Knowledge graph aware text classification

Nela Petrželková*
Jožef Stefan Institute
Ljubljana, Slovenia
nela.petrzelkova@seznam.cz

Blaž Škrlj
Jožef Stefan Institute and
Jožef Stefan Int. Postgraduate School
Ljubljana, Slovenia
blaz.skrlj@ijs.si

Nada Lavrač
Jožef Stefan Institute
Ljubljana, Slovenia
nada.lavrac@ijs.si

ABSTRACT

Knowledge graphs are becoming ubiquitous in many scientific and industrial domains, ranging from biology, industrial engineering to natural language processing. In this work we explore how one of the largest currently available knowledge graphs, the Microsoft Concept Graph, can be used to construct interpretable features that are of potential use for the task of text classification. By exploiting graph-theoretic feature ranking, introduced as part of the existing tax2vec algorithm, we show that massive, real-life knowledge graphs can be used for the construction of features, derived from the relational structure of the knowledge graph itself. To our knowledge, this is one of the first approaches that explores how interpretable features can be constructed from the Microsoft Concept graph with more than five million concepts and more than 80 million IsA relations for the task of text classification. The proposed solution was evaluated on eight real-life text classification data sets.

KEYWORDS

knowledge graphs, text classification, feature construction, semantic enrichment

1 INTRODUCTION

Text classification is the process of assigning labels to text according to its content. It is one of the fundamental tasks in Natural Language Processing (NLP) with various applications such as spam detection, topic labeling, sentiment analysis, news categorization and many more [1]. In recent years, *knowledge graphs*—real-life graph-structured sources of knowledge—are becoming an interesting source of background knowledge, potentially useful in contemporary machine learning [2]. Knowledge graphs, such as DBPedia[1] or the Microsoft Concept Graph[2] span tens of millions of triplets of the form subject-predicate-object, and include many potentially interesting relations, from which a given machine learning algorithm can potentially benefit.

In this work we propose an approach to scalable *feature construction* from one of the largest freely available knowledge graphs, and demonstrate its utility on multiple real life data sets. The main contributions of this work are as follows:

(1) We propose an extension to the tax2vec [3] algorithm for semantic feature construction, adapting it to operate with real-life knowledge graphs comprised of tens of millions of triplets.
(2) The proposed method is extensively empirically evaluated, indicating that the proposed semantic feature construction aids the classification performance on many real-life datasets.
(3) The implemented method is freely available[3] with a simple-to-use, scikit-learn API.

The paper is structured as follows. Section 2 presents the background and related work. Section 3 presents the proposed approach to semantic feature construction using the information from a given knowledge graph. Section 4 describes the experimental setting and the results, followed by a summary and further work in Section 5.

2 BACKGROUND AND RELATED WORK

In text classification tasks, characterized by short documents or small amounts of documents, deep learning methods are frequently outperformed by more standard approaches, including SVMs [4]. In such settings, it was shown that approaches capable of using semantic context may outperform the naïve learning approaches, the examples are among other based on Latent Dirichlet Allocation [5], Latent Semantic Analysis [6] or word embeddings [7], which is referred to as first-level context.

Second-level context can be introduced by adding *background knowledge* into a learning process, which may help to increase performance and improve interpretability. Usage of knowledge graphs also helped in classification with extending neural network based lexical word embedding objective function [8]. Elhadad et al. [9] present an ontology-based web document, while Kaur et al. [10] propose a clustering-based algorithm for document classification that also benefits from knowledge stored in the underlying ontologies. Use of hypernym-based features was performed already in e.g., the Ripper rule learning algorithm [11]. Wang and Domeniconi [12] used the derived background knowledge from Wikipedia for text enriching. In short document classification, it was shown that the tax2vec algorithm (described below) can help those classifiers gain better results by adding *extra semantic knowledge* to the feature vectors.

The tax2vec [3] is an algorithm for *semantic feature construction* that can be used to enrich the feature vectors constructed by the established text processing methods such as the tf-idf. It takes as input a labeled or unlabeled corpus of documents and a word taxonomy, i.e. a directed graph to which parts of a given document map to. It outputs a matrix of *semantic feature vectors* where each row represents a semantics-based vector representation of one input document. It makes it by mapping the words from the document to a given taxonomy, WordNet or in this work Microsoft Concept Graph, by which it creates the collection of terms for each document and from it, a *corpus taxonomy*—a relational structure specific to the considered document space. The terms presented in the corpus taxonomy represent the potential features.

[1] https://wiki.dbpedia.org/
[2] https://concept.research.microsoft.com/Home/Introduction
[3] https://github.com/SkBlaz/tax2vec

Information society '20, October 5–9, 2020, Ljubljana, Slovenia

3 KNOWLEDGE GRAPH-BASED SEMANTIC FEATURE CONSTRUCTION

Semantic features are constructed as follows. With the help of spaCy library [13], we first find *nouns* in each document in the corpus and for every noun we find all *hypernyms* in the associated knowledge graph. Next, we add the most frequent n such hypernyms to the document-based taxonomy (the number in the third column in Table 1). We identified this step as critical, as the crawl-based knowledge graphs are commonly noisy, and prunning out *uncertain relations* is of high relevance. After performing this for all documents in the corpus, document-based taxonomies are concatenated into corpus-based taxonomy. Next, we perform feature selection, discussed next.

3.1 Feature selection

During feature selection we choose a predefined number of features within the set of features with the goal to *select* the most useful or important features. Hence, from the set of hypernyms which we constructed from the knowledge graph, we choose only top d features (= dimension of the space) based on one of the heuristics described below. **Closeness centrality** of a node is a measure of centrality in a network, calculated as $C(x) = \frac{1}{\sum_y d(y,x)}$, where $d(y,x)$ is the distance (path length) between vertices x and y. The bigger the closeness centrality value a given node has, the closer it is to all other nodes. The **rarest terms** are the most document-specific and are more likely to provide more information than the ones frequently occurring. Hence this heuristic simply takes overall counts of all the hypernyms, sorts them in ascending order by their frequency of occurrence and takes the top d. The **mutual information** between two random discrete variables represented as vectors X_i (the i-th hypernym feature) and Y (the target binary class) is defined as follows:

$$MI(X_i, Y) = \sum_{x,y \in \{0,1\}} p(X_i = x, Y = y) \log_2 \frac{p(X_i = x, Y = y)}{p(X_i = x)p(Y = y)}$$

where $p(X_i = x)$ and $p(Y = y)$ correspond to marginal distributions of the joint probability distribution of X_i and Y. Tax2vec computes the mutual information (MI) between all hypernym features and a given class. So for each target class a vector of mutual information scores is obtained, corresponding to MI between individual hypernym features and a given target class. Then the MI scores for each target class are summed up and the final vector is obtained. The features are sorted by MI scores in descending order and the first d features are chosen as the final semantic space. The **personalized PageRank** 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. The scores 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) or jumps back to one of the starting nodes (with probability 1-p). In our experiments probability p was set to 0.85. The tax2vec exploits the idea initially introduced in [14], where personalized PageRank scores are computed w.r.t. the terms, present throughout the document space. This way, a graph-based, completely unsupervised ranking is obtained, and is used in similar manner to other feature selection heuristics discussed in the previous paragraphs. In this section we introduce how the knowledge graph is used for semantic feature construction, how the text is being processed prior to that and how are semantic features used after that.

Table 1: Part of the Microsoft Concept Graph. The row is in form of hypernym - hyponym - frequency of relation

social network	facebook	4987
symptom	fever	4966
sport	tennis	4964
fruit	strawberry	4824
activity	fishing	4789

3.2 Microsoft Concept Graph

We are using Microsoft Concept Graph[4] [15] [16] for obtaining the extra semantic information. This large relational graph consists of more than 5.4 million concepts that are a part of more than **80 million triplets**. It was created by harnessing billions of web pages, so it is very general and various, offering a lot knowledge to add to our text we want to classify. It contains mostly IsA relations, which was the part we use to obtain hypernyms for nouns in the input text and enrich the feature vectors by some of them. A part of the downloaded knowledge graph is shown in Table 1. The number in the third column is the count of times this relation was found when creating the knowledge graph, so a frequency of the relation's occurrence. We removed relations that had frequency of one, which immediately reduced the graph approximately to half the size and removed mostly noisy relations. Later we used the NetworkX library [17] to transform the Microsoft Knowledge Graph from bare text to a directed graph. This step makes the subsequent exploitation of the knowledge graph easier.

3.3 Proposed approach extending tax2vec

Firstly, we tokenize each document and assign part-of-speech tags to the tokens with the help of the spaCy library [13]. Then for each noun in the text, we find its hypernyms in the knowledge graph. The number of hypernyms for each noun is a parameter chosen by the user, we choose those hypernyms based on the highest frequencies of relation between the current noun and the hypernyms. As shown later in the paper, bigger number of hypernyms does not help a lot, but increases execution time significantly, so it is more sensible to choose a smaller number. Then we create a document-based taxonomy, which is a directed graph where edges are created as hypernym-noun for each hypernym and each noun. We merge the document-based taxonomies into one corpus-based taxonomy (maintaining unique nodes, mergeGraph method in the pseudocode) and on it we perform one of the above mentioned heuristics to choose the best d hypernyms. Those steps are outlined in Algorithm 1.

4 EXPERIMENTS AND RESULTS

This section presents the setting of the experiments and the data sets on which the experiments were conducted. We also describe the metrics used to estimate classification performance.

4.1 Data sets

We conducted the experiments on eight different data sets, which are described below. They were chosen intentionally from different domains and the basic information about them can be seen in Table 2.

[4] https://concept.research.microsoft.com/

Data: corpus, knowledgeGraph, maxHypernyms
corpusTaxonomy = [ ];
foreach *doc* ∈ *corpus* **do**
  documentTaxonomy = [ ];
  tokens = tokenize(*doc*);
  foreach *token* ∈ *tokens* **do**
    if *token is noun* **then**
      edges = knowledgeGraph.edgesFrom(*token*);
      foreach *edge* ∈ *edges* **do**
        if *len*(*documentTaxonomy*) >= *maxHypernyms* **then**
          break;
        documentTaxonomy.add(*edge* ∈ *edges*)
  corpusTaxonomy.mergeGraph(documentTaxonomy)
featureSelection(corpusTaxonomy)
Result: Selected semantic features

Algorithm 1: Semantic feature construction.

Table 2: Data sets used for evaluation of knowledge graph's extra features impact on learning.

Data set	Classes	Words	Unique w.	Documents
PAN 2017 Gender	2	5169966	607474	3600
PAN 2017 Age	5	992742	185713	402
SMSSpam	2	86910	15691	5571
CNN-news	7	1685642	159463	2107
MedicalRelation	18	1136326	66235	22176
Articles	20	5524333	178443	19990
SemEval2019	2	295354	39319	13240
Yelp	5	1298353	88539	10000

PAN 2017 (Gender) Given a set of tweets per user, the task is to predict the user's gender [18].

PAN 2017 (Age) Given a set of tweets per user, the task is to predict the user's age group [19].

CNN News Given a news article (composed of a number of paragraphs), the task is to assign to it a topic from a list of topic categories. [20].

SMS Spam Given a SMS message, the task is to predict whether it is a spam or not. [21].

Medical Relations Given an article with biomedical topic, the task is to predict the relationship between the medical terms annotated. [22].

SemEval 2019 Given a tweet, the task is to predict whether it contains offensive content [23].

Articles Given an web article, the goal is to assign to it a topic. [24].

Yelp Given an review of a restaurant, the goal is to predict the ranking from one to five stars.

Settings. In all the datasets the stop words were removed. Stop words are for example "the", "is", "are" etc. There is no universal list of stop words in NLP research, however we used NLTK (Natural Language Toolkit) [25] for filtering stop words. The documents were tokenized with the help of spaCy's NLP tool. The data sets were divided into 90% training data and 10% test data by using random splits. Number of hypernyms for each noun was 10. We used linear SVM classifier for classification and F_1 measure for performance.

4.2 Results

Figure 1 shows that on some datasets (namely Yelp, PAN 2017 Age, PAN 2017 Gender and on SemEval 2019 and Articles) the extra semantic features constructed from the knowledge graph help in some cases. We compare those results to the classification without any semantic features which is plotted as a grey horizontal line. On the other hand, on the datasets CNN News, Medical Relation and SMS Spam we didn't see any improvement with the addition of semantic features. Figure 2 shows the relation between feature space size and the execution times.

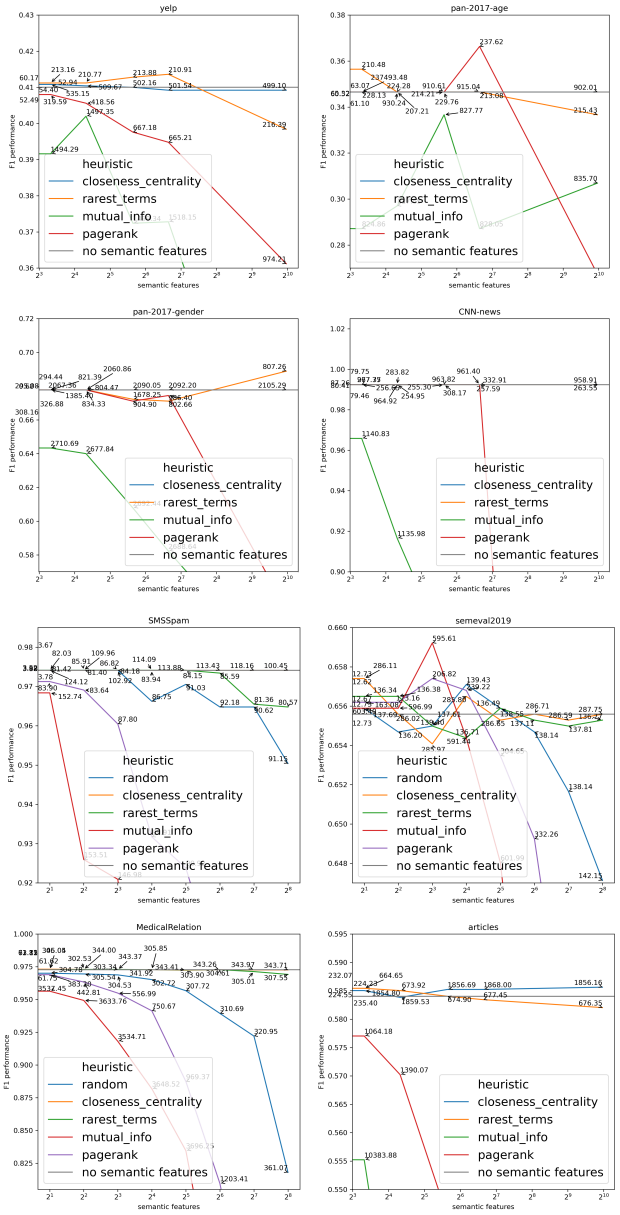

Figure 1: Results of text classification on data sets Yelp, pan-2017-age, pan-2017-gender, CNN News, SMSSpam, SemEval 2019, Medical Relation and Articles with execution times as the numbers in the plot.

5 CONCLUSION

We showed that information from a large, real-life knowledge graph can improve text classification. Our approach aims at short texts like tweets, shorter articles, messages and similar. We firstly process the document with spaCy, find nouns with their corresponding hypernyms, from which we create a taxonomy and from that we later choose the most helpful features with one

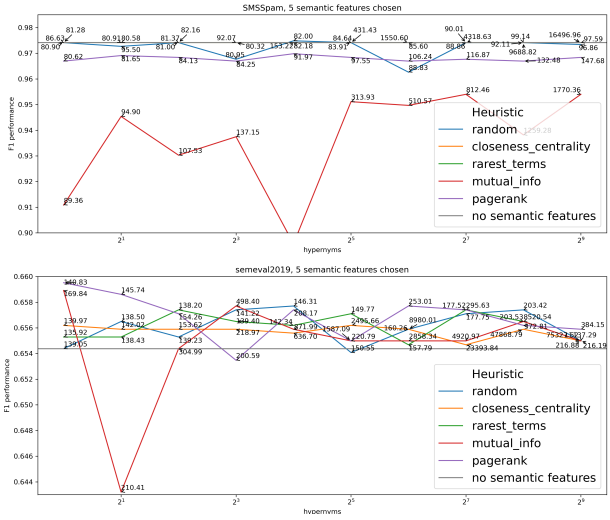

Figure 2: Results of text classification on data sets SMSSpam and SemEval 2019 with execution times as the numbers in the plot.

of the heuristics. The result remains *interpretable*, which is an advantage of this approach. This approach could be potentially improved by performing some type of word sense disambiguation and by finding objects in texts, which consists of more than one word. Further, other knowledge graphs can be used for the hypernym search. Also, because the hypernym search in each document is independent, the documents can be processed in parallel; however, such processing can be memory-intensive, which is to be addressed.

ACKNOWLEDGMENTS

The work of BŠ was financed via a junior research grant (ARRS). This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No. 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The authors acknowledge also the financial support from the Slovenian Research Agency for research core funding for the programme Knowledge Technologies (No. P2-0103), the project TermFrame - Terminology and Knowledge Frames across Languages (No. J6-9372) and the ARRS ERC complementary grant SDM-Open.

REFERENCES

[1] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown. 2019. Text classification algorithms: A survey. *CoRR*, abs/1904.08067.

[2] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge graph embedding: a survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*.

[3] 2020. Tax2vec: constructing interpretable features from taxonomies for short text classification. *Computer Speech & Language*.

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

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation.

[6] T. K. Landauer. 2006. Latent semantic analysis.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. [n. d.] Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.

[8] A. Celikyilmaz, D. Hakkani-Tür, P. Pasupat, and R. Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. In.

[9] M. K. Elhadad, K. M. Badran, and G. I. Salama. 2018. A novel approach for ontology-based feature vector generation for web text document classification.

[10] R. Kaur and M. Kumar. 2018. Domain Ontology Graph Approach Using Markov Clustering Algorithm for Text Classification. *Advances in Intelligent Systems and Computing*, 632.

[11] S. Scott and S. Matwin. 1998. Text classification using WordNet hypernyms. In *Usage of WordNet in Natural Language Processing Systems*.

[12] P. Wang and C. Domeniconi. 2008. Building semantic kernels for text classification using wikipedia. In (August 2008).

[13] M. Honnibal and I. Montani. spaCy 2: natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, (2017).

[14] J. Kralj, M. Robnik-Sikonja, and N. Lavrac. 2019. Netsdm: semantic data mining with network analysis. *Journal of Machine Learning Research*, 20, 32, 1–50.

[15] J. Cheng, Z. Wang, J.-R. Wen, J. Yan, and Z. Chen. 2015. Contextual text understanding in distributional semantic space. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

[16] W. Wu, H. Li, H. Wang, and K. Q. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *ACM International Conference on Management of Data (SIGMOD)*.

[17] A. A. Hagberg, D. A. Schult, and P. J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, 11 –15.

[18] F. Rangel, P. Rosso, M. Potthast, and B. Stein. [n. d.] Overview of the 5th author profiling task at pan 2017: gender and language variety identification in twitter.

[19] F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast, and B. Stein. 2016. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations.

[20] M. Qian and C. Zhai. 2014. Unsupervised feature selection for multi-view clustering on text-image web news data, 1963–1966.

[21] T. A. Almeida and J. M. G. Hidalgo. 2011. Sms spam collection v. 1. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/. (2011).

[22] 2015. Medical information extraction. https://appen.com/datasets/medical-sentence-summary-and-relation-extraction/. (2015).

[23] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

[24] 2019. Text classification 20. https://www.kaggle.com/guiyihan/text-classification-20. (2019).

[25] S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.