RatVec: A General Approach for Low-dimensional Distributed Vector Representations via Rational Kernels

Eduardo Brito[1,2], Bogdan Georgiev[1,2], Daniel Domingo-Fernández[1,3,4], Charles Tapley Hoyt[1,3,4], and Christian Bauckhage[1,2,4]

[1] Fraunhofer Center for Machine Learning, Germany
`eduardo.alfredo.brito.chacon@iais.fraunhofer.de`
[2] Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
[3] Fraunhofer SCAI, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
[4] B-IT, University of Bonn, Endenicher Allee 19a, 53115 Bonn, Germany

Abstract. We present a general framework, *RatVec*, for learning vector representations of non-numeric entities based on domain-specific similarity functions interpreted as rational kernels. We show competitive performance using k-nearest neighbors in the protein family classification task and in Dutch spelling correction. To promote re-usability and extensibility, we have made our code and pre-trained models available at https://github.com/ratvec.

Keywords: Representation Learning· Kernel Principal Component Analysis· Bioinformatics· Natural Language Processing

1 Introduction

The success of distributed vector representations in natural language processing (NLP) tasks has motivated their use for other areas such as biological sequence analysis [1]. Most of them rely on the *distributional hypothesis* [8]: "words that appear in similar context are similar". We aim to relax this constraint with a general framework to learn vector representations of non-numeric entities including text, DNA sequences, and protein sequences based on similarity functions that can be expressed as rational kernels [5]. Supported by the robust theoretical framework behind rational kernels, we obtain vector representations via kernel PCA (KPCA) [11] that, together with a simple k nearest neighbors (kNN) classifier, constitute an efficient and explainable classification pipeline showing competitive performance in two different tasks: protein family classification and Dutch spelling correction.

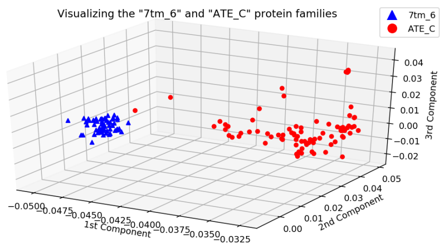

Fig. 1. Visualization of learned vectors for two different protein families using bigram similarity and a representative vocabulary of 1,000 sequences. A clear separation can be observed by only using the first three components of the vectors. Best seen in color.

2 Related Work

Our approach extends our previous work on *KPCA embeddings*, where we studied vector representations for words and DNA sequences via KPCA replacing the dot product by a specific similarity function [3]. There are also a myriad of previous kernel methods for NLP tasks such as text classification [6,10] and named entity recognition [7]. However, they are either restricted to a specific kernel function (i.e. to a concept of similarity) or lack in explicit vector representations.

3 Approach

Let X be a dataset consisting of n elements, to which we will further refer as the *full vocabulary*; and k a rational kernel (a particular similarity function). Computing a kernel matrix $\mathbf{K}$ from X may be computationally prohibitive for large datasets due to its time and space complexity ($O(n^2)$). Hence, we select only the m elements considered "most representative" (in a domain-specific formulation) from X to construct our *representative vocabulary* V and compute a kernel matrix $\mathbf{K_V}$ from all element pairs from V. After centering and diagonalizing $\mathbf{K_V}$, we construct our projection matrix P_V. This is required to generate m-dimensional representations for the full vocabulary. In a last step, we assign the first $d \leq m$ components of each computed vector to each full vocabulary element. There are the resulting d-dimensional vector representations obtained with our approach. When k is suitable for the task, d being of a lower order of magnitude than m can lead to optimal results, as we will see in Section 4.1. The choice of m can be adjusted to fit the computational resources of the user.

The produced representations tend to cluster naturally according to the domain-specific concept of similarity applied by the selected rational kernel, as we can see in Fig. 1. This has two main advantages:

1. A simple kNN classifier (eventually 1NN) can suffice for classification tasks.
2. Learning the vector representations and kNN constitute an *explainable* classification pipeline: an entity is assigned to a particular class because it is *similar* to the labeled entities used to train the classifier.

Table 1. Accuracy of RatVec in different subdatasets compared to BioVec [1]

Dataset	BioVec (baseline)	RatVec (bigram sim.)	RatVec (trigram sim.)
Top 1,000 families	0.94	0.94	0.91
Top 2,000 families	0.93	0.93	0.91
Top 3,000 families	0.92	0.93	0.91
Top 4,000 families	0.91	0.93	0.91
All families	0.93	0.93	0.91

Table 2. Accuracy of BioVec representations with a SVM [1] and a kNN classifier

Top 1000 families		Top 2000 families		Top 3000 families		Top 4000 families		All families	
SVM	1NN	SVM	1NN	SVM	1NN	SVM	1NN	SVM	1NN
0.94	0.94	0.93	0.94	0.92	0.93	0.91	0.93	0.93	0.93

4 Experimental Results

4.1 Protein Family Classification

In this task, we classify protein sequences to their corresponding families with the same Swiss-Prot dataset as in [1], containing 7,027 protein families and 324,018 protein sequences[5]. The system is evaluated for each protein family by 10-fold cross-validation on a balanced dataset consisting of all amino acid sequences of the family and as many randomly sampled sequences from all other families.

We produced protein representations with 25 dimensions applying our approach with bigram and trigram similarity and trained a nearest neighbor classifier (1NN). We generated our representative vocabulary by taking the shortest sequence of the 1,000 most frequent protein families. We evaluated this pipeline in the same setting as [1]. We report the weighted average accuracy results in Table 1. Also, the results of the evaluation limited to subsets of the full dataset are presented (e.g. sequences belonging to the 1,000 most frequent protein families).

The results from Table 1 show that our approach with the bigram similarity reaches the same accuracy as the baseline and or even outperforms it when we restrict the dataset to the first 3,000 or 4,000 most frequent protein families. Considering that the reported BioVec representations are four times longer than ours, our approach to encode proteins seems to be more efficient.

In contrast to [1], we do not train any support-vector machine (SVM) but a 1NN classifier. To assess the impact of the different classification algorithm, we evaluated a pretrained BioVec model [6] with our setup. The results (Table 2), showing that 1NN is more suitable than SVMs for this task, are in line with our previous experiments on classification tasks using distributed vector representations, where simple algorithms such as kNN and logistic regression outperform more complex ones such as random forests or neural networks [3, 4].

[5] http://dx.doi.org/10.7910/DVN/JMFHTN
[6] https://github.com/kyu999/biovec

Table 3. Results of our system compared to Valkuil (CLIN28 shared task baseline) in terms of successful corrections (TP), wrong corrections (FN), and accuracy [2][a].

	Capitaliz. (165 errors)			Non-word (62 errors)			Red. punct (53 errors)			Mis. punct (18 errors)			Archaic (5 errors)			Total (303 errors)		
System	TP	FN	Acc.	TP	FN	Acc.	TP	FN	Acc.	TP	FN	Acc.	TP	FN	Acc.	TP	FN	Acc.
Valkuil	0	0	-	21	0	1.0	0	3	0.0	1	0	1.0	0	0	-	22	3	0.88
RatVec	105	12	0.90	24	15	0.62	8	5	0.62	3	3	0.50	1	0	1.0	141	35	0.80

[a] https://github.com/LanguageMachines/CLIN28_ST_spelling_correction

4.2 Dutch Spelling Correction

We also evaluated our approach by participating in the CLIN28 shared task, where our system obtained the best F1 score among the competing teams [2]. The task focused on correcting errors in extracts from Dutch Wikipedia pages. We used an older implementation of our approach that we keep for reproducibility[7].

Spell checkers involve two steps: misspelling detection and correction. The latter generally requires ranking a set of correction candidates. Generating them implies finding all valid words differing from the detected misspelling less than a determined edit distance, which is mostly set to 1 for real-world applications to limit computation time [12]. We avoid this strong constraint with our approach.

In our RatVec framework, our full vocabulary is *wdutch*, a word list from the OpenTaal project, from which the 3000 most frequent words form our representative vocabulary. The applied rational kernel is the composition of the bigram similarity [9] with the homogeneous polynomial kernel of degree 2. We generated a vector representation for each full vocabulary word with 2000 dimensions.

Once a misspelling is detected, we compute its RatVec representation and search for the closest precomputed vector. Its related word (its nearest neighbor) is our correction to the misspelling. Formally, this is equivalent to training a 1NN classifier where each valid word is assigned a different label.

Our RatVec framework is only relevant during the correction phase for spelling mistakes that are related to the word form. Hence, we restrict our analysis to the correction results on the five error categories where the word form is relevant, namely those displayed in Table 3. Although the baseline system Valkuil achieves a better average accuracy than our approach for the analyzed misspellings, RatVec outperforms Valkuil in some categories where it completely fails (redundant punctuation errors) or where it cannot be even evaluated because it failed to detect any error (capitalization and archaic spelling errors). From these results, we interpret that our word vectors encode word forms in a suitable way so that similar words can be retrieved.

5 Conclusion and Future Work

We showed that our approach involving rational kernels on KPCA for vector space embeddings provides rich representations for two different kinds of en-

[7] https://github.com/fraunhofer-iais/kpca_embeddings

tities: proteins and words. In the first application, we presented how to learn protein vectors from amino acid sequences and how to apply them to predict their protein family. In the second, we learned word representations that encode word form information so that they can be applied to correct misspellings once these are detected. In both tasks, our results are comparable to state-of-the-art approaches. Thanks to the simplicity of kNN classifiers and the interpretable similarity concept we apply to generate the vector representations, our approach may be advantageous for some real-world applications compared to more complex machine learning models such as deep neural networks.

In future work, we will explore possibilities of learning optimal similarity metrics (modeled as transducers) that, incorporated in our presented approach, solve a particular task.

References

1. Asgari, E., Mofrad, M.R.: Continuous distributed representation of biological sequences for deep proteomics and genomics. PloS one **10**(11), e0141287 (2015)
2. Beeksma, M., van Gompel, M., Kunneman, F., Onrust, L., Regnerus, B., Vinke, D., Brito, E., Bauckhage, C., Sifa, R.: Detecting and correcting spelling errors in high-quality dutch wikipedia text. Computational Linguistics in the Netherlands Journal **8**, 122–137 (2018)
3. Brito, E., Sifa, R., Bauckhage, C.: KPCA embeddings: an unsupervised approach to learn vector representations of finite domain sequences. In: LWDA. pp. 87–96 (2017)
4. Brito, E., Sifa, R., Cvejoski, K., Ojeda, C., Bauckhage, C.: Towards German word embeddings: A use case with predictive sentiment analysis. In: Data Science–Analytics and Applications, pp. 59–62. Springer (2017)
5. Cortes, C., Haffner, P., Mohri, M.: Rational kernels: Theory and algorithms. Journal of Machine Learning Research **5**, 1035–1062 (Dec 2004)
6. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. Journal of Intelligent Information Systems **18**(2-3), 127–152 (2002)
7. Giuliano, C.: Fine-grained classification of named entities exploiting latent semantic kernels. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning. pp. 201–209. Association for Computational Linguistics (2009)
8. Harris, Z.S.: Distributional structure. Word **10**(2-3), 146–162 (1954)
9. Kondrak, G.: N-gram similarity and distance. In: Int. Symp. on String Processing and Information Retrieval. pp. 115–126. Springer (2005)
10. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research **2**(Feb), 419–444 (2002)
11. Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: International conference on artificial neural networks. pp. 583–588. Springer (1997)
12. Tijhuis, L.: Context-based spelling correction for the Dutch language: Applied on spelling errors extracted from the Dutch wikipedia revision history (2014)