KPCA Embeddings: an Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences

A Use Case for Words and DNA Sequences

Eduardo Brito, Rafet Sifa, and Christian Bauckhage

Fraunhofer IAIS
Schloss Birlinghoven, Sankt Augustin, Germany
{Eduardo.Alfredo.Brito.Chacon,Rafet.Sifa,Christian.Bauckhage}
@iais.fraunhofer.de
https://multimedia-pattern-recognition.info

Abstract. Most of the well-known word embeddings from the last few years rely on a predefined vocabulary so that out-of-vocabulary words are usually skipped when they need to be processed. This may cause a significant quality drop in document representations that are built upon them. Additionally, most of these models do not incorporate information about the morphology of the words within the word vectors or if they do, they require labeled data. We propose an unsupervised method to generate continuous vector representations that can be applied to any sequence of finite domain (such as text or DNA sequences) by means of kernel principal component analysis (KPCA). We also show that, apart from their potential value as a preprocessing step within a more complex natural language processing system, our KPCA embeddings also can capture valuable linguistic information without any supervision, in particular word morphology of German verbs. When they are applied to DNA sequences, they also encode enough information to detect splice junctions.

1 Introduction

Machine learning approaches for natural language processing (NLP) generally demand a numeric vector representation for words. We can distinguish any two different words from a fixed vocabulary by assigning them a one-hot vector, where all entries of the vector are zero-valued but in a single position that identifies the word. This is a very sparse representation that encodes no information about the words but their position in the vocabulary.

A more information-rich alternative to one-hot vectors are the so-called *word embeddings*. They are distributed vector representations, which are dense, low-dimensional, real-valued and can capture latent features of the word [13]. Based on the *distributional hypothesis* [5] (words that appear in similar contexts have similar meanings), they exploit word co-occurrence so that similar words are

In: M. Leyer (Ed.): Proceedings of the LWDA 2017 Workshops: KDML, FGWM, IR, and FGDB.
Rostock, Germany, 11.-13. September 2017, published at http://ceur-ws.org

mapped close to each other in the word vector space. Part of the success of the word embeddings is due to their efficient shallow neural network architectures such as the continuous skip-gram model and the continuous bag of words model [9], widely popularized after the release of `word2vec`[1]. Word embeddings also inspired research in other areas different from NLP to learn vector representations such as node representations from a graph [4].

Although syntax and semantics can be encoded with word2vec embeddings, they do not incorporate morphological information about the word. As a consequence, morphologically similar words may not be nearby in the word vector space. Some approaches make use of existing linguistic resources so that the word embeddings capture not only contextual information but also morphological information[3, 7]. Due to the their dependence on language-specific resources, they will not work for languages whose available linguistic resources are scarce.

The subword-based models such as `FastText` [1, 6] learn indirectly morphology by learning not only word vectors but also n-gram vectors. This enables not only complete unsupervised learning but also the possibility of inferring out-of-vocabulary (OOV) words, which is an important issue for all other approaches mentioned, notably when noisy informal text needs to be processed. Despite these advantages of subword-based models, some morphologically rich languages (where these models are supposed to perform specially well) may contain very long words. This leads to a dramatic increase of the number of necessary n-gram representations, increasing time and space complexity as well.

We propose an alternative unsupervised method by means of kernel principal component analysis (KPCA) that encodes morphology while learning word representations and that generate new vectors for OOV words after training. In addition, our approach is general enough to learn vector representations for any sequence whose elements belong to a fixed predefined finite set. In particular, we test our KPCA embeddings in two different tasks: classifying the verb category for German verbs and detecting splice junctions in DNA sequences.

2 Approach

KPCA indirectly maps vectors to a feature space (of higher dimension) in order to obtain the principal components from that space [12]. No explicit calculation in the feature space is required since we only need to be able to compute the inner product in the feature space and this can be achieved by using kernel functions. We can exploit the freedom to select any inner product of our choice so that we can also perform KPCA to non-numeric entities. Formally, given a zero-mean column data matrix $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ containing n m-dimensional data points, principal component analysis (PCA) deals with representing the data through principal components maximizing the variance in $\mathbf{X}$ by solving

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}, \tag{1}$$

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

where $\mathbf{C} = \frac{\mathbf{1}}{\mathbf{n}}\mathbf{X}\mathbf{X}^\mathbf{T}$ is the covariance matrix, λ an arbitrary eigenvalue and $\mathbf{v}$ its corresponding eigenvector. Considering (1), we can represent every eigenvector as a linear combination of the data points the data matrix $\mathbf{X}$ as

$$\frac{1}{n\lambda}\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{v} = \mathbf{X}\beta = \mathbf{v}, \tag{2}$$

where $\beta \in R^m$. Substituting (2) in (1) we obtain

$$\frac{1}{n}\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X}\beta = \lambda\mathbf{X}\beta, \tag{3}$$

which upon data projection can be represented as

$$\frac{1}{n}\mathbf{X}^\mathbf{T}\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{X}\beta = \lambda\mathbf{X}^\mathbf{T}\mathbf{X}\beta. \tag{4}$$

Replacing the occurrences of the gram matrix $\mathbf{X}^\mathbf{T}\mathbf{X}$ by a selected kernel matrix $\mathbf{K}$ as

$$\frac{1}{n}\mathbf{K}\mathbf{K}\beta = \gamma\mathbf{K}\beta. \tag{5}$$

and eliminating $\mathbf{K}$ as

$$\frac{1}{n}\mathbf{K}\beta = \gamma\beta, \tag{6}$$

we result in a kernelized representation of the conventional PCA.

For the particular case of words and DNA sequences, the inner product can be any string similarity. For our experiments, we adapt the the Sørensen-Dice coefficient by considering not only bigrams, but n-grams of any length in general. Let $\mathcal{V}$ a vocabulary of words, $\mathcal{G}_n(w)$ the n-grams of a word $w \in \mathcal{V}$. We define our similarity function s of two words $x, y \in V$ as follows:

$$s(x,y) = \sum_{n\in\mathbb{N}^+} \alpha_n \frac{2|\mathcal{G}_n(x) \cap \mathcal{G}_n(y))|}{|\mathcal{G}_n(x) \cup \mathcal{G}_n(y)|}, \quad \sum_n \alpha_n = 1 \tag{7}$$

where α_i determines the weight of the Sørensen-Dice coefficient term for each n-gram length. We compute a similarity matrix $\mathbf{S}$ by applying this similarity function s to all word pairs of our vocabulary V:

$$S_{ij} = s(w_i, w_j) \quad \forall w_i, w_j \in \mathcal{V}. \tag{8}$$

Then, we calculate the kernel matrix $\mathbf{K}$ by applying a non-linear kernel function (for instance RBF kernel or polynomial kernel) to the similarity matrix $\mathbf{S}$. After computing the eigenvectors and eigenvalues of the resulting matrix $\mathbf{K}$, we construct our projection matrix $\mathbf{P}$ by selecting d eigenvectors $\mathbf{v_1}, \ldots, \mathbf{v_d}$ and dividing them by their respective eigenvalues $\lambda_1, \ldots, \lambda_d$:

$$\mathbf{P} = [\frac{\mathbf{v_1}}{\lambda_\mathbf{1}}, \ldots, \frac{\mathbf{v_d}}{\lambda_\mathbf{d}}]. \tag{9}$$

We can now generate a KPCA embedding for any word w_t. We only need to compute the similarity function of the word against all the words processed from

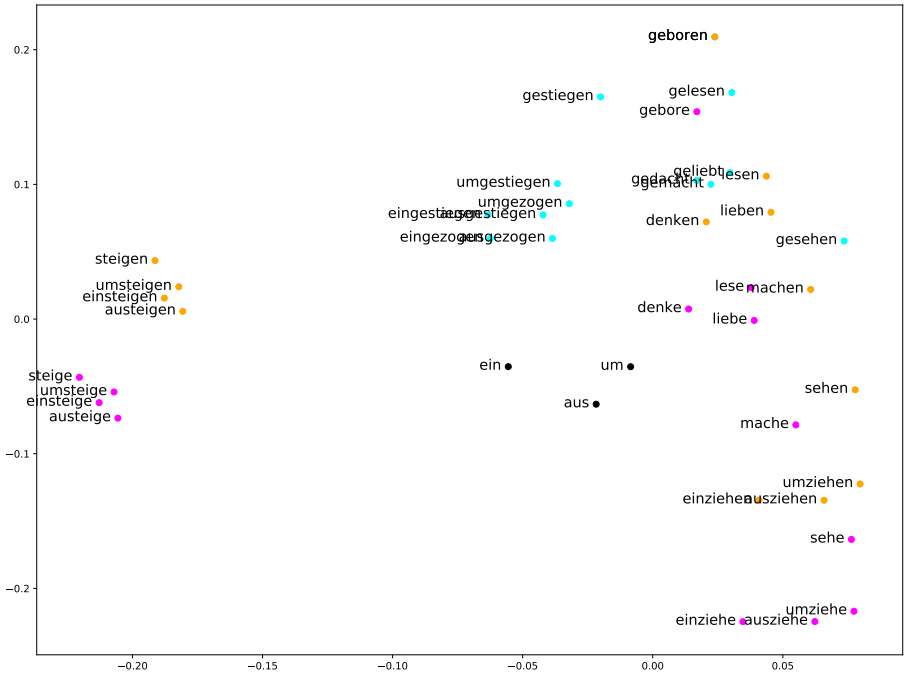

Fig. 1: Visualization of KPCA embeddings generated from a reduced German vocabulary with a RBF kernel, $\sigma = 0.7$ and two principal components. Each color represents a different verb tense. Black dots refer to German prefixes. Best seen in color.

the vocabulary $\mathcal{V}$ and apply the kernel function k to the resulting vector. This results in a kernelized distance vector $\mathbf{r_t}$. The product of $\mathbf{r_t}$ with the projection matrix $\mathbf{P}$ constitutes the d-dimensional KPCA embedding $\mathbf{u_t}$ of the word w_t.

$$\mathbf{r_t} = \mathbf{k}(\mathbf{s}(\mathbf{w_t}, \mathcal{V})) \qquad \text{and} \qquad \mathbf{u_t} = \mathbf{P}^\top \mathbf{r_t}. \tag{10}$$

It is important to note that, this approach can be generalized to encode any other non-numeric entity as long as we can define an equivalent similarity function between each pair of entities.

3 Experiments

We show how our KPCA embeddings can be used for different kinds of sequential data. In particular, we apply our approach to represent words to classify different types of verb forms and to represent DNA sequences to recognize splice junctions.

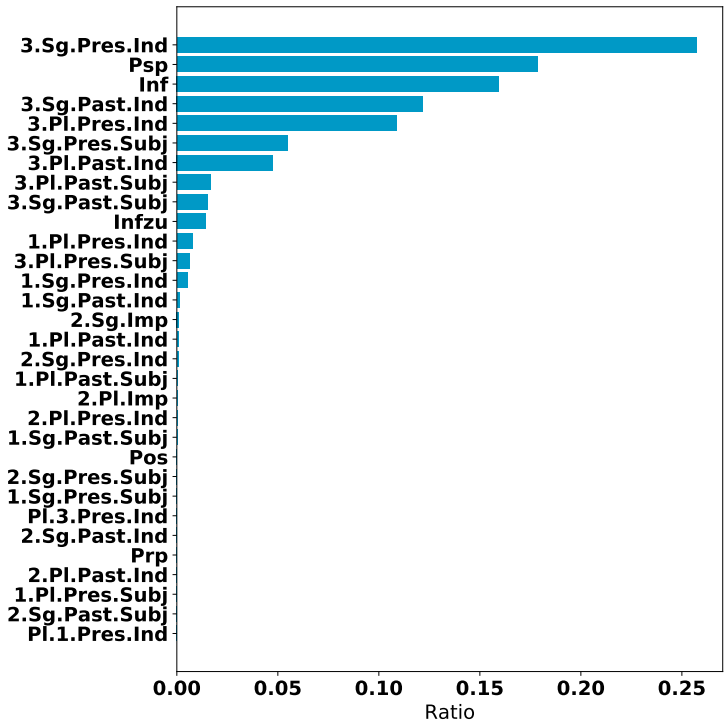

Fig. 2: Distribution of 31 different morphological tags from the TIGER treebank. Note the imbalance especially with respect to first and second person forms.

3.1 Verb classification

We test how our KPCA embeddings can encode morphology with a fine-grain POS tagging task. We restrict our vocabulary to the tokens tagged as verbs from the TIGER treebank [2]. This simplifies the problem to classify the correct morphological tag (consisting of grammatical person, number, tense and mode when they apply) of a German verb only from its KPCA embedding.

First, we extract all tokens tagged as verb (corresponding to the TIGER tags VVFIN, VAFIN, VMFIN, VVIMP, VAIMP, VVINF, VVIZU, VAINF, VMINF, VVPP, VMPP, VAPP) and remove all duplicates. This leads to 13370 unique verbs with 31 different morphological tags, whose distribution is showed in figure 2. Then, we apply the approach described in section 2. We build a training set consisting of 80% of the verbs and a test set with the remaining 20%. We consider only bigrams and trigrams to compute the similarity function for each pair of verbs by selecting five different weight distributions (different values for α_2 and α_3 in equation 7). We also incorporate an additional character at the beginning and at the end of each verb when producing the n-grams. After running KPCA on the training set, we infer vector representations of the verbs from the test

set. By using the KPCA embeddings as a features and the morphological tag as label, we train k-nearest neighbors classifiers to predict the morphological tag of a word from only the KPCA embedding. As baseline representations, we also learn a word2vec [10, 9] model for each different vector size. These word vectors were learned applying the default hyperparameter values.

From Table 1 we can observe that a mean accuracy above 77% can be achieved by classifiers taking only the nearest neighbor ($k = 1$). This can be interpreted as a high accuracy considering the extremely unbalanced label distribution (see figure 2). Among the trained classifiers, we can also find some improvement when the trigram similarity weight (α_3) is at least as high as the bigram similarity (α_2). In addition, any KPCA embedding model beats all word2vec models for this task. For the sake of a fair comparison, the displayed word2vec results from Table 1 correspond to models where their vector size matches with a tested number of principal components d of KPCA embeddings and tested with the same k values for k-nearest neighbors. However, we also tested additional word2vec models with vector sizes up to 100 and up to 100 neighbors. None of these larger models reached a mean accuracy above 27%.

3.2 Splice junction recognition on DNA sequences

We encode DNA sequences from the dataset "Molecular Biology (Splice-junction Gene Sequences) Data Set" from UCI Machine Learning Repository [8][2]. This dataset consists of DNA subsequences represented as 30 characters out of the four nucleobases (A, T, C, G) plus other four characters (D, N, S, R) which mark ambiguity. The sequences may contain a spline junction between the 30 first and the 30 last characters. They are thus labeled with three different categories depending if they contain exon/intron boundary (EI class), intron/exon boundary (IE class) or neither (N). The distribution of the classes is displayed in Table 2.

We compute the similarity function considering all n-grams, giving all terms from (7) the same weight:

$$\alpha_i = \begin{cases} 1/58, & i \in \{2, \cdots, 59\} \\ 0, & i \notin \{2, \cdots, 59\} \end{cases} \tag{11}$$

Using the learned representations, we train k-nearest neighbors classifiers to predict to which of the three classes each DNA sequence belongs to. Analyzing the prediction performance, as we can see from Table 3, we achieve a mean accuracy 94.67% with two different kernels. This result beats all baseline systems that are presented with the dataset, including knowledge-based artificial neural networks (KBANN) [11].

[2] `https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)`

d	α_3	$k=1$	$k=2$	$k=3$	$k=5$	$k=10$
KPCA embeddings						
5	1	76.76	65.19	65.20	63.02	60.56
5	0.75	76.67	66.10	65.94	63.74	60.85
5	0.5	76.56	65.73	65.78	63.33	60.50
5	0.25	76.72	65.33	65.14	63.01	60.27
5	0	**77.01**	65.19	65.14	62.64	59.60
word2vec						
5		12.08	10.85	10.92	13.43	15.56
KPCA embeddings						
10	1	**77.38**	66.93	66.64	64.92	62.59
10	0.75	77.13	66.42	66.26	63.70	61.37
10	0.5	76.59	65.05	64.87	62.19	59.91
10	0.25	76.63	65.45	65.31	62.60	59.65
10	0	77.05	65.81	65.96	63.72	60.68
word2vec						
10		13.91	11.22	12.23	15.41	17.39
KPCA embeddings						
15	1	77.55	66.52	66.96	65.13	63.27
15	0.75	76.97	65.70	65.96	64.02	61.79
15	0.5	77.28	66.34	66.55	64.62	62.22
15	0.25	77.46	67.24	67.45	65.45	63.65
15	0	**77.72**	67.38	67.46	65.43	63.83
word2vec						
15		14.29	13.39	13.99	17.69	20.08
KPCA embeddings						
20	1	77.22	66.67	66.43	64.92	62.75
20	0.75	77.11	65.61	65.84	64.68	61.65
20	0.5	**77.56**	66.59	66.42	65.11	62.52
20	0.25	77.55	66.88	67.03	65.16	63.20
20	0	77.49	66.65	66.96	65.47	63.59
word2vec						
20		13.76	13.24	14.47	18.61	20.91

(a) Polynomial kernel (degree 3)

d	α_3	$k=1$	$k=2$	$k=3$	$k=5$	$k=10$
KPCA embeddings						
5	1	**77.08**	65.29	64.95	63.02	60.48
5	0.75	76.89	66.23	66.03	64.50	61.38
5	0.5	76.75	66.12	66.05	63.78	61.86
5	0.25	76.95	65.66	65.69	63.64	61.02
5	0	76.89	65.95	65.53	63.65	61.26
word2vec						
5		12.08	10.85	10.92	13.43	15.56
KPCA embeddings						
10	1	76.80	66.01	66.38	64.54	61.92
10	0.75	76.72	66.13	66.83	64.30	61.95
10	0.5	77.17	66.45	66.70	64.36	61.96
10	0.25	**77.23**	66.30	66.78	64.40	62.43
10	0	77.13	66.26	66.67	64.42	62.16
word2vec						
10		13.91	11.22	12.23	15.41	17.39
KPCA embeddings						
15	1	77.28	66.45	66.96	64.82	62.60
15	0.75	77.05	66.12	66.66	64.44	61.90
15	0.5	**77.23**	65.94	66.17	64.01	62.00
15	0.25	76.82	65.15	65.51	63.448	61.11
15	0	76.51	65.52	65.34	63.76	60.98
word2vec						
15		14.29	13.39	13.99	17.69	20.08
KPCA embeddings						
20	1	**77.51**	66.16	67.00	65.06	62.67
20	0.75	76.93	65.95	66.00	64.23	62.22
20	0.5	77.05	65.01	65.29	63.76	61.61
20	0.25	76.67	64.85	64.94	63.10	61.22
20	0	76.43	64.88	64.70	62.76	60.94
word2vec						
20		13.76	13.24	14.47	18.61	20.91

(b) RBF kernel ($\sigma = 2.26$)

Table 1: Mean accuracy in % predicting the verb tag with k nearest neighbors, trigram ratio α_3 ($\alpha_2 = 1 - \alpha_3$) and d principal components applying different kernel functions. For the word2vec baselines, d refers to the word vector size.

Class	Nr. sequences	Ratio
EI	767	25%
IE	768	25%
Neither	1655	50%

Table 2: Class distribution of the splice-junction DNA sequences dataset

d	k					
	1	3	6	10	14	19
1	57.21	55.49	58.46	61.91	**62.54**	62.07
2	73.98	76.18	79.31	79.78	**81.03**	80.72
3	92.16	92.63	93.42	93.57	**94.51**	93.89
4	91.22	92.32	93.10	93.26	**93.42**	**93.42**
5	90.44	92.63	93.10	93.26	**93.89**	**93.89**
6	89.66	92.79	93.42	92.95	93.57	**94.04**
7	90.75	92.48	92.95	93.57	93.57	**94.51**
8	90.13	93.57	**93.89**	**93.89**	**93.89**	**93.89**
9	90.13	91.22	92.95	93.73	**94.67**	93.57
10	89.50	91.69	92.16	92.95	**93.42**	**93.42**

(a) Polynomial kernel (degree 2)

d	k					
	1	3	6	10	14	19
1	52.66	56.11	59.09	58.78	**61.44**	60.19
2	70.06	75.24	74.92	**77.12**	76.18	76.49
3	90.60	90.44	90.28	91.38	92.01	**92.32**
4	92.79	93.42	94.04	93.26	93.26	**94.04**
5	92.32	92.79	93.26	94.04	**94.20**	93.89
6	91.85	92.79	93.42	**93.89**	92.95	93.26
7	90.91	93.42	93.89	**94.04**	93.73	93.73
8	90.13	92.16	93.42	**93.57**	92.95	93.42
9	88.87	91.22	92.16	93.89	**94.67**	94.04
10	88.87	91.85	92.63	**93.26**	92.63	91.54

(b) RBF kernel ($\sigma = 0.72$)

Table 3: Mean accuracy in % predicting splice junctions with k nearest neighbors and d principal components applying different kernel functions. Several results beat the baseline system KBANN (93.68% mean accuracy).

4 Discussion and future work

We showed that our KPCA embedding approach to learn vector word representations can encode the morphology of the words in an unsupervised fashion, at least for the particular case of German verbs. The learned KPCA embeddings could beat by far any word2vec model in the task of predicting the grammatical tag. The highest accuracy was achieved by taking the nearest neighbor to predict the verb category. Due to this fact, we suspect that our proposed word representations tend to form clusters according to their word form, from which predicting a grammatical tag is a feasible task with simple classifiers such as k-nearest neighbors classifiers.

Since many NLP applications require also syntactic and semantic information about the words, good word embeddings should also incorporate information not only from the form of the represented word but also about the context in which they appear. In this direction, we will enhance our approach by adapting our similarity function so that it also considers the frequency of each evaluated word pair appearing in the same context or, alternatively, by taking our KPCA representations as input representation of a neural network architecture. For the latter, our KPCA would "just" substitute the one-hot encoding of most of neural

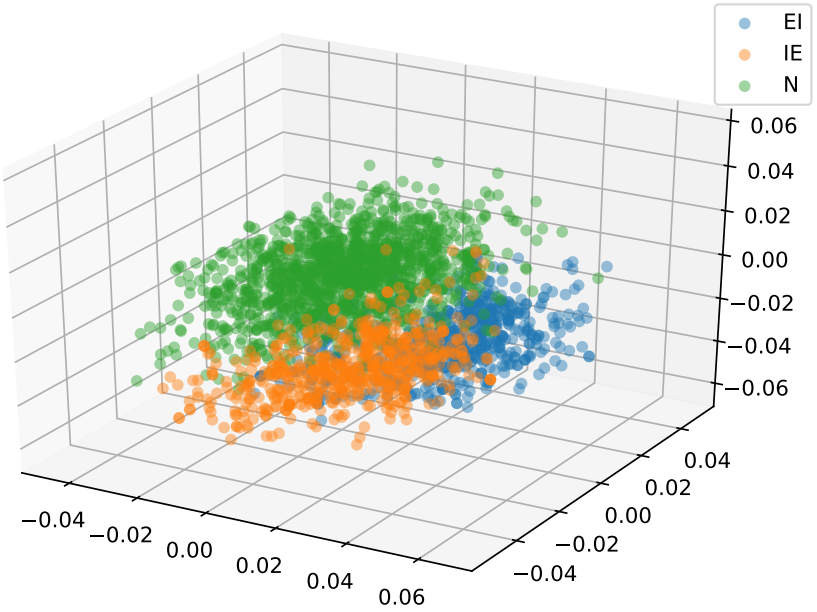

Fig. 3: Visualization of KPCA embeddings generated from the DNA sequences with RBF kernel, $\sigma = 0.72$ and seven principal components. Our dataset consists of DNA subsequences represented as 30 characters out of the four nucleobases (A, T, C, G) plus other four characters (D, N, S, R) which mark ambiguity. The sequences may contain a spline junction between the 30 first and the 30 last characters. They are thus labeled with three different categories depending if they contain exon/intron boundary (EI class), intron/exon boundary (IE class) or neither (N). Only their first three components are plotted. Best seen in color.

language models. Additionally, we will also extend our research evaluating the same approach on other morphologically rich inflected languages (like any of the Romance languages) or agglutinative languages (such as Turkish). We assume they may profit the most from our approach since their word forms reveal more grammatical information than word forms from more analytic languages such as English. To this extent, KPCA embeddings may also help to overcome the lack of linguistic resources of some of these non-English languages.

Furthermore, we presented how we can learn KPCA embeddings for DNA sequences. These representations proved to be useful in the task of predicting splice junctions. It would be also interesting to generalize our method to encode other types of discrete sequential data where also n-grams could be extracted, for instance text paragraphs (word n-grams), protein sequences (amino acid n-grams) or even sheet music (note n-grams).

References

1. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.
2. S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620, 2004.
3. R. Cotterell and H. Schütze. Morphological word-embeddings. In *HLT-NAACL*, pages 1287–1292, 2015.
4. A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.
5. Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.
6. A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
7. G. Jurdzinski et al. Word embeddings for morphologically complex languages. *Schedae Informaticae*, 25:127–138, 2016.
8. M. Lichman. UCI machine learning repository, 2013.
9. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013.
10. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
11. M. O. Noordewier, G. G. Towell, and J. W. Shavlik. Training knowledge-based neural networks to recognize genes in dna sequences. In *Advances in Neural Information Processing Systems*, pages 530–536, 1991.
12. B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.
13. J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In A. for Computational Linguistics, editor, *48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, 2010.