Computing Classifier-based Embeddings with the Help of text2ddc

Tolga Uslu, Alexander Mehler, and Daniel Baumartz

Goethe University, Frankfurt am Main, Germany
{uslu, mehler, baumartz}@em.uni-frankfurt.de
https://www.texttechnologylab.org/

Abstract. We introduce a method for computing classifier-based semantic spaces on top of text2ddc. To this end, we optimize text2ddc, a neural network-based classifier for the *Dewey Decimal Classification* (DDC). By using a wide range of linguistic features, including sense embeddings, we achieve an F-score of 87,4%. To show that our approach is language independent, we evaluate text2ddc by classifying texts in six different languages. Based thereon, we develop a topic model that generates probability distributions over topics for linguistic input at the word (sense), sentence and text level. In contrast to related approaches, these probabilities are estimated with text2ddc, so that each dimension of the resulting embeddings corresponds to a separate DDC class. We finally evaluate this *C*lassifier-b*a*sed *S*emantic spac*e* (CaSe) in the context of text classification and show that it improves the classification results.

Keywords: Topic model · Text classification · Sense embeddings.

1 Introduction

We present a model for calculating neural network-based *Classifier-Induced Semantic Space*s (CaSe) using the Dewey Decimal Classification (DDC), that is, an international standard for topic classification in libraries. Based on this model, input units on the sense-, word-, sentence- or text level can be mapped onto the same feature space to compute, for example, their semantic similarity [1, 17]. Such an approach is needed whenever multiresolutional semantic information has to be processed to interrelate, for example, units of different levels of linguistic resolution (e.g., words or phrases to texts). Contrary to related approaches [9, 2] we use classifiers to define the dimensions of CaSe, which are directly labeled by the underlying target class. This has the advantage that embeddings of linguistic units in semantic spaces can be interpreted directly in relation to the class labels.

To this end, we generate several DDC corpora by exploring information from Wikidata, Wikipedia and the Integrated Authority File (*Gemeinsame Normdatei* – GND) of the German National Library. Since Wikipedia is offered for a wide range of languages, such corpora can be created for different languages. In this

paper, we focus on Arabic, English, French, German, Spanish, and Turkish while performing a deeper analysis by example of the German corpus.

Our *Classifier-Induced Semantic Space* (CaSe) utilizes a classifier that is a further development of a feedforward neural network (FNN) which has previously been evaluated in text classification and automatic disambiguation (Anonymous, 2017, 2018). In this paper, we optimize this classifier with respect to feature selection, extend it to the 3rd level of the DDC (comprising up to 641 target classes) and train it for six languages. To this end, we consider a wide range of pre-processing steps including lemmatization, part of speech (POS) tagging, word sense disambiguation (WSD) and sense embeddings.

Our approach is in line with [17] and, thus, disambiguates input words to obtain sense representations as input for calculating sense embeddings. To this end, we disambiguate the entire German Wikipedia and calculate sense embeddings using word2vec [15]. Based on this approach, we achieve the best classification results. This is shown for the 2nd (including up to 98 classes) and the 3rd level of the DDC (including up to 641 classes).

Using the so trained and evaluated DDC-related classifier, we derive a novel model of CaSe. For each sense-, word-, or text-level input CaSe generates a probability distribution over the DDC classes (of either the 2nd or 3rd level). In this way, each input unit is mapped onto an n-dimensional feature vector whose dimensions are uniquely labeled by the corresponding DDC classes. In order to demonstrate the expressiveness of CaSe, we conduct two classification tasks and show that using CaSe-based feature vectors improve any of these classifications.

The paper is organized as follows. Section 2 discusses previous literature related to this article. In Section 3 we describe the models and corpora used for the experimental framework. The main studies and results are presented in Section 4, followed by a discussion of the outcome and an error analysis in Section 5. Section 6 provides some final conclusions and directions for future work.

2 Related Work

In this section, we review related work based on similar approaches, methods or experiments.

[20] and [3] introduce SVM-based topic classifiers regarding the DDC as the target scheme. However, these approaches require a minimum number of words and thus reduce the number of classes. In previous work (Anonymous, 2018), we introduced a topic classifier which outperforms these approaches and even considers the third level of the DDC. Text classification, regardless of the classification scheme, has made significant progress by means of neural networks. [7] and [8] show, for example, that convolutional neural networks perform comparable to the state-of-the-art in text classification. [6] present an FNN-based classifier consisting of only one hidden layer. They show that this classifier is competitive, even though it is many times faster than its competitors.

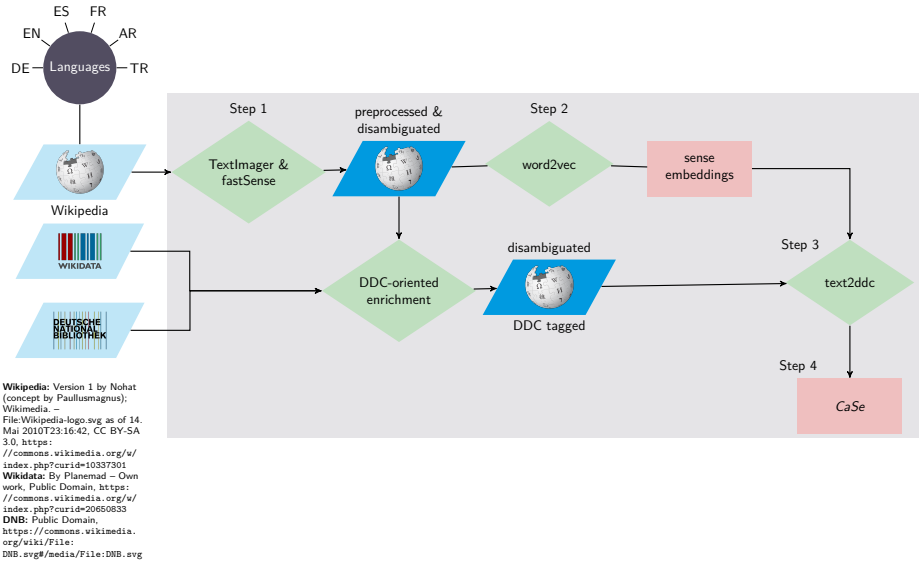

Fig. 1. Flowchart of the experimental framework.

[22] also experiment with neural networks in text classification by considering different classification tasks. In the present study, we adopt their scenarios to evaluate our DDC-based CaSe.

The best-known topic model is probably the one being based on Latent Dirichlet Allocation (LDA) as introduced by [2]. Irrespective of its outstanding applications, a central problem of this approach is that topics being detected are not directly labeled. This makes it difficult to interpret the resulting topic distributions assigned, for example, to input texts. There exist many approaches to automatically labeling topics as detected by LDA by means of heuristic methods [14, 13]. In this way, topic labels are derived from the underlying training corpus. In contrast to this, our topic model refers to the DDC as a pre-established standard used by libraries worldwide for topic classification. Thus, any of the topic labels used by our CaSe can be directly interpreted and related to other classifications for which the DDC has also been used.

To this end, we perform a feature analysis which is related to [11] who analyze the effect of sense embeddings on tasks in Natural Language Processing (NLP). li2015multi compute sense embeddings based on a manually generated dictionary (e.g., WordNet). The embeddings are then assigned to tokens of input texts to disambiguate them. Several classification tasks are conducted to evaluate this approach. They show that sense embeddings lead to improvements compared to word-related embeddings.

[19], [12] and [16] also generate sense embeddings by heuristically utilizing classical word embeddings. [12] use synonyms with only one sense and try to learn sense embeddings based thereon. [19] compute sense embeddings using the

mouse (computer)	**mouse (rodent)**	**ball (sport)**	**ball (dance)**
remote control	cat	hockey puck	masquerade ball
keyboard	rat	bat	clown
Wii remote control	snake	batsman	costume ball
scanner	tick	strike	dinner
video camera	cavy	ball carrier	dance night
touchpad	ant	basket	stag party

Table 1. Nearest neighbors of ambiguous words (both translated from German to English) using cosine similarity operating on the space of sense embeddings.

Language	**articles**	**avg tokens/article**	**classes (2nd)**	**classes (3rd)**
German	15 136	1 228	98	641
English	10 991	2 799	95	603
Arabic	9 910	788	96	591
Turkish	7 998	539	94	566
French	12 406	1 740	96	616
Spanish	13 221	2 053	95	620

Table 2. Training & test corpora for six languages, number of articles, average number of tokens per article, number of target classes on the 2nd and 3rd level for which the apparatus of Figure 1 was trained.

normalized sum of all word embeddings that occur in a dictionary for the current sense. [16] create a graph based on word embeddings and use cluster algorithms to create sense embeddings.

In this paper, we learn sense embeddings similar to the approach of [5]. iacobacci2015sensembed disambiguate the complete Wikipedia with the help of Babelfy[1] and calculate sense embeddings using word2vec [15]. In summary, by looking at different NLP tasks, these approaches show that sense embeddings are effective and usually outperform their word-based competitors.

CaSe have been introduced by [10] who uses *Support Vector Machine*s (SVM) to induce the dimensions of the semantic space. Contrary to this approach, we compute CaSe by means of sense embeddings combined with neural networks.

In this way, we extend previous models of semantic spaces by providing a largely language-independent (Wikipedia- and Wikidata-based) approach that differentiates between the 2nd and the 3rd DDC level to provide semantic spaces of different granularity. This also means that CaSe generates low-dimensional spaces that are much more compact than feature spaces derived from semantic networks.

[1] www.babelfy.org

3 Model

The overall architecture of the experimental framework is depicted in Figure 1. It consists of four steps: in step 1 we use TextImager [4] for preprocessing and fastSense [18] to disambiguate the German Wikipedia. Currently, the focus of fastSense is on the disambiguation of nouns. The disambiguated Wikipedia is then used in Step 2 to create sense embeddings by means of word2vec [15]. The aim is to obtain disambiguated articles and sense embeddings for training a DDC classifier in Step 3 and thus generating text2ddc. For this we enrich the disambiguated Wikipedia articles with DDC information using Wikidata/GND. Next, in Step 4, we utilize text2ddc to generate CaSe for a given input A: in this way, each input unit on the sense-, word- or text-level can be mapped onto an n-dimensional feature vector whose dimensions correspond to DDC classes.

The following subsections briefly describe the models and data used to generate CaSe.

3.1 Step 1 & 2: word sense disambiguation

Word Sense Disambiguation (WSD) is performed by means fastSense [18]. fastSense is trained on the entire German Wikipedia. For this purpose, the Wikipedia's link structure is explored to resolve ambiguous words. The context information at paragraph level of any ambiguous token is given as input. In this way, the neural network based classifier fastSense learns which senses are more likely to be associated in a given context. fastSense achieves an F-score of over 80% in disambiguating the German Wikipedia and state-of-the-art results concerning various SemEval and Senseval WSD tasks. Furthermore, fastSense is very time efficient compared to related taggers. It disambiguates the entire German Wikipedia in about 12 hours. Using the resulting corpus, we created sense embeddings by means of word2vec [15]. Table 1 exemplifies the results by example of two ambiguous words and their nearest neighbors computed by our sense embeddings.

3.2 Step 3: classifier

We introduce a classifier called text2ddc, to classify an input on the sense-, word, sentence- or document-level regarding the DDC as the target classification. text2ddc is based on a FNN and is able to use word embeddings as additional input. We extend text2ddc by alternatively using sense embeddings combined with a disambiguated corpus and many more features (see Table 4).

3.3 Step 4: classification scheme

We use the 2nd and 3rd level of the Dewey Decimal Classification (DDC) as two alternative classification schemes. The DDC includes three levels of thematic resolution: the first level distinguishes 10 main topics, each of which is subdivided

into maximally 10 topics on the 2nd level (summing up to 99 classes), which in turn are subdivided into maximally 10 topics on the 3rd level (summing up to 915 classes). For example, DDC 500 denotes *Natural Sciences* and contains the subtopics *Physics* (DDC 530), *Chemistry* (DDC 540) and *Life Sciences* (DDC 570). On the 3rd level one finds more specific classes such as *Magnetism* (DDC 538) and *Light & Related Radiations* (DDC 535). References to the DDC are provided by Wikidata and the the GND of the German National Library. Since many Wikipedia articles refer to Wikidata or the GND, we were able to explore these articles as training examples of the corresponding DDC classes. In addition, translations provided by both Wikipedia and Wikidata enable the creation of language-specific training corpora by evaluating translation relationships between articles assigned to the DDC and articles for which these assignments do not exist. Table 2 lists the statistics of the training & test corpora that we generated for six target languages.

4 Experiment

In this section, we report on the results of evaluating Step 3 (text2ddc) and Step 4 (CaSe) of our algorithm depicted in Figure 1.

4.1 Evaluating text2ddc

We evaluate text2ddc with regarding the question which features are most successful in DDC-oriented text classification. To this end, we split the corpora described in Section 3.3 into 80%/20% for training/testing. We used the following standard parameters to train the FNN underlying text2ddc:

- number of hidden layers: 1
- hidden layer dimension: 100
- learning rate: 0,1
- update rate: 100
- minimal number of word occurrence: 1
- number of negatives sampled: 5

text2ddc by example of the German Wikipedia We started by testing which parts of an article have the greatest impact on classification when being used for training. That is, for each article in the training corpus, we alternatively trained text2ddc by means of the first sentence, the first paragraph, the first section and the complete article. We did so by additionally testing the number of training epochs as a test parameter. Figure 2 shows that using the entire article as input achieves best results. However, the use of the first paragraph or section as input leads to comparable results. In any event, the larger the input, the better the results. We also see that 100 epochs are minimally required to achieve an F-score of at least 50%.

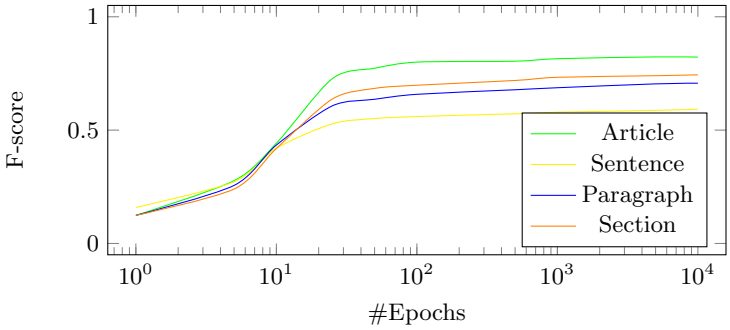

Fig. 2. The effect of different units of the logical document structure (first sentence, first paragraph etc.) and of different numbers of epochs on F-score.

Content	**Epoch 100**	**Epoch 10 000**
Article	0,8	0,822
Section	0,56	0,744
Paragraph	0,658	0,707
Sentence	0,698	0,592

Table 3. F-Scores considering different contents of the corpus.

Now that we detected the optimal input unit and the minimal number of epochs, we tested additional features to improve the classification. This includes information about the lemmas of tokens in input text streams. Figure 3 shows that lemmatization improves the results in all experiments. This makes sense because in this way, one gets more information about the association of DDC classes and text content. For example, `prays`, `prayed` and `praying` are wordforms of the same lexeme *pray*. Under this regime, rare occurrences can still be covered because of being mapped onto the corresponding lemma: even if, for example, the wordform *praying* was never observed in training, it can be processed later on when other wordforms of the same lemma *pray* have been observed. In addition, we also considered lexical information from different sources to measure their effect on classification. This includes Wikipedia categories and Wikidata properties. That is, we used the Wikipedia categories assigned to articles in the training corpus as additional input for training text2ddc and alternativly did the same using the Wikidata properties of the corresponding Wikidata article. Figure 4 shows that Wikipedia categories improve the classification, while Wikidata properties worsen it. Therefore, we excluded Wikidata properties from the subsequent evaluation.

Next, we examined the impact of embeddings, disambiguation and function words. Since the results of these features are very close to each other (see Table 4), we also carried out these experiments with 1 000 epochs. This shows that word embeddings improve the classification, but by means of disambiguation we can

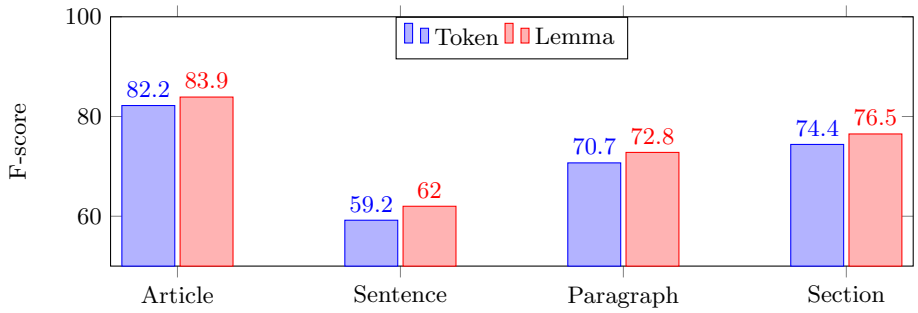

Fig. 3. The effect of lemmatization on classification.

Nr	Features	Epoch 100	1000
1	Lemma	80,8%	
2	1 + Categories	82,1%	
3	2 + Wikidata	81,4%	
4	2 + Embeddings	84,9%	85,5%
5	4 + Disambiguation	84,9%	86,7%
6	5 + No functors	85,0%	87,1%
7	6 + Subwords	85,3%	86,7%
8	6 + Word n-grams	84,7%	

Table 4. Feature analysis by example of the German Wikipedia.

perform even better. Moreover, the removal of function words also improves the classification. Next, we experimented with sub-word units and n-gram features, showing that they do not improve classification. In fact, word n-grams (2 and 3-grams) even worsen the results (see Table 4).

After determining which approach based on the standard parameters achieves the best results, we conducted a parameter study to find the optimal hyperparameters. This includes the following parameters:

- learning rate: 0,2
- update rate: 150
- minimal number of word occurrences: 5
- epochs: 10 000

In this way, we have increased the F-score to 87,4%.

Tackling language independence By means of the language-specific corpora that we generated for different languages (see Section 3.3) we additionally trained text2ddc for Arabic, English, French, Spanish, and Turkish. Table 5 shows that though text2ddc performs worse in the case of the latter five languages compared to German, the results for the 2nd level of the DDC are nevertheless close to 80%.

Language	DDC 2	DDC 3
German	87,4%	78,1%
English	79,8%	72,6%
Arabic	79,8%	68,8%
Turkish	78,9%	67,5%
French	79,4%	68,1%
Spanish	79,7%	70,5%

Table 5. F-scores for different languages for 2nd and 3rd level DDC.

Input	DBpedia	AG News
Text without CaSe	97,89%	89,88%
Text + CaSe (DDC 2)	98,00%	90,18%
Text + CaSe (DDC 3)	98,06%	90,33%

Table 6. F-Scores in the DBpedia and AG News classification tasks.

This makes it possible to determine the topic of linguistic units (senses, words, sentences etc.) of any of these languages to a remarkable degree. Since corpus generation for these languages is straightforward, this also demonstrates that our approach is largely language independent at least what concerns languages that are sufficiently manifested by language specific releases of Wikipedia.

Third Level DDC In order to perform a more detailed analysis, we also analyzed text2ddc with respect to the 3rd level of the DDC. This increased the number of target classes, however, any topic vector is enriched by providing more detailed information. Table 2 shows the frequency distributions of the DDC classes and the number of examples for each language. Table 5 shows the corresponding results: we observe a drop in F-score when switching to the 3rd level, while the case of the German Wikipedia we still perform at about 78% – a result that is similar the one that we observed for the other languages on the 2nd level of the DDC.

4.2 Evaluating CaSe

To show that our DDC-based topic model improves classification, we performed two classification tasks. To this end, we consider two data sets: the DBpedia Ontology Classification Dataset and the AG's news corpus[2]. The DBpedia Ontology Classification Dataset is created by selecting 14 non-intersecting classes from DBpedia 2014[3]. [22] constructed a topic classification dataset using the AG's news corpus by selecting the 4 largest classes.

[2] www.di.unipi.it/ gulli/AG_corpus_of_news_articles.html
[3] www.wiki.dbpedia.org/data-set-2014

For each input text of these two datasets we generated CaSe-related feature vectors. That is, each text is represented by a feature vector of DDC classes of either the 2nd (DDC 2) or 3rd level (DDC 3). Next, we trained a FNN-based classifier that uses these feature vectors as input in addition to the input texts. That is, the FNN is trained in such a way that the words of the input text are presented to the input layer in conjunction with 98 (DDC 2) or 641 (DDC 3) input neurons representing the corresponding DDC classes. For any input word the corresponding input neuron is activated with weight 1. The DDC input neurons are activated using the output values generated by text2ddc for the given text and DDC class. To be independent of the classifier, this experiment was conducted by means of StarSpace [21], a text classification framework developed by Facebook's research team. Table 6 shows the results and the impact of CaSe. The improvements over purely the text-based classifier are not very large, but with such a high classification quality (in the case of DBpedia over 97%), every percentage is important. These two experiments document an impact of CaSe on text classification.

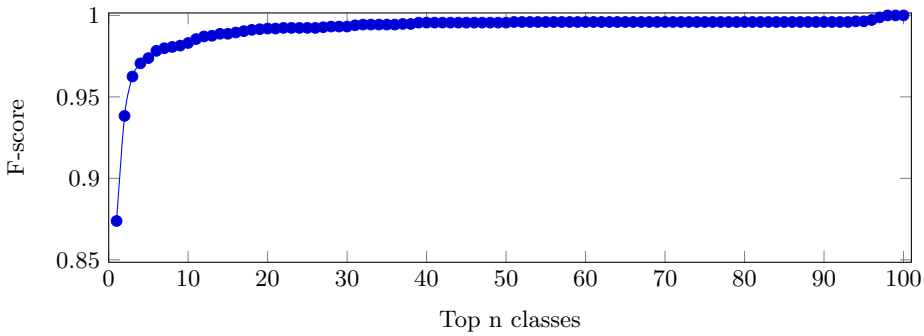

Fig. 4. Analysis of the F-score by considering the top n classes.

5 Discussion

We have increased the performance of the DDC-related classification to over 87% for German. This is remarkable considering that our corpus was automatically generated from Wikipedia articles in conjunction with data from Wikidata and GND. That is, we developed CaSe as an approach that maps any lexical or textual input onto probability distributions over DDC classes. In this way, it is possible to automatically label the topic of this input, referring to interpretable subject names as provided by DDC. On the other hand, topic vectors provided by CaSe can also be used as additional input for classification experiments. We have shown that these vectors improve classification by example of two tasks based on DBpedia and AG News data.

Language	Successful	Unsuccessful
German	3 364,40	3 066,72
English	3 031,78	2 503,29
Arabic	880,36	684,28
Turkish	570,44	395,32
French	2 119,06	1 768,47
Spanish	1 808,43	1 237,92

Table 7. Tokens per example. A comparison between successful and unsuccessful classification.

5.1 Error analysis

We additionally performed an error analysis of the DDC classification. We examined which classes are most often misclassified. On the other hand, we also investigated the performance when considering the top n predicted classes. In the previous experiments we only selected the class with the highest score. Here we discovered that we already achieve a score of over 96% when considering the top 3 classes (see Figure 4). We also discovered that the most common errors occur since some classes are quite similar. One of the most common mistakes is the tagging of DDC 590 (Zoological sciences) instead of DDC 560 (Paleontology). This is very comprehensible, because animals are mentioned in both topics. Another common mistake is the classification of DDC 530 (Astronomy & allied sciences) instead of DDC 520 (Physics). Thus, we recognize that the remaining errors are not necessarily actual errors, but rather similar topics which still might have a high probability.

We also analyzed the average number of words in a successful and unsuccessful classification (see Table 7. Here we discovered that the unsuccessful classifications always contain fewer words on average than the successful ones. The more data the classifier receives, the better is the classification.

6 Conclusion

We have presented a neural network based classifier to categorize DDC classes. For this we have used various features and resources to achieve the best possible classification. This includes POS tagging, lemmatizing and disambiguating the German Wikipedia. Together with this information, we have managed to achieve a classification quality of over 87%. Considering the top three classes, we even exceed 96%. For a given text, the classifier generates a probability distribution over the DDC classes and thus a vector. This vector can be used as input for other classification tasks and we have shown that improvements can be achieved.

We have also trained the classifier in English, Spanish, French, Arabic and Turkish in order to be able to use these vectors in different languages. In future work we would optimize the classifiers of the different languages in the same way

we did for the German version. We offer the classifier (text2ddc) and the DDC topic model (CaSe) as API for all above mentioned languages via GitHub.

References

1. Bär, D., Biemann, C., Gurevych, I., Zesch, T.: UKP: Computing semantic textual similarity by combining multiple content similarity measures. In: Proc. of SemEval '12. pp. 435–440. Stroudsburg (2012)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
3. vor der Brück, T., Eger, S., Mehler, A.: Complex decomposition of the negative distance kernel. In: IEEE International Conference on Machine Learning and Applications (2015)
4. Hemati, W., Uslu, T., Mehler, A.: Textimager: a distributed uima-based system for nlp. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations. pp. 59–63 (2016)
5. Iacobacci, I., Pilehvar, M.T., Navigli, R.: Sensembed: Learning sense embeddings for word and relational similarity. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). vol. 1, pp. 95–105 (2015)
6. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
7. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
8. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
9. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological Review **104**(2), 211–240 (1997)
10. Leopold, E.: Models of semantic spaces. In: Mehler, A., Köhler, R. (eds.) Aspects of Automatic Text Analysis, Studies in Fuzziness and Soft Computing, vol. 209, pp. 117–137. Springer, Berlin/Heidelberg (2007)
11. Li, J., Jurafsky, D.: Do multi-sense embeddings improve natural language understanding? arXiv preprint arXiv:1506.01070 (2015)
12. Li, Q., Li, T., Chang, B.: Learning word sense embeddings from word sense definitions. In: Natural Language Understanding and Intelligent Applications, pp. 224–235. Springer (2016)
13. Magatti, D., Calegari, S., Ciucci, D., Stella, F.: Automatic labeling of topics. In: Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on. pp. 1227–1232. IEEE (2009)
14. Mei, Q., Shen, X., Zhai, C.: Automatic labeling of multinomial topic models. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 490–499. KDD '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1281192.1281246, http://doi.acm.org/10.1145/1281192.1281246
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

16. Pelevina, M., Arefyev, N., Biemann, C., Panchenko, A.: Making sense of word embeddings. arXiv preprint arXiv:1708.03390 (2017)
17. Pilehvar, M.T., Navigli, R.: From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. Artificial Intelligence **228**, 95–128 (2015)
18. Uslu, T., Mehler, A., Baumartz, D., Henlein, A., Hemati, W.: fastsense: An efficient word sense disambiguation classifier. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018) (2018)
19. Vial, L., Lecouteux, B., Schwab, D.: Sense embeddings in knowledge-based word sense disambiguation. In: 12th International Conference on Computational Semantics (2017)
20. Waltinger, U., Mehler, A., Lösch, M., Horstmann, W.: Hierarchical classification of OAI metadata using the DDC taxonomy. In: Bernardi, R., Chambers, S., Gottfried, B., Segond, F., Zaihrayeu, I. (eds.) Advanced Language Technologies for Digital Libraries (ALT4DL), pp. 29–40. LNCS, Springer (2011)
21. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: Starspace: Embed all the things! CoRR **abs/1709.03856** (2017), http://arxiv.org/abs/1709.03856
22. Zhang, X., LeCun, Y.: Text understanding from scratch. CoRR **abs/1502.01710** (2015), http://arxiv.org/abs/1502.01710