of paragraphs)[4], the goal is to assign to it a topic from a list of topic categories[5] [62].

We also consider two biomedical data sets related to drug consumption. Here, the same training instances in the form of short user commentaries were used to predict two different targets.

Drug side effects. This data set links user opinions to side effects of a drug they are taking as treatment. The goal is to predict the side effects prior to experimental measurement [21].[6]

Drug effectiveness. Similarly to side effects (previous data set), the goal of this task is to predict drug effectiveness [21].

4.2. The classifiers used

As tax2vec serves as a preprocessing method for data enrichment with semantic features, arbitrary classifiers can use the resulting semantic features for learning. Note that in the experiments, the final feature space is composed of both semantic and non-semantic (original) features, i.e., the final feature set

[3] `https://pan.webis.de/clef18/pan18-web`

[4] Split to paragraphs according to the double new line is presented in Appendix B.

[5] `https://github.com/suraj-deshmukh/BBC-Dataset-News-Classification/blob/master/dataset/dataset.csv`

[6] `http://archive.ics.uci.edu/ml/datasets`

used for learning is formed *after* the semantic features have been constructed and selected, by concatenating the original features and the semantic features. We use the following learners:

PAN 2017 approach. An SVM-based approach that relies heavily on the method proposed by Martinc et al. [60] for the author profiling task in the PAN 2017 shared task [5]. This method is based on sophisticated hand-crafted features calculated on different levels of preprocessed text including optional social media text cleaning (e.g., Twitter hashtag, mentions, url replacement with filler tokens). The following features were used:

tf-idf weighted word unigrams calculated on lower-cased text with stopwords removed;

tf-idf weighted word bigrams calculated on lower-cased text with punctuation removed;

tf-idf weighted word bound character tetragrams calculated on lower-cased text;

tf-idf weighted punctuation trigrams (the so-called beg-punct [63], in which the first character is punctuation but other characters are not) calculated on lower-cased text;

tf-idf weighted suffix character tetragrams (the last four letters of every word that is at least four characters long [63]) calculated on lower-cased text;

emoji counts of the number of emojis in the document, counted by using the list of emojis created by [64][7]; this feature is only useful if the input text contains emojis;

document sentiment using the above-mentioned emoji list that contains the sentiment of a specific emoji, used to calculate the senti-

[7] http://kt.ijs.si/data/Emoji_sentiment_ranking/

ment of the entire document by simply adding the sentiment of all the emojis in the document; this feature is only useful if the input text contains emojis;

character flood counts calculated by the number of times that three or more identical character sequences appear in the document;

In contrast to the original approach proposed [60], we do not use POS tag sequences as features and a Logistic regression classifier is replaced by a Linear SVM. Here, we experimented with the regularization parameter C, for which values in range $\{1, 20, 50, 100, 200\}$ were tested. This SVM variant is from this point on referred to as "SVM (Martinc et al.)". As this feature construction pipeline consists of too many parameters, we were not able to perform extensive grid search due to computational complexity. Thus, we did not experiment with feature construction parameters, and kept the configuration proposed in the original study.

Linear SVM with automatic feature construction. The second learner is a libSVM linear classifier [65], trained on a predefined number of word and character level n-grams, constructed using Scikit-learn's *TfidfVectorizer* method. To find the best setting, we varied the SVM's C parameter in range $\{1, 20, 50, 100, 200\}$, the number of word features between $\{10000, 50000, 100000, 200000\}$ and character features between $\{0, 30\}$[8]. Note that the word features were sorted by decreasing frequency. Here, we considered (word) n-grams of lengths between two and six. This SVM variation is from this point on referred to as "SVM (generic)". The main difference between "SVM (generic)" and "SVM (Martinc et al.)" is that the latter approach also considers punctuation-based and suffix-based features. Further, it is capable of constructing features that represent document sentiment, which was proven to work well for social media data

[8] In Figure C.9 (Appendix C), the reader can observe the results of the initial experiments on the number of word features that led to selection of this hyperparameter range.

sets (e.g., tweets). Finally, Martinc's approach also accounts for character repetitions and has a parameter for social-media text cleaning in preprocessing. Note that for both SVM approaches we fine-tuned the hyperparameter C, as is common when employing SVMs. The hyperparameter's values govern how penalized the learner is for a miss-classified instance, which is a property that was shown to vary across data sets (see for example [66]).

Hierarchical attention networks (HILSTM). The first neural network baseline is the recently introduced hierarchical attention network [67]. Here, we performed a grid search over $\{64, 128, 256\}$ hidden layers sizes, embedding sizes of $\{128, 256, 512\}$, batch sizes of $\{8, 24, 52\}$ and number of epochs $\{5, 15, 20, 30\}$. For detailed explanation of the architecture, please refer to the original contribution [67]. We discuss the best-performing architecture in Section 5 below.

Deep feedforward neural networks. As tax2vec constructs feature vectors, we also attempted to use them as inputs for a standard feedforward neural network architecture [68, 69]. Here, we performed a grid search across hidden layer settings: $\{(128, 64), (10, 10, 10)\}$ (where for example $(128, 64)$ corresponds to a two hidden layer neural network, where in the first hidden layer there are 128 neurons and 64 in the second), batch sizes $\{8, 24, 52\}$ and the number of training epochs $\{5, 15, 20\}$.[9]

4.3. Semantic features

In addition to the semantic features constructed by tax2vec, doc2vec-based semantic features [71] were used as a baseline in order to allow for a simple comparison between two semantic feature construction approaches. They were concatenated with the features constructed by Martinc et al.'s SVM approach

[9]The two deep architectures were implemented using TensorFlow [70], and trained using a Nvidia Tesla K40 GPU. We report the best result for top 30 semantic features with the rarest terms heuristic.

described in Section 4.2, in order to compare the benefits merging the BoW-based representations with a different type of semantic features (embedding-based ones). We set the embedding dimension to 256, as it was shown that lower dimensional embeddings do not perform well [72].

4.4. Description of the experiments

The experiments were set up as follows. For the drug-related data sets, we used the splits given in the original paper [21]. For other data sets, we trained the classifiers using stratified 90% : 10% splits. For each classifier, 10 such splits were obtained. The measure used in all cases is F_1, where for the multiclass problems (e.g., MBTI), we use the micro-averaged F_1. All experiments were repeated five times using different random seeds. The features obtained using tax2vec are used in combination with SVM classifiers, while the other classifiers are used as baselines.[10]

5. Classification results

In this section we provide the results obtained by conducting the experiments outlined in the previous section. We begin by discussing the overall classification performance with respect to different heuristics used. Next, we discuss how tax2vec augments the learner's ability to classify when the number of text segments per user is reduced.

5.1. Classification performance evaluation

The F_1 results are presented in Table 2. The first observation is that combining BoW-based representations with semantic features (tax2vec or doc2vec) leads to performance improvements in five out of six cases (MBTI being the only data set where no improvement is detected). Tax2vec outperforms doc2vec-based vectors in three out of five data sets (PAN 2016 (Age), BBC News and

[10]Note that simple feedforward neural networks could also be used in combination with hypernym features—we leave such computationally expensive experiments for further work.

Drugs (effect)), while doc2vec-based features outperform tax2vec on two data sets (PAN 2017 (gender) and Drugs (Side)).

When it comes to tax2vec, up to 100 semantic features aid the SVM learners to achieve better accuracy. The most apparent improvement can be observed for the case of PAN 2016 (Age) data set, where the task was to predict age. Here, 10 semantic features notably improved the classifiers' performance (up to approximately 7% for SVM (generic)). Further, a minor improvement over the state-of-the-art was also observed on the PAN 2017 (Gender) data set and the BBC news categorization (see results for SVM (Martinc et al.)). Hierarchical attention networks outperformed all other learners for the task of side effects prediction, yet semantics-augmented SVMs outperformed neural models when general drug effects were considered as target classes. Similarly, no performance improvements were offered by tax2vec on the MBTI data set.

Table 2: Effect of the added semantic features to classification performance, where all text segments (tweets/comments per user or segments per news article) are used. The best performing feature selection heuristic for the majority of top performing classifiers was "rarest terms" or "Closeness centrality", indicating that only a handful of hypernyms carry added value, relevant for classification. Note that the results in the table correspond to the best performing combination of a classifier and a given heuristic.

# Semantic	Learner	PAN (Age)	PAN (Gender)	MBTI	BBC News	Drugs (effect)	Drugs (side)
0	HILSTM	0.422	0.752	0.407	0.833	0.443	0.514
0	SVM (Martinc et al.)	0.417	0.814	0.682	0.983	0.468	0.503
0	SVM (generic)	0.424	0.751	0.556	0.967	0.445	0.462
256 (doc2vec)	SVM (Martinc et al.)	0.422	0.817	0.675	0.979	0.416	0.523
30 (tax2vec)	DNN	0.400	0.511	0.182	0.353	0.400	0.321
10 (tax2vec)	SVM (Martinc et al.)	0.445	0.815	0.679	0.996	0.47	0.506
	SVM (generic)	0.502	0.781	0.556	0.972	0.445	0.469
25 (tax2vec)	SVM (Martinc et al.)	0.454	0.814	0.681	0.984	0.468	0.500
	SVM (generic)	0.484	0.755	0.554	0.967	0.449	0.466
50 (tax2vec)	SVM (Martinc et al.)	0.439	0.814	0.681	0.983	0.462	0.499
	SVM (generic)	0.444	0.751	0.554	0.963	0.446	0.463
100 (tax2vec)	SVM (Martinc et al.)	0.424	0.816	0.678	0.984	0.466	0.496
	SVM (generic)	0.422	0.749	0.551	0.958	0.443	0.46
500 (tax2vec)	SVM (Martinc et al.)	0.383	0.797	0.662	0.975	0.45	0.477
	SVM (generic)	0.400	0.724	0.532	0.909	0.424	0.438
1000 (tax2vec)	SVM (Martinc et al.)	0.368	0.783	0.647	0.964	0.436	0.466
	SVM (generic)	0.373	0.701	0.512	0.851	0.407	0.420

We now present the classification results in the form of critical distance diagrams, shown in Figures 4, 5 and 6. The diagrams show average ranks of different algorithms according to the (micro) F_1 measure. A red line connects groups of classifiers that are not statistically significantly different from each other at a confidence level of 5%. The significance levels are computed using Friedman multiple test comparisons followed by Nemenyi post-hoc correction [73]. For each data set, we selected the best performing parametrization (hyperparameter settings). The best (on average) performing C parameter for both SVM models was 50. The number of features that performed the best for all hyperparameter settings of the SVM (generic) considered in this study is 100,000. The HILSTM architecture's topology varied between data sets, yet we observed that the best results were obtained when more than 15 epochs of training were conducted, combined with the hidden layer size of 64 neurons, where the size of the attention layer was of the same dimension.

In terms of feature selection, the following can be observed (Figure 4). On average, the best performance was obtained when rarest terms heuristic was considered (first and fifth rank). Further, rarest terms, as well as the Personalized PageRank performed better (on average) than mutual information, which can be considered as a baseline in this comparison. The results indicate that my-

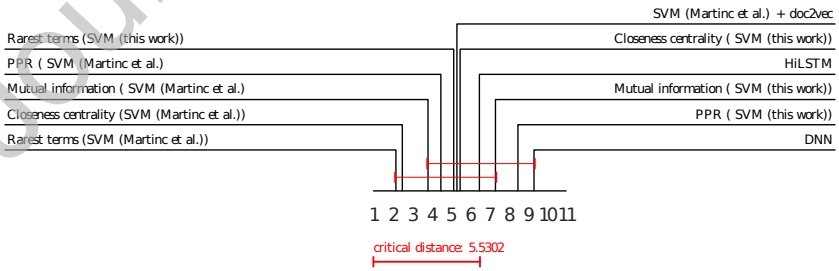


Figure 4: Average overall classifier ranks. The top (on average) performing classifier is an SVM (Martinc et al.) classifier augmented with semantic features, selected using either simple frequency counts or closeness centrality.

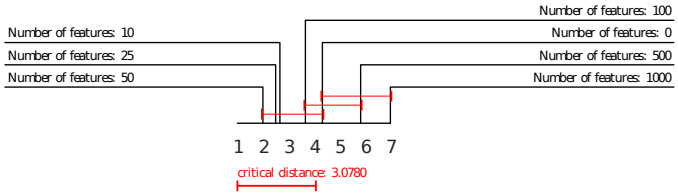


Figure 5: Effect of semantic features on average classifier rank. Up to 100 semantic features positively affects the classifiers' performance.

Figure 6: Overall model performance. SVMs dominate the short text classification. The diagram shows performance averaged over all data sets, where the best model parameterizations (see Table 2) were used for comparison.

opic feature selection is not optimal when considering novel semantic features. We can also observe that on average the configuration with doc2vec semantic features (SVM (Martinc et al.) + doc2vec) performs worse (ranking as sixth) than all other configurations with SVM (Martinc et al.).

In Figure 5, the reader can observe the performances *of all learners*, averaged w.r.t. to the number of semantic features. The drawn diagram indicates that adding 10, 25 or 50 features to a classifier perform similarly well, however, as also discussed in the previous paragraph, the performance drops when larger semantic space is considered.

Finally, in Figure 6 it can be observed that the overall performance of Martinc et al.'s SVMs is the best, followed by generic SVMs, as well as HILSTMs. We believe such performance drop with deep neural networks in general is due to concatenation of documents prior to learning, and as only a fixed sequence

Table 3: Effect of added semantic features to classification performance—few shot learning.

Semantic (tax2vec)	Learner	PAN (Age)	PAN (Gender)	MBTI	BBC News	Drugs (effect)	Drugs (side)
0	SVM (Martinc et al.)	0.378	0.617	0.288	0.977	0.468	0.503
	SVM (generic)	0.429	0.554	0.225	0.936	0.445	0.462
10	SVM (Martinc et al.)	0.39	0.616	0.292	0.981	0.47	0.503
	SVM (generic)	0.429	0.557	0.225	0.948	0.444	0.464
25	SVM (Martinc et al.)	0.429	0.618	0.288	0.979	0.465	0.5
	SVM (generic)	0.439	0.562	0.226	0.933	0.445	0.458
50	SVM (Martinc et al.)	0.402	0.617	0.288	0.974	0.474	0.504
	SVM (generic)	0.42	0.557	0.225	0.919	0.442	0.46
100	SVM (Martinc et al.)	0.382	0.614	0.286	0.974	0.476	0.493
	SVM (generic)	0.411	0.552	0.223	0.906	0.437	0.457
500	SVM (Martinc et al.)	0.359	0.604	0.276	0.959	0.465	0.471
	SVM (generic)	0.365	0.548	0.22	0.8	0.419	0.435
1000	SVM (Martinc et al.)	0.34	0.59	0.266	0.925	0.442	0.46
	SVM (generic)	0.359	0.535	0.213	0.704	0.412	0.417

length can be considered, potentially large parts of the token space were neglected during learning. A similar result was, for example observed in the most recent PAN competition [74].

5.2. Few-shot (per instance) learning

As discussed in the introductory sections, one of the goals of this paper was also to explore the setting, where only a handful of text segments per user are considered. Even though such setting is not strictly a few-shot learning [25], reducing the number of text segments per instance (e.g., user) aims to simulate a setting where there is limited information available. In Table 3, we present the results for the setting, where only (up to) 10 text segments (e.g., tweets or paragraphs in a given news article) were used for training.

The segments were sampled randomly. Only a single text segment per user was considered for the medical texts, as they consist of at max of three commentaries. Similarly, as the BBC news data set consists of news article-genre pairs, we split the news article to paragraphs, which we randomly sampled. The rationale for such sampling is to be able to evaluate tax2vec's performance when, for example, only a handful of paragraphs are available (e.g., only the lead).

We observe that tax2vec based features improve the learners' performance on all of the data sets, albeit by a small margin. The results indicate that

adding semantic information improves the performance as only a handful of text segments does not necessarily contain the relevant information.

5.3. Few-shot learning results

We next discuss the results of few-shot learning, as to our knowledge this type of experiments were not conducted before in combination with semantic feature construction methods. The first observation is, semantic features indeed offer more consistent performance improvements than those observed in Table 4, where incremental improvements were not observed on all data sets. In a few-shot learning scenario, however, on all data sets, the inclusion of semantic space either resulted in similar or better performance, indicating a consistent positive effect on the learning in a limited setting. The differences in learner's performance vary around 1% improvement. For example, a 1% improvement was observed for PAN 2016 (Age), BBC News and MBTI data sets.

We finally comment on the classification performance when considering the BBC data set when comparing to reported state-of-the-art. The observed results ($\geq$98%) are competitive to neural approaches, such as for example as reported in [75], where similar span of accuracy was observed. Furthermore, doc2vec-based models have been observed to perform similarly [76]. The results of this work indicate that by considering smaller number of paragraphs (instead of whole documents), competitive performance can be observed on the BBC data set.

5.4. Interpretation of results

In this section we explain the intuition behind the effect of semantic features on the classifier's performance. Note that the best performing SVM models consisted of thousands of tf-idf word and character level features, yet only up to 100 semantic features, when added, notably improved the performance. This effect can be understood via the way SVMs learn from high-dimensional data. With each new feature, we increase the dimensionality of the feature space. Even a single feature, when added, potentially impacts the hyperplane construction. Thus, otherwise problem-irrelevant features can become relevant when novel

features are added. We believe that adding semantic features to (raw) word tf-idf vector space introduces new information, crucial for successful learning, and potentially aligns the remainder of features so that the classifier can better separate the points of interest.

The other explanation for the notable differences in predictive performance is possibly related to small data set sizes, where only a handful of features can be of relevance and thus notably impact a given classifier's performance. We next discuss the impact of the number of selected semantic features on performance.

5.5. How large semantic space should be considered?

Tables 3 and 4 show that a relatively small number of semantic features are needed for potential performance gains. Note that the number of semantic features that need to be considered is around ≤ 100 in most of the cases. The results indicate that a relatively small proportion of the semantic space carries relevant (additional) information, whereas the remainder potentially introduces noise that degrades the performance. Note that in the limit every term from the taxonomy derived from a given corpus could be considered. In such a scenario, many terms would be irrelevant and would only introduce noise. The experiments conducted in this paper indicate that the threshold for the number of features is in the order of hundreds, yet not more features.

6. Qualitative assessment and explainability of tax2vec

This section discusses the properties of the resulting semantic space in Section 6.1, which is followed by a discussion on the explainability of the proposed tax2vec algorithm in Section 6.2.

6.1. Analysis of the resulting semantic space

In this section we discuss the qualitative properties of the obtained corpus-based taxonomies. We present the results concerning hypernym frequency distributions, as well as the overall structure of an example corpus-based taxonomy.

As the proposed approach is entirely symbolic—each feature can be traced back to a unique hypernym—we explored the feature space qualitatively by exploring the statistical properties of the induced taxonomy using graph-statistical approaches. Here, we modeled hypernym frequency distributions to investigate possible similarity with the Zipf's law [77]. The analysis was performed using the Py3plex library [78]. We also visualized the document-based taxonomy of the PAN 2016 (Age) data set using Cytoscape [79].

The examples in this section are all based on the corpus-based taxonomy, constructed from the PAN 2016 (Age) data set. The results of fitting various heavy-tailed distributions to the hypernym frequencies are given in Figure 7.

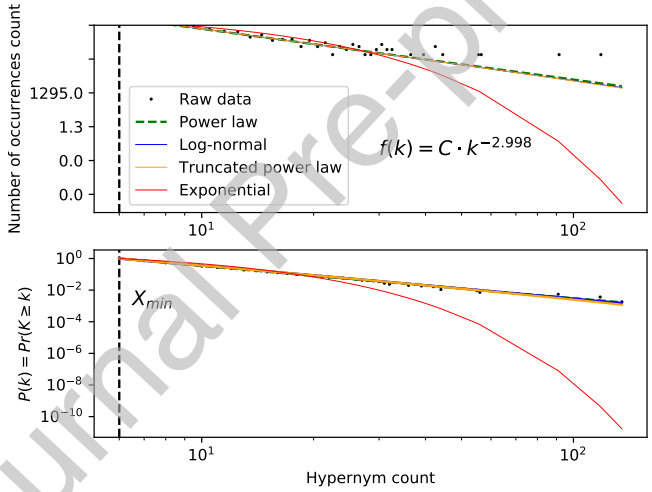


Figure 7: Hypernym frequency distribution for the PAN 2016 (Age) data set. The equation above the upper plot denotes the coefficients of a power law distribution (C is a constant). In real world phenomena, the exponent of the rightmost expression was observed to range between ≈ 2 and ≈ 3, indicating the hypernym structure of the feature space is subject to a heavy-tailed (possibly best fit—power law) distribution. The X_{min} denotes the hypernym count, after which notable differences in hypernym counts—scale free behavior is observed. Such distribution is to some extent expected, as some hypernyms are more general than others, and thus present in more document-hypernym mappings.

We fitted power law, truncated normal, log-normal and exponential distribu-

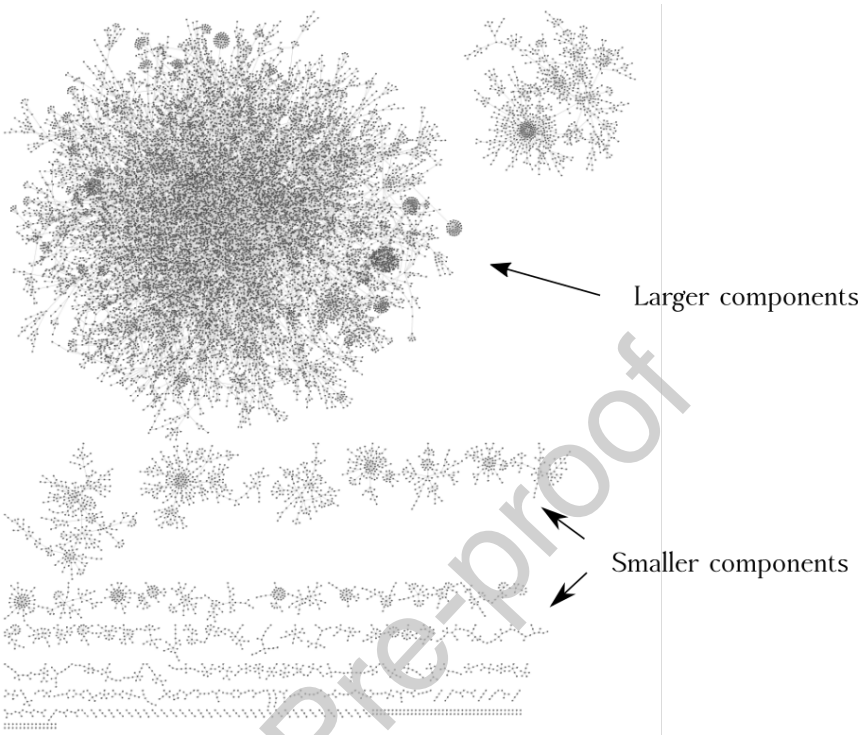


Figure 8: Topological structure of the hypernym space, induced from the PAN 2016 (Age) data set. Multiple connected components emerged, indicating not all hypernyms map to the same high-level concepts. Such segmentation is data set-specific, and can also potentially provide the means to compare semantic spaces of different data sets. It can be observed that the obtained space is organized in multiple separate components. The largest are drawn at the topmost part of the figure, whereas the smaller ones at the bottom. Such segmentation corresponds to generalizations based on different parts of speech, e.g., nouns and verbs.

tions to the hypernym frequency data. For detailed overview of the distributions we refer the reader to [80]. One of the key properties we researched was whether the underlying hypernym distribution is exponential or not, as non-exponential distributions indicate similarity with the well known Zipf's law [77]. The hypernym corpus-based taxonomy is visualized in Figure 8.

Here, each node represents a hypernym obtained in word-to-hypernym mapping phase of tax2vec. The edges represent the hypernymy relation between a given pair of hypernyms.

We next present the results of modeling the corpus-based hypernym frequency distributions. The two functions representing the best fit to hypernym frequency distributions are indeed the power law and the truncated power law. As similar behavior is observed for word frequency in documents [77], we believe hypernym distributions are a natural extension, as naturally, if a high-frequency word maps to a given hypernym, the hypernym will be relatively more common with respect to the occurrence of other hypernyms.

We observe that multiple connected components of varying sizes emerge. There exists only a single largest connected component, which consists of more general noun hypernyms, such as *entity* and similar. Interestingly, many smaller components also emerged, indicating parts of the word vector space could be mapped to very specific, disconnected parts of the WordNet taxonomy. Some examples of small disconnected components include (one component per line), indicating also verb-level semantics can be captured and taken into account:

$'spot.v.02', 'discriminate.v.03' 'homestead.v.01', 'settle.v.21'$
$'smell.v.05', 'perceive.v.02', 'understand.v.02'$
$'dazzle.v.01', 'blind.v.01'$
$'romance.v.02', 'adore.v.01', 'care\_for.v.02', 'love.v.03', 'love.v.01'$
$'surrender.v.01', 'yield.v.12', 'capitulate.v.01'$

6.2. Explainability of tex2vec

As discussed in the previous sections, tax2vec selects a set of hypernyms according to a given heuristic and uses them for learning. One of the key benefits of such approach is that the selected semantic features can easily be inspected, hence potentially offering interesting insights into the semantics, underlying the problem at hand.

We discuss here a set of 30 features which emerged as relevant according to the "mutual information" heuristic when the BBC News and PAN 2016 (Age) data sets were considered. Here, tax2vec was trained on 90% of the data, the rest was removed (test set). The features and their corresponding mutual information scores are shown in Table 4.

Table 4: Most informative features with respect to the target class (ranked by MI)—Classes represent news topics (BBC) and different age intervals (PAN 2016 (Age)). Individual target classes are sorted according to a descending mutual information with respect to a given feature.

		Sorted target class-mutual information pairs				
Semantic feature	Average MI	Class 1	Class 2	Class 3	Class 4	Class 5
BBC News data set						
tory.n.03	0.057	politics:0.14	entertainment:0.05	business:0.03	sport:0.01	x
movie.n.01	0.059	business:0.14	politics:0.04	entertainment:0.04	sport:0.02	x
conservative.n.01	0.061	politics:0.15	entertainment:0.05	business:0.03	sport:0.01	x
vote.n.02	0.061	business:0.15	entertainment:0.04	politics:0.04	sport:0.02	x
election.n.01	0.063	entertainment:0.16	business:0.05	politics:0.04	sport:0.0	x
topology.n.04	0.063	entertainment:0.16	business:0.05	politics:0.04	sport:0.0	x
mercantile_establishment.n.01	0.068	politics:0.17	business:0.07	entertainment:0.03	sport:0.01	x
star_topology.n.01	0.069	politics:0.17	business:0.07	entertainment:0.03	sport:0.01	x
rightist.n.01	0.074	politics:0.18	business:0.06	entertainment:0.04	sport:0.01	x
marketplace.n.02	0.087	entertainment:0.22	business:0.06	politics:0.05	sport:0.01	x
PAN (Age) data set						
hippie.n.01	0.007	25-34:0.01	35-49:0.01	18-24:0.0	65-xx:0.0	50-64:0.0
ceremony.n.03	0.007	25-34:0.01	35-49:0.01	18-24:0.01	65-xx:0.0	50-64:0.0
resource.n.02	0.008	50-64:0.02	18-24:0.01	25-34:0.0	65-xx:0.0	35-49:0.0
draw.v.07	0.008	25-34:0.02	35-49:0.01	50-64:0.01	65-xx:0.0	18-24:0.0
observation.n.02	0.008	25-34:0.02	35-49:0.01	50-64:0.01	65-xx:0.0	18-24:0.0
wine.n.01	0.008	35-49:0.02	25-34:0.01	18-24:0.01	50-64:0.01	65-xx:0.0
suck.v.02	0.008	25-34:0.02	50-64:0.02	35-49:0.0	65-xx:0.0	18-24:0.0
sleep.n.03	0.008	25-34:0.02	50-64:0.02	35-49:0.0	65-xx:0.0	18-24:0.0
recognize.v.09	0.009	25-34:0.02	35-49:0.02	18-24:0.0	50-64:0.0	65-xx:0.0
weather.v.04	0.009	25-34:0.02	50-64:0.02	35-49:0.0	18-24:0.0	65-xx:0.0
invention.n.02	0.009	25-34:0.02	35-49:0.01	18-24:0.01	50-64:0.0	65-xx:0.0
yankee.n.03	0.01	50-64:0.02	18-24:0.01	25-34:0.01	35-49:0.0	65-xx:0.0

We can observe that the "sport" topic (BBC data set) is not well associated with the prioritized features. On the contrary, terms such as "rightist" and "conservative" emerged as relevant for classifying into the "politics" class. Similarly, "marketplace" for example, appeared relevant for classifying into the "entertainment" class. Even more interesting associations emerged when the same feature ranking was conducted on the PAN 2016 (Age) data set. Here, terms such as "resource" and "wine" were relevant for classifying middle-aged ("wine") and older adult ("resource") populations. Note that the older population (65-xx class) was not associated with any of the hypernyms. We believe the reason for this is that the number of available tweets decreases with age.

We repeated a similar experiment (BBC data set) using the "rarest terms"

heuristic. The terms which emerged are:

'problem.n.02', 'question.n.02', 'riddle.n.01', 'salmon.n.04', 'militia.n.02', 'orphan.n.04', 'taboo.n.01', 'desertion.n.01', 'dearth.n.02', 'outfitter.n.02', 'scarcity.n.01', 'vasodilator.n.01', 'dilator.n.02', 'fluoxetine.n.01', 'high blood pressure.n.01', 'amlodipine besylate.n.01', 'drain.n.01', 'imperative mood.n.01', 'fluorescent.n.01', 'veneer.n.01', 'autograph.n.01', 'oak.n.02', 'layout.n.01', 'wall.n.01', 'firewall.n.03', 'workload.n.01', 'manuscript.n.02', 'cake.n.01', 'partition.n.01', 'plasterboard.n.01'

Even if the feature selection method is unsupervised (not directly associated to classes), we can immediately observe that the features correspond to different topics, raging from medicine (e.g., high blood presure), politics (e.g., militia), food(e.g., cake) and more, indicating that the rarest hypernyms are indeed diverse and as such potentially useful for the learner.

The results suggest that tax2vec could potentially also be used to inspect the semantic background of a given data set directly, regardless of the learning task. We believe there are many potential uses for the obtained features, including the following, to be addressed in further work.

- Concept drift detection, i.e. topics change over time; could it be qualitatively detected?
- Topic domination, i.e. what type of topic is dominant with respect to e.g., a geographical region inspected?
- What other learning tasks can benefit by using second level semantics? Can the obtained features be used, for example, for fast keyword search?

7. Implementation and availability

The tax2vec algorithm is implemented in Python 3, where Multiprocessing[11], SciPy [81] and Numpy [82] libraries are used for fast (sparse), vectorized

[11] `https://docs.python.org/2/library/multiprocessing.html`

operations and parallelism.

As performing a grid search over several parameters is computationally expensive, the majority of the experiments were conducted using the SLING supercomputing architecture.[12]

We developed a stand-alone library that relatively seamlessly fits into existing text mining workflows, hence the Scikit-learn's model syntax was adopted [83]. The algorithm is first initiated as an object:

```
vectorizer = tax2vec(heuristic,number of features)
```

followed by standard *fit* and *transform* calls:

```
new_features = vectorizer.fit_transform(corpus, optional labels)
```

Such implementation offers fast prototyping capabilities, needed ubiquitously in the development of learning algorithms and executable NLP and text mining workflows.

The proposed tax2vec approach is freely available as a Python 3 library at `https://github.com/SkBlaz/tax2vec`, which includes also the installation instructions.

8. Conclusions and future work

In this work we propose tax2vec, a parallel algorithm for taxonomy-based enrichment of text documents. Tax2vec first maps the words from individual documents to their hypernym counterparts, which are considered as candidate features and weighted according to a normalized tf-idf metric. To select only a user-specified number of relevant features, tax2vec implements multiple feature selection heuristics, which select only the potentially relevant features. The sparse matrix of constructed features is finally used alongside the bag-of-words document representations for the task of text classification, where we study its

[12] `http://www.sling.si/sling/`

performance on small data sets, where both the number of text segments per user, as well as the number of overall users considered are small.

The tax2vec approach considerably improves the classification performance especially on data sets consisting of tweets, but also on the news. The proposed implementation offers a simple-to-use API, which facilitates inclusion into existing text preprocessing workflows.

As the next step, the tax2vec will be tested on SMS spam data [84], which is another potentially interesting short text data set where taxonomy-based features could improve performance and help the user better understand what classifies as spam (and what not).

One of the drawbacks we plan to address is the support for arbitrary directed acyclic multigraphs—structures commonly used to represent background knowledge. Support for such knowledge would offer a multitude of applications in e.g., biology, where gene ontology and other resources which annotate entities of interest are freely available.

In this work we focus on BoW representation of documents, yet we believe tax2vec could also be used along Continuous Bag-of-Words (CBoW) models. We leave such experimentation for further work.

Even though we use Lesk for the disambiguation task, we believe recent advancements in neural disambiguation [85] could also be a "drop-in" replacement for this part of tax2vec. We leave the exploration of such options for further work.

In this work we explored how WordNet could be adapted for scalable feature construction, however tax2vec is by no means limited to manually curated relational (hierarchical) structures. As part of the further work, we believe feature construction based on *knowledge graphs* could also be an option.

The abundance of neural embedding methods introduced in the recent years can be complementary to tax2vec. Understanding how the performance can be improved by jointly using both tax2vec's features and neural network-based ones is a potential interesting research opportunity. Further, in NLP setting, not much attention is devoted to this topic, thus we believe these results offer

new trajectories for few-shot learning research.

Other further work considers joining the tax2vec features with existing state-of-the-art deep learning approaches, such as the hierarchical attention networks, which are—according to this study—not very suitable for learning on scarce data sets. We believe that the introduction of semantics into deep learning could be beneficial for both performance, as well as the interpretability of currently poorly understood black-box models.

Finally, as the main benefit of tax2vec is its explanatory power, we believe it could be used for fast keyword search; here, for example, new news or articles could be used as inputs, where the ranked list of semantic features could be directly used as candidate keywords.

Acknowledgements

We would first like to thank the reviewers for insightful comments that improved this paper. The work of the first author was funded by the Slovenian Research Agency through a young researcher grant (TSP). The work of other authors was supported by the Slovenian Research Agency (ARRS) core research programme *Knowledge Technologies* (P2-0103), an ARRS funded research project *Semantic Data Mining for Linked Open Data* (financed under the ERC Complementary Scheme, N2-0078) and European Unionś Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). This research has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 769661, SAAM project. We also gratefully acknowledge the support of NVIDIA Corporation for the donation of Titan-XP GPU.

References

[1] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805.

[3] D. Tang, B. Qin, T. Liu, Document modeling with gated recurrent neural network for sentiment classification, in: Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, pp. 1422–1432.

[4] M. Kusner, Y. Sun, N. Kolkin, K. Weinberger, From word embeddings to document distances, in: International Conference on Machine Learning, 2015, pp. 957–966.

[5] F. Rangel, P. Rosso, M. Potthast, B. Stein, Overview of the 5th author profiling task at pan 2017: Gender and language variety identification in twitter, Working Notes Papers of the CLEF.

[6] A. Ławrynowicz, Semantic Data Mining: An Ontology-based Approach, Vol. 29, IOS Press, 2017.

[7] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, C. Rudin, Learning certifiably optimal rule lists, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 35–44.

[8] A. Vavpetič, N. Lavrač, Semantic subgroup discovery systems and workflows in the sdm-toolkit, The Computer Journal 56 (3) (2013) 304–320.

[9] M. Perovšek, A. Vavpetič, B. Cestnik, N. Lavrač, A wordification approach to relational data mining, in: J. Fürnkranz, E. Hüllermeier, T. Higuchi (Eds.), Discovery Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 141–154.

[10] P. R. Adhikari, A. Vavpetič, J. Kralj, N. Lavrač, J. Hollmén, Explaining mixture models through semantic pattern mining and banded matrix visualization, Machine Learning 105 (1) (2016) 3–39.

[11] S. Scott, S. Matwin, Text classification using wordnet hypernyms, Usage of WordNet in Natural Language Processing Systems.

[12] C. Kim, P. Yin, C. X. Soto, I. K. Blaby, S. Yoo, Multimodal biological analysis using NLP and expression profile, in: 2018 New York Scientific Data Summit (NYSDS), 2018, pp. 1–4.

[13] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, T. S. Huang, Heterogeneous network embedding via deep architectures, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, ACM, New York, NY, USA, 2015, pp. 119–128.

[14] L. C. Freeman, Research Methods in Social Network Analysis, Routledge, 2017.

[15] J. Wang, Z. Wang, D. Zhang, J. Yan, Combining knowledge with deep convolutional neural networks for short text classification, in: Proceedings of IJCAI, Vol. 350, 2017, p. online.

[16] M. Chen, X. Jin, D. Shen, Short text classification improved by learning multi-granularity topics, in: Twenty-Second International Joint Conference on Artificial Intelligence, 2011.

[17] F. Rangel, P. Rosso, I. Chugur, M. Potthast, M. Trenkmann, B. Stein, B. Verhoeven, W. Daelemans, Overview of the 2nd author profiling task at PAN 2014, in: Working Notes Papers of the CLEF 2014, 2014, pp. 1–30.

[18] F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast, B. Stein, Overview of the 4th author profiling task at pan 2016: cross-genre evaluations, in: Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al., 2016, pp. 750–784.

[19] Z. Chu, S. Gianvecchio, H. Wang, S. Jajodia, Detecting automation of twitter accounts: Are you a human, bot, or cyborg?, IEEE Transactions on Dependable and Secure Computing 9 (6) (2012) 811–824.

[20] Z. Chu, S. Gianvecchio, H. Wang, S. Jajodia, Who is tweeting on twitter: human, bot, or cyborg?, in: Proceedings of the 26th annual computer security applications conference, ACM, 2010, pp. 21–30.

[21] F. Grässer, S. Kallumadi, H. Malberg, S. Zaunseder, Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning, in: Proceedings of the 2018 International Conference on Digital Health, DH '18, ACM, New York, NY, USA, 2018, pp. 121–125.

[22] R. Boyce, G. Gardner, H. Harkema, Using natural language processing to identify pharmacokinetic drug-drug interactions described in drug package inserts, in: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing, Association for Computational Linguistics, 2012, pp. 206–213.

[23] J. Cho, K. Lee, E. Shin, G. Choy, S. Do, How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?, arXiv preprint arXiv:1511.06348.

[24] R. Socher, M. Ganjoo, C. D. Manning, A. Ng, Zero-shot learning through cross-modal transfer, in: Proceedings of the Advances in neural information processing systems, 2013, pp. 935–943.

[25] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.

[26] T. K. Landauer, Latent Semantic Analysis, Wiley Online Library, 2006.

[27] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (1) (2003) 993–1022.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 3111–3119.

[29] L. Cagliero, P. Garza, Improving classification models with taxonomy information, Data & Knowledge Engineering 86 (2013) 85–101.

[30] N. Xu, J. Wang, G. Qi, T. S. Huang, W. Lin, Ontological random forests for image classification, in: Computer Vision: Concepts, Methodologies, Tools, and Applications, IGI Global, 2018, pp. 784–799.

[31] M. K. Elhadad, K. M. Badran, G. I. Salama, A novel approach for ontology-based feature vector generation for web text document classification, International Journal of Software Innovation (IJSI) 6 (1) (2018) 1–10.

[32] R. Kaur, M. Kumar, Domain ontology graph approach using markov clustering algorithm for text classification, in: International Conference on Intelligent Computing and Applications, Springer, 2018, pp. 515–531.

[33] T. N. Mansuy, R. J. Hilderman, Evaluating wordnet features in text classification models., in: FLAIRS Conference, 2006, pp. 568–573.

[34] M. Bunge, Causality and Modern Science, Routledge, 2017.

[35] J. Pearl, Causality, Cambridge university press, 2009.

[36] U. Stańczyk, L. C. Jain, Feature selection for Data and Pattern Recognition, Springer, 2015.

[37] A. G. Kakisim, I. Sogukpinar, Unsupervised binary feature construction method for networked data, Expert Systems with Applications 121 (2019) 256 – 265.

[38] B. Škrlj, N. Lavrač, J. Kralj, Symbolic graph embedding using frequent pattern mining, in: P. Kralj Novak, T. Šmuc, S. Džeroski (Eds.), Discovery Science, Springer International Publishing, Cham, 2019, pp. 261–275.

[39] N. Tomašev, K. Buza, K. Marussy, P. B. Kis, Hubness-aware classification, Instance Selection and Feature Construction: Survey and Extensions to Time-series, in: Feature selection for data and pattern recognition, Springer, 2015, pp. 231–262.

[40] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[41] Z. M. Hira, D. F. Gillies, A review of feature selection and feature extraction methods applied on microarray data, Advances in bioinformatics 2015.

[42] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 855–864.

[43] Y. Dong, N. V. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'17, ACM, New York, NY, USA, 2017, pp. 135–144.

[44] S. Jaeger, S. Fulle, S. Turk, Mol2vec: Unsupervised machine learning approach with chemical intuition, Journal of Chemical Information and Modeling 58 (1) (2018) 27–35.

[45] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, Struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'17, ACM, New York, USA, 2017, pp. 385–394.

[46] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017, p. online.

[47] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 1024–1034.

[48] F. Železnỳ, N. Lavrač, Propositionalization-based relational subgroup discovery with RSD, Machine Learning 62 (1-2) (2006) 33–63.

[49] M. Perovšek, A. Vavpetič, B. Cestnik, N. Lavrač, A wordification approach to relational data mining, in: International Conference on Discovery Science, Springer, 2013, pp. 141–154.

[50] F. Rangel, P. Rosso, L. Cappellato, N. Ferro, H. Müller, D. Losada, Overview of the 7th author profiling task at pan 2019: Bots and gender profiling, in: CLEF, 2019, p. online.

[51] G. A. Miller, Wordnet: A lexical database for english, Commun. ACM 38 (11) (1995) 39–41.

[52] A. Gonzalez-Agirre, E. Laparra, G. Rigau, Multilingual central repository version 3.0: upgrading a very large lexical knowledge base, in: Proceedings of the 6th Global WordNet Conference (GWC 2012), Matsue, 2012, p. online.

[53] R. Navigli, Word sense disambiguation: A survey, ACM Comput. Surv. 41 (2) (2009) 10:1–10:69.

[54] P. Basile, A. Caputo, G. Semeraro, An enhanced lesk word sense disambiguation algorithm through a distributional semantic model, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 1591–1600.

[55] C. D. Manning, P. Raghavan, H. Schtze, Scoring, term weighting, and the vector space model, Cambridge University Press, 2008, Ch. first, p. 100123.

[56] U. Brandes, A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology 25 (2) (2001) 163–177.

[57] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on pattern analysis and machine intelligence 27 (8) (2005) 1226–1238.

[58] J. Kralj, M. Robnik-Sikonja, N. Lavrac, NetSDM: Semantic data mining with network analysis, Journal of Machine Learning Research 20 (32) (2019) 1–50.

[59] J. Kralj, Heterogeneous information network analysis for semantic data mining: Doctoral dissertation, Ph.D. thesis, J. Kralj (2017).

[60] Matej Martinc and Iza Škrjanec and Katja Zupan and Senja Pollak, Pan 2017: Author profiling - gender and language variety prediction, in: CLEF, 2017, p. online.

[61] I. B. Myers, The myers-briggs type indicator: Manual (1962).

[62] D. Greene, P. Cunningham, Practical solutions to the problem of diagonal dominance in kernel document clustering, in: Proceedings of the 23rd International Conference on Machine learning (ICML'06), ACM Press, 2006, pp. 377–384.

[63] U. Sapkota, S. Bethard, M. Montes-y-Gómez, T. Solorio, Not all character n-grams are created equal: A study in authorship attribution, in: NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, 2015, pp. 93–102.

[64] P. K. Novak, J. Smailović, B. Sluban, I. Mozetič, Sentiment of emojis, PloS one 10 (12) (2015) e0144296.

[65] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM transactions on intelligent systems and technology (TIST) 2 (3) (2011) 27.

[66] D. Meyer, F. Leisch, K. Hornik, The support vector machine under test, Neurocomputing 55 (1) (2003) 169 – 186, support Vector Machines. doi:https://doi.org/10.1016/S0925-2312(03)00431-4.

[67] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.

[68] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.

[69] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.

[70] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).

[71] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, 2014, pp. 1188–1196.

[72] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[73] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.

[74] M. Martinc, B. Škrlj, S. Pollak, Fake or not: Distinguishing between bots, males and females, in: Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019), 2019, p. online.

[75] M. N. Asim, M. U. G. Khan, M. I. Malik, A. Dengel, S. Ahmed, A robust hybrid approach for textual document classification, arXiv preprint arXiv:1909.05478.

[76] L. Q. Trieu, H. Q. Tran, M.-T. Tran, News classification from social media using twitter-based doc2vec model and automatic query expansion, in: Proceedings of the Eighth International Symposium on Information and Communication Technology, 2017, pp. 460–467.

[77] S. T. Piantadosi, Zipfs word frequency law in natural language: A critical review and future directions, Psychonomic bulletin & review 21 (5) (2014) 1112–1130.

[78] B. Škrlj, J. Kralj, N. Lavrač, Py3plex: A library for scalable multilayer network analysis and visualization, in: Complex Networks and Their Applications VII, Springer International Publishing, Cham, 2019, pp. 757–768.

[79] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, T. Ideker, Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome research 13 (11) (2003) 2498–2504.

[80] S. Foss, D. Korshunov, S. Zachary, et al., An introduction to Heavy-tailed and Subexponential Distributions, Vol. 6, Springer, 2011.

[81] E. Jones, T. Oliphant, P. Peterson, SciPy: Open source scientific tools for Python, http://www.scipy.org/ (2001–).

[82] S. v. d. Walt, S. C. Colbert, G. Varoquaux, The numpy array: a structure for efficient numerical computation, Computing in Science & Engineering 13 (2) (2011) 22–30.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.

[84] S. J. Delany, M. Buckley, D. Greene, Sms spam filtering: Methods and data, Expert Systems with Applications 39 (10) (2012) 9899–9908.

[85] I. Iacobacci, M. T. Pilehvar, R. Navigli, Embeddings for word sense disambiguation: An evaluation study, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Vol. 1, 2016, pp. 897–907.

Appendix A. Personalized PageRank algorithm

The Personalized PageRank (PPR) algorithm is described below. Let V represent the nodes of the corpus-based taxonomy. For each node $u \in V$, a feature vector is computed by calculating the stationary distribution of a random walk, starting at node u. The stationary distribution is approximated by using power iteration, where the i-th component of the approximation in the k-th iteration is computed as

$$\gamma_u(i)^{(k+1)} = \alpha \cdot \sum_{j \to i} \frac{\gamma_u(j)^{(k)}}{d_j^{out}} + (1-\alpha) \cdot v_u(i); k = 1, 2, \ldots \tag{A.1}$$

The number of iterations k is increased until the stationary distribution converges to the stationary distribution vector (PPR value for node i). In the above equation, α is the damping factor that corresponds to the probability that a random walk follows a randomly chosen outgoing edge from the current node rather than restarting its walk. The summation index j runs over all nodes of the network that have an outgoing connection toward j, (denoted as $j \to i$ in the sum), and d_j^{out} is the out degree of node d_j. The term $v_u(i)$ is the restart distribution that corresponds to a vector of probabilities for a walker's return to the starting node u, i.e. $v_u(u) = 1$ and $v_u(i) = 0$ for $i \neq u$. This vector guarantees that the walker will jump back to the starting node u in case of a restart.[13]

[13]Note that if the binary vector were instead composed exclusively of ones, the iteration would compute the global PageRank vector, and Equation A.1 would correspond to the standard PageRank iteration.

Appendix B. Example document split

While for the data sets consisting of tweets and short comments, the number of segments in a document corresponds to the number of tweets or comments by a user, in the news data set, we varied the size of the news (to create short documents) by splitting the news into paragraphs (we denote such paragraph splits with ———). An example of segmentation of a news from the BBC data set[14] is listed below.

——— The decision to keep interest rates on hold at 4.75% earlier this month was passed 8-1 by the Bank of England's rate-setting body, minutes have shown.——— One member of the Bank's Monetary Policy Committee (MPC) - Paul Tucker - voted to raise rates to 5%. The news surprised some analysts who had expected the latest minutes to show another unanimous decision. Worries over growth rates and consumer spending were behind the decision to freeze rates, the minutes showed. The Bank's latest inflation report, released last week, had noted that the main reason inflation might fall was weaker consumer spending.——— However, MPC member Paul Tucker voted for a quarter point rise in interest rates to 5%. He argued that economic growth was picking up, and that the equity, credit and housing markets had been stronger than expected.——— The Bank's minutes said that risks to the inflation forecast were "sufficiently to the downside" to keep rates on hold at its latest meeting. However, the minutes added: "Some members noted that an increase might be warranted in due course if the economy evolved in line with the central projection". Ross Walker, UK economist at Royal Bank of Scotland, said he was surprised that a dissenting vote had been made so soon. He said the minutes appeared to be "trying to get the market to focus on the possibility of a rise in rates". "If the economy pans out as they expect then they are probably going to have to hike rates." However, he added, any rate increase is not likely to happen until later

[14] https://github.com/suraj-deshmukh/BBC-Dataset-News-Classification/blob/master/dataset/dataset.csv

this year, with MPC members likely to look for a more sustainable pick up in consumer spending before acting.

This news article is split by a parser into the following four segments (and in short document setting only one paragraph is used to represent the document).

- The decision to keep interest rates on hold at 4.75% earlier this month was passed 8-1 by the Bank of England's rate-setting body, minutes have shown.

- One member of the Bank's Monetary Policy Committee (MPC) - Paul Tucker - voted to raise rates to 5%. The news surprised some analysts who had expected the latest minutes to show another unanimous decision. Worries over growth rates and consumer spending were behind the decision to freeze rates, the minutes showed. The Bank's latest inflation report, released last week, had noted that the main reason inflation might fall was weaker consumer spending.

- However, MPC member Paul Tucker voted for a quarter point rise in interest rates to 5%. He argued that economic growth was picking up, and that the equity, credit and housing markets had been stronger than expected.

- The Bank's minutes said that risks to the inflation forecast were"sufficiently to the downside" to keep rates on hold at its latest meeting. However, the minutes added: "Some members noted that an increase might be warranted in due course if the economy evolved in line with the central projection." Ross Walker, UK economist at Royal Bank of Scotland, said he was surprised that a dissenting vote had been made so soon. He said the minutes appeared to be "trying to get the market to focus on the possibility of a rise in rates." "If the economy pans out as they expect then they are probably going to have to hike rates." However, he added, "any rate increase is not likely to happen until later this year, with MPC members likely to look for a more sustainable pick up in consumer spending before

acting."

Appendix C. Impact of different number of features across data sets

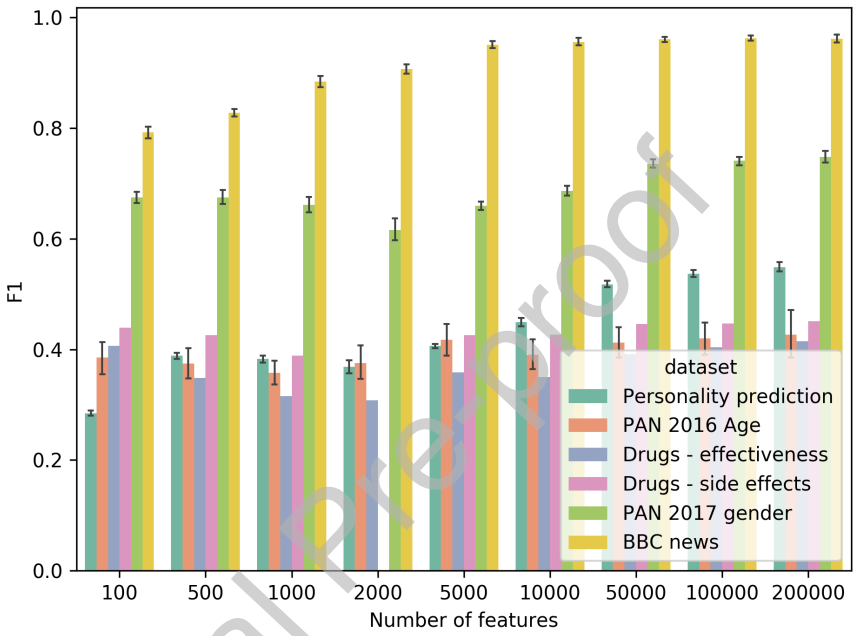


Figure C.9: Impact of the number of features used by the SVM (generic) on the F1 performance. The best performances were observed for feature numbers (word tokens) $\geq$ 10,000, hence these feature numbers were considered in the more expensive experiment stage with semantic vectors.