Improving Word Embeddings Using Kernel PCA

Vishwani Gupta **Sven Giesselbach** **Stefan Rüping** **Christian Bauckhage**
Fraunhofer IAIS, Sankt Augustin, Germany
firstname.lastname@iais.fraunhofer.de

Abstract

Word-based embedding approaches such as Word2Vec capture the meaning of words and relations between them, particularly well when trained with large text collections; however, they fail to do so with small datasets. Extensions such as fastText reduce the amount of data needed slightly, however, the joint task of learning meaningful morphology, syntactic and semantic representations still requires a lot of data. In this paper, we introduce a new approach to warm-start embedding models with morphological information, in order to reduce training time and enhance their performance. We use word embeddings generated using both word2vec and fastText models and enrich them with morphological information of words, derived from kernel principal component analysis (KPCA) of word similarity matrices. This can be seen as explicitly feeding the network morphological similarities and letting it learn semantic and syntactic similarities. Evaluating our models on word similarity and analogy tasks in English and German, we find that they not only achieve higher accuracies than the original skip-gram and fastText models but also require significantly less training data and time. Another benefit of our approach is that it is capable of generating a high-quality representation of infrequent words as, for example, found in very recent news articles with rapidly changing vocabularies. Lastly, we evaluate the different models on a downstream sentence classification task in which a CNN model is initialized with our embeddings and find promising results.

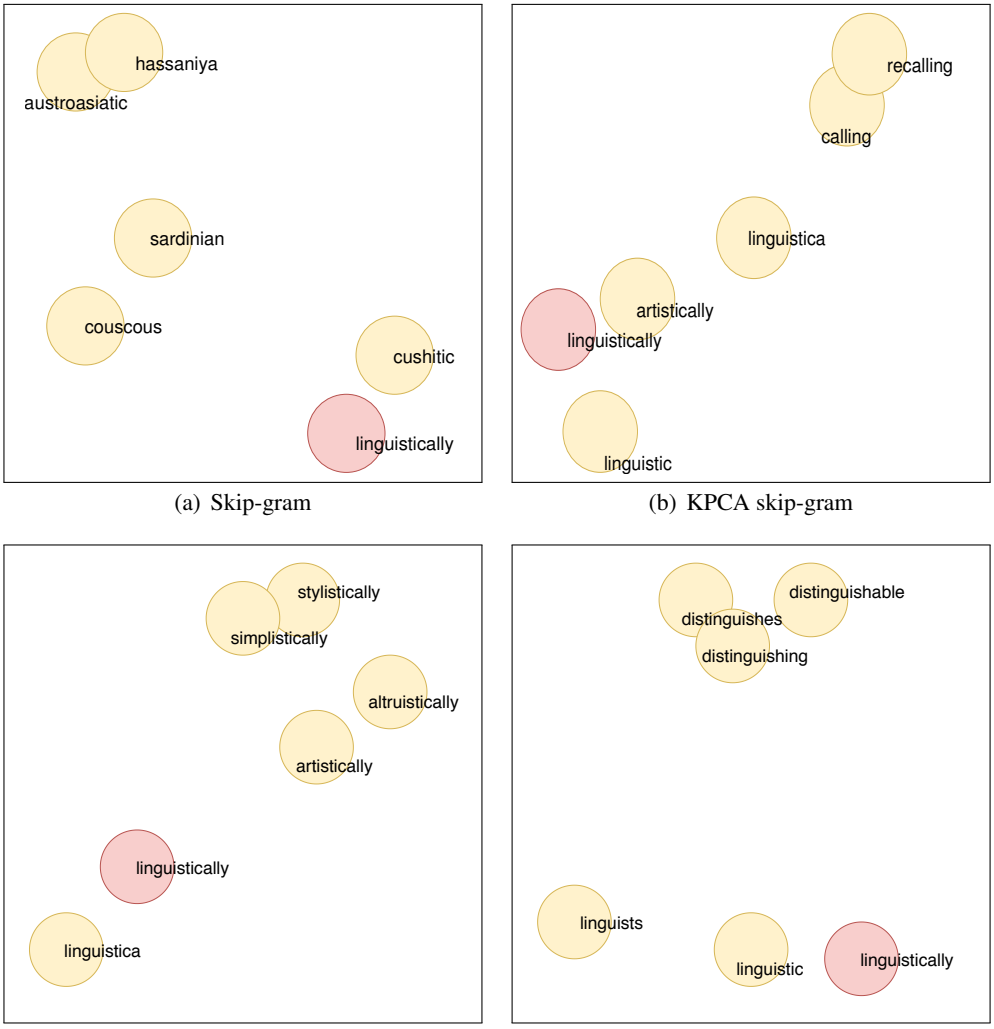
1 Introduction

Continuous vector representations of words learned from unstructured text corpora are an effective way of capturing semantic relationships among words. Approaches to computing word embeddings are typically based on the context of words, their morphemes, or corpus-wide co-occurrence statistics. As of this writing, arguably the most popular approaches are the *Word2Vec* skip-gram model (Mikolov et al., 2013a) and the fastText model (Bojanowski et al., 2017). The skip-gram model generates embeddings based on windowed word contexts. While it incorporates semantic information, it ignores word morphology. Yet, the latter might be beneficial especially for morphologically rich languages such as German and Turkish. Bojanowski et al. (2017) therefore introduced *fastText* which builds on the Word2Vec approach but also incorporates morphology by considering sub-word units and representing a word by a sum of its character n-grams as well as the word itself.

To learn high-quality embeddings, Word2Vec requires huge text corpora with billions of words and still fails to generate high-quality vector representations for less frequent or unknown words. Although fastText improves the results by incorporating subword information, it still fails in many cases. This is particularly evident in the news domain where frequently new words such as names occur over time which, in turn, impacts the performance of downstream applications. In this paper, we therefore propose an alternative approach which not only makes use of morphological information but also performs well when trained on smaller datasets or domains with rapidly changing vocabulary. Research questions we answer in this paper are:

1. Can high-quality word embeddings be trained on small datasets?
2. Can high-quality embeddings be generated for infrequent words?
3. Can the use of morphological information increase the efficiency of learning semantic and syntactic similarities?

Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pages 200–208
Florence, Italy, August 2, 2019.



(a) Skip-gram (b) KPCA skip-gram

(c) FastText (d) KPCA fastText

Figure 1: Visualization of 5-nearest neighbors for the word “Linguistically”.

2 Related work

Mikolov et al. (2013a) proposed log-bilinear models to learn vector representations of words from the context in which they appear in large corpora. These are the Continuous Bag-of-Words Model (CBOW) and the Continuous Skip-gram Model (skip-gram) which predict target words from source context words and source context words from target words, respectively. An extension proposed by Mnih and Kavukcuoglu (2013) involves training lightweight log-bilinear language models with noise-contrastive estimation and achieves results comparable to the best previous models with one quarter of the training data and in less computing time.

There are some recent works which try to incorporate morphological structures into the computation of embeddings. Soricut and Och (2015) learn vector representation of morphological transformations and are able to obtain representations for unseen words. Cotterell et al. (2016) presented a morpheme-based post-processor for word embeddings. They proposed a Gaussian graphical model which can be extended to continuous representations for unknown words as well as helps in smoothing the representations of the observed words in the training dataset. Bhatia et al. (2016) proposed a new unified probabilistic framework in which they combine morphological and distributional information. The word embeddings act as a latent variable for which morphological information provides a prior distribution. This in turn condition a likelihood function over an observed dataset. Bojanowski et al. (2017) proposed *fastText*, an extension of the skip-gram model, which learns word representations by in-

cluding sub-word information. This is achieved by not only representing words with vectors but also the subword parts they consist of. Word vector representations are finally built as the sum of their sub-word vectors and their own representation.

3 KPCA-based skip-gram and fastText models

In this section, we propose a general extension to word embedding methods which we evaluate on the skip-gram model as well as on the fastText model. We propose pre-training embeddings with a kernel PCA computed on word similarity matrices, generated using a string similarity function, for words in a vocabulary and then injecting the pre-trained embeddings in the Word2Vec and fastText embeddings by initializing them with the KPCA word and subword embeddings. This seamlessly incorporates sub-word structures in Word2Vec and yields a better starting point for fastText training. It is especially useful for morphologically rich languages because their semantically similar words often share some common morphemes such as roots, affixes, and syllables.

3.1 Kernel PCA on string similarities

Embedding words according to morphological similarities can be seen as a clustering problem in a higher dimensional feature space which can be tackled using Kernel PCA (Schölkopf et al., 1997), a nonlinear form of principal component analysis. Suppose a vocabulary V of words w_i, a string similarity measure $\mathcal{S}$ (e.g. the n-gram similarity (Brito et al., 2017)), and a non-linear kernel function $\mathcal{K}$ (e.g. the Gaussian) to be given. This allows us to compute a $|V| \times |V|$ word similarity matrix $\mathbf{K}$ where

$$K_{ij} = \mathcal{K}\big(\mathcal{S}(w_i, w_j)\big) \tag{1}$$

Centering this kernel matrix (Schölkopf et al., 1997) yields a feature space representation of words in V, because column vector $\mathbf{k}_i$ of $\mathbf{K}$ can be seen as a $|V|$-dimensional representation of w_i. Performing PCA in this feature space then allows for selecting the first $d < |V|$ nonlinear principal components $\mathbf{v_i}$ which, in turn, allow for projecting word vectors into lower dimensional spaces. Using projection matrix,

$$\mathbf{P} = \left[\frac{\mathbf{v_1}}{\lambda_1}, ..., \frac{\mathbf{v_d}}{\lambda_d}\right] \tag{2}$$

generated by selecting d eigenvectors $\mathbf{v_1}$ to $\mathbf{v_d}$ corresponding to the highest eigenvalues, λ_1 to λ_d, the d-dimensional projections are

$$\mathbf{e}_i = \mathbf{P}^\intercal \mathbf{k}_i. \tag{3}$$

3.2 Models with KPCA embeddings

Computing Kernel PCA for a large vocabulary is computationally demanding. We thus restrict our vocabulary V to contain only the most frequent words of a text corpus. For any *new* or *out of vocabulary word* s_{new} not contained in V, we can compute its kernel vector

$$\mathbf{k}_{new} = \mathcal{K}\big(\mathcal{S}(s_{new}, V)\big) \tag{4}$$

and obtain a lower dimensional representation as

$$\mathbf{e}_{new} = \mathbf{P}^\intercal \mathbf{k}_{new}. \tag{5}$$

Note that word embeddings computed this way only encode morphological similarity. To incorporate semantic similarities, we initialize Word2Vec and fastText models with the pre-trained KPCA embeddings. We train the Word2Vec skip-gram model with negative sampling (Mikolov et al., 2013b) on our morphological embeddings using the C implementation of Word2vec package[1].

To initialize the fastText model with morphological embeddings, we also compute the KPCA vectors for the subword units in the dictionary, as fastText also has vector representations for character n-grams contained in words. We train the fastText model initialized with our morphological embeddings using the C++ implementation of fastText [2]. After training both the models, we obtain embeddings encoded with semantic, syntactic and morphological similarities whose practical merits we evaluate in the next section.

4 Experimental Results

To evaluate our models' performance when trained on datasets of different sizes, we consider English and German datasets such as *Text8* [3], *20 News*

[1] https://code.google.com/p/word2vec/

[2] https://github.com/facebookresearch/fastText/

[3] http://mattmahoney.net/dc/text8.zip

Dataset name	Language	Corpus size (words)	Vocab Size (words)
20 Newsgroups	English	1 million	$\approx 19,000$
Text8	English	10 million	$\approx 70,000$
English Wiki 2016	English	1,192 million	$\approx 330,000$
German news 2013	German	183 million	$\approx 247,000$

Table 1: Details of the datasets used for evaluation.

groups [4], *English Wiki 2016* and *German news 2013* [5] for training. Each dataset contains articles crawled from news websites or Wikipedia. These raw text corpora contain a large amount of irrelevant text and are pre-processed using a script by Mahoney[6]. In Table 1, we list all the datasets that we have used to train the models.

4.1 Baseline

One of our main results is the observation that the KPCA fastText model and the KPCA skip-gram model generate high-quality word embeddings even when trained only on small datasets when compared to fastText or skip-gram model respectively. To compare how well our models perform in comparison to the original skip-gram model and fastText model, we consider both of them as the baselines in all our experiments and use the same parameters and datasets for generating and evaluating embeddings for all models.

4.2 Evaluation

We evaluate our models using *intrinsic evaluation tasks* which assess how well the vectors capture meanings of and relationships between words. In particular, we evaluate all the models with respect to

1. *Word similarity tasks* which include finding a word's nearest neighbors.

2. *Word analogy tasks* which include calculating the semantic and the syntactic similarities between words and their relations.

3. *Performance in a downstream application* which illustrates how well the embeddings work for subsequent processing steps such as a sentence classification task (Kim, 2014).

4.3 Word similarity Evaluation

Word similarity tasks evaluate word embeddings in terms of their k-nearest neighbors. For selected words, we show nearest neighbors according to cosine similarity for vectors trained using the proposed models as well as the baseline models. Here we illustrate how the models performed for frequent as well as for infrequent words. Table 2 presents examples for all the models obtained after being trained for an epoch on the small *Text8* dataset.

The table illustrates that for the frequent words, all the models learn a good representation and are able to produce relevant nearest neighbors. For the infrequent words, the skip-gram did not learn a very good representation as there are not enough examples for the word to learn from. In these cases the nearest neighbor of the skip-gram model are not very meaningful, e.g. it places firecracker close to "prochnow", while the KPCA fastText model places it closer to "cracker" and "fire" related words. Since the fastText model uses sub-word information, it achieves better performance at this task compared to the skip-gram model. It finds meaningful neighbors for "placental", however it fails for words such as "cruel". KPCA skip-gram gets a warm start with the morphological information learned from KPCA, which helps in learning a better representation for scarce words, thus producing better k-nearest neighbors for words such as "cruel".

When we compare KPCA skip-gram with KPCA fastText, we observe that KPCA fastText generally generates better neighbors. We assume this is because of the fact that it benefits from the fastText approach of jointly refining subword and word representations. Comparing fastText with the better initialized KPCA fastText model, KPCA produces decidedly better neighbors, especially for "scrubbing", "firecracker", "linguistically" and

[4] http://qwone.com/~jason/20Newsgroups/

[5] http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2013.de.shuffled.gz

[6] http://mattmahoney.net/dc/textdata.html

Word (frequency)	Model	k-nearest neighbors (sorted by similarity)
three (114,775)	Skip-gram	four, five, seven, eight, runways
	KPCA skip-gram	four, seven, five, eight, nine
	fastText	four, five, seven, zero, eight
	KPCA fastText	seven, five, four, eight, nine
history (12,623)	Skip-gram	overview, timeline, prehistory, origins, beginnings
	KPCA skip-gram	article, origins, pamphlets, references, offshoot
	fastText	prehistory, historique, historica, historiques, historiography
	KPCA fastText	historical, prehistory, histories, historiography, historic
april (3,069)	Skip-gram	june, march, august, february, november
	KPCA skip-gram	august, march, june, february, december
	fastText	february, january, october, september, november
	KPCA fastText	february, august, june, october, november
biblical (703)	Skip-gram	talmud, josephus, apocrypha, tanakh, commentaries
	KPCA skip-gram	judaica, mythical, mexica, metaphysical, micah
	fastText	evangelical, biblically, bibliographical, noncanonical, mythological
	KPCA fastText	bible, bibles, bibl, testament, biblically
moscow (622)	Skip-gram	warsaw, armistice, versailles, hostage, daoud
	KPCA skip-gram	kabul, prague, beirut, bonn, cpsu
	fastText	mosby, mokhehle, moonwalks, mocha, rsfsr
	KPCA fastText	borisovich, soviet, helsinki, denisovich, warsaw
cruel (140)	Skip-gram	injustice, urge, fears, zeal, appease
	KPCA skip-gram	cruelty, foes, lawful, imbued, idols
	fastText	cruzi, crusoe, crux, cruijff, duel
	KPCA fastText	cruelty, cruelly, ruelle, cruzi, duel
linguistically (46)	Skip-gram	cushitic, hassaniya, couscous, austroasiatic, sardinian
	KPCA skip-gram	linguistic, recalling, artistically, calling, linguistics
	fastText	simplistically, altruistically, stylistically, artistically ,linguistica
	KPCA fastText	linguistic, linguists, distinguishes, distinguishing, distinguishable
placental (30)	Skip-gram	intestine, condense, spikes, greasy, sideways
	KPCA skip-gram	placenta, centaurus, labiodental, centaur, centaurs
	fastText	placentals, placenta, dental, placement, placement, segmental
	KPCA fastText	placentals, placenta, parental, placement, concentrates
firecracker (5)	Skip-gram	prochnow, caff, gwen, hillis, horovitz
	KPCA skip-gram	mccracken, racked, wracked, rackets, racket
	fastText	cracker, nutcracker, acker, thacker, skywalker
	KPCA fastText	cracker, crackers, fireplace, firestrom, firewall
scrubbing (5)	Skip-gram	underpowered, transceivers, refineries, heliport, gasification
	KPCA skip-gram	ingot, ingots, xing, mcing, plying
	fastText	scrying, dubbing, rubbing, ebing, screwing
	KPCA fastText	rubbing, clubbing, scrubbed, scraping, scrying

Table 2: The $k = 5$-nearest neighbors of word embeddings in $\mathbb{R}^{128}$, trained on the *Text8* dataset.

"cruel". In figure 1, we can visualize the t-SNE 2-D representation of the nearest neighbors for the word "linguistically".

Epoch No.	Model	Total	Semantic	Syntactic
1	Skip-gram	7.84%	3.92%	11.24%
	KPCA skip-gram	15.74%	2.96%	26.71%
	fastText	36.89%	0.89%	67.80%
	KPCA fastText	38.04%	1.12%	69.76%
2	Skip-gram	17.09%	9.59%	23.50%
	KPCA skip-gram	21.14%	7.26%	33.07%
	fastText	41.19%	1.60%	75.19%
	KPCA fastText	40.48%	2.10%	73.43%
5	Skip-gram	24.21%	19.28%	28.45%
	KPCA skip-gram	26.20%	15.56%	35.34%
	fastText	43.60%	5.51%	76.31%
	KPCA fastText	43.42%	7.37%	74.39%
10	Skip-gram	26.68%	24.61%	28.45%
	KPCA skip-gram	28.65%	22.22%	34.18%
	fastText	44.49%	12.10%	71.75%
	KPCA fastText	44.68%	13.17%	72.29%

Table 3: Analogy accuracies of embeddings in $\mathbb{R}^{128}$ trained for different epochs on *Text8* dataset.

Epoch No.	Model	Total	Semantic	Syntactic
1	Skip-gram	0.18%	0.20%	0.17%
	KPCA skip-gram	15.47%	0.66%	19.80%
	fastText	3.07%	0.26%	3.89%
	KPCA fastText	33.66%	0.79%	43.26.%
2	Skip-gram	0.99%	1.26%	0.91%
	KPCA skip-gram	14.83%	1.65%	18.68%
	fastText	28.52%	0.33%	36.76%
	KPCA fastText	45.85%	1.19%	58.91%
3	Skip-gram	0.96%	1.46%	0.81%
	KPCA skip-gram	14.07%	1.92%	17.62%
	fastText	44.84%	0.79%	57.71%
	KPCA fastText	47.44%	1.39%	60.90%

Table 4: Analogy accuracies of embeddings in $\mathbb{R}^{128}$ trained for different epochs on *20 Newsgroups* dataset.

4.4 Word Analogy Evaluation

Pre-trained word vectors are available for a dataset of 100 billion words from Google News. Mikolov et al. (2013a) observed that, when word vectors are trained on such a large dataset, they are able to answer very subtle relationships between words. Yet, for the news data or for a small dataset such results cannot be achieved. Warm-starting the models with our KPCA embeddings, however, yields good performance in such settings, too.

To assess accuracies in the word analogy task, we use a comprehensive test set provided by Mikolov et al. (2013a). This test consists of semantic and syntactic similarity questions which include relationships like adjective-to-adverb, currency, plural-verbs, city-in-state, comparative, superlative relationships, and others. A question is assumed to be correctly answered only if the closest word to the vector is exactly the same as the correct word in the question; synonyms are considered as mistakes. In order to use this evaluation to compare our models' results to those of the skip-gram and the fastText models, we train the models on the different datasets shown in Table 1. Results are reported in Tables 3, 4, 5 and 6.

A comparison of 300-dimensional and 128-dimensional embeddings on the analogy tasks on the *text8* and *20-Newsgroups* datasets showed that all models (including baselines) perform best when we picked 128-dimensional embeddings. For the sake of simplicity we used 128-dimensions in all tasks. From Table 4, it is evident that our KPCA fastText model outperforms the skip-gram as well as the fastText model when trained on a small dataset. KPCA skip-gram as well as KPCA fastText models have better accuracies for both semantic and syntactic questions in the initial epochs compared to their cold-start counterparts. One question arising in this context is whether the skip-

gram or the fastText models can also learn from smaller datasets. The answer for the *20 Newsgroup* as well as for the *text8* datasets is "yes", but only if they are trained for several epochs.

The results for the *20 Newsgroups* dataset in Table 4 also show that the skip-gram models completely fail to learn analogies on this dataset. The KPCA fastText embeddings benefit from their warm-start and show a quicker convergence rate. We make the following observations from the accuracies obtained after each epoch of training. During the 1^{st} epoch, the skip-gram and the fastText model do not perform well. However, after the 2^{nd} epoch, the fastText model starts performing better on the syntactic questions. Meanwhile KPCA skip-gram and KPCA fastText models still achieve higher accuracies than the respective skip-gram and fastText models. Hence, considering accuracies from the initial epochs, we can conclude that training of our model converges faster than the training of the fastText model and the skip-gram model.

From Table 3, we observe that the skip-gram model always seems to perform better on the semantic questions but when we compare these accuracies with the nearest neighbors results from Table 2, it can be observed that although KPCA models seem to work badly on semantic tasks, they generate better k-nearest neighbors than the respective skip-gram and fastText models.

We also compare the accuracies achieved by the models when trained for one epoch on a large data set, namely *English Wikipedia* dataset. The results in Table 5, illustrate the performance of the different models when training them on a large training dataset size. When compared to fastText, KPCA skip-gram performs better on semantic questions, but worse on syntactic questions. Noticeably KPCA fastText performs better on semantic questions than all the other models. However plain fastText outperforms it on the syntactic questions. The overall accuracy of fastText is also slightly higher than for KPCA fastText model.

We also report accuracies for the analogy task when the models are trained on the *German news* dataset for morphologically rich German language. We use the German version of the semantic/syntactic analogy dataset, introduced by (Köper et al., 2015) for evaluation. Table 6 shows how different models perform on the analogies tasks. We note that morphological information significantly improves the syntactic tasks when we compare KPCA fastText and fastText with KPCA skip-gram and skip-gram. While the semantic accuracy degrades for both KPCA fastText and fastText. Initializing the skip-gram and fastText with KPCA embeddings has improved the performance of both the models. The KPCA skip-gram model shows the overall best performance. This shows that the initialization of the models with morphological information, is beneficial for the German language as well.

Epochs	Model	Total	Sem	Syn
1	Skip-gram	70.95%	85.29%	66.67%
	KPCA skip-gram	75.00%	85.29%	71.93%
	fastText	81.43%	82.35%	80.46%
	KPCA fastText	80.41%	88.24%	78.07%

Table 5: Analogy accuracies of embeddings in $\mathbb{R}^{128}$ trained for one epoch on *English Wiki 2016* dataset.

Epochs	Model	Total	Sem	Syn
1	Skip-gram	34.43%	35.89%	31.69%
	KPCA skip-gram	35.71%	37.72%	31.94%
	fastText	29.56%	14.34%	58.20%
	KPCA fastText	30.15%	14.54%	59.56%

Table 6: Analogy accuracies of embeddings in $\mathbb{R}^{128}$ trained for one epoch on *German news 2012* dataset.

Dataset	Model	Accuracies
20 Newsgroups	Skip-gram	72.82%
	KPCA Skip-gram	73.57%
	fastText	72.07%
	KPCA fastText	72.73%
english Wiki 2016	Skip-gram	73.90%
	KPCA Skip-gram	74.56%
	fastText	73.96%
	KPCA fastText	74.09%

Table 7: Sentence classification task trained using pretrained embeddings obtained from different models after 10 epochs of training.

4.5 Evaluation of performance on downstream applications

Finally, we investigate how well the embeddings obtained from the different models and different datasets perform on a downstream task in a neural network architecture. We choose the convo-

lutional neural network proposed in (Kim, 2014) and evaluate it with the embeddings in a sentence classification task. We initialize the CNN with the embeddings obtained from the different embedding models and keep the embeddings static during training. Initializing the word vectors with the pre-trained word embeddings instead of random embeddings improves the performance as noted by (Collobert et al., 2011) and (Iyyer et al., 2014).

In our experiment, the classification task is a sentiment classification task, i.e. detecting whether reviews are positive or negative. The dataset used for it, consists of movie reviews[7] with one sentence per review. We use embeddings generated by training the embedding models on the *20 Newsgroups* and *English Wikipedia* datasets. The CNN network is trained for 10 epochs.

In Table 7, we list the accuracies for the model trained with different embeddings obtained from the two datasets. In both cases, the model initialized with embeddings generated using KPCA skip-gram model and the KPCA fastText outperform the models initialized with the respective cold-start embeddings, albeit with a small margin. The models initialized with the KPCA skip-gram model achieve the best results in both cases.

5 Conclusion

In this paper, we explored a simple method to improve models for computing word embeddings and evaluated it with the popular skip-gram and fastText models.

Our approach relies on string similarity matrices computed from small vocabularies which, in a first step, are subjected to kernel PCA (KPCA) in order to generate non-linear, morphologically informed word embeddings. In a second step, the KPCA-based vector representations of words are used as input to the skip-gram model in order to obtain embeddings that also account for word contexts.

In practical experiments, we evaluated the quality of our embeddings using intrinsic measures such as word similarity and word analogy. In our experiments the KPCA skip-gram and KPCA fastText were found to outperform the original continuous skip-gram and fastText model. In particular, we found that the continuous skip-gram model can learn similarity among words only when it has seen a sufficiently large number of examples. When feeding the models with morphologically informed vector representations of words, they seem to be able to learn from a better starting point when computing semantically informed embeddings. Using KPCA fastText or KPCA skip-gram model, we found that it is possible to obtain high-quality word vectors even when training with small datasets and fewer epochs.

[7] http://www.cs.cornell.edu/people/pabo/movie-review-data/

6 Future Work

Future work concerns deeper analysis of how to choose words in the vocabulary to construct the projection matrix used to generate Kernel PCA embeddings. We would like to explore different string similarity functions which would help in creating better clusters of the similar words.
We would also like to extend our experiments to different embeddings such as GloVe as well as to further downstream tasks such as Named Entity Recognition or Relation Extraction.

7 Acknowledgements

This work was funded by the German Federal Ministry of Education and Research under the Competence Center Machine Learning Rhine-Ruhr ML2R, Foerderkennzeichen 01S18038B.

References

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. *arXiv preprint arXiv:1608.01056*.

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

Eduardo Brito, Rafet Sifa, Kostadin Cvejoski, Cesar Ojeda, and Christian Bauckhage. 2017. Towards german word embeddings: A use case with predictive sentiment analysis. In *Proceedings of Data Science, Analytics, and Applications*, pages 59–62.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the*

54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1651–1660.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1113–1122.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and semantic structure of continuous word spaces. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1997. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 1627–1637.