Natural Language Processing for tasks with specialized domain language – Project Report

data.camp097@audi.de

October 22, 2019

Contents

1 Definition

1.1 Overview

Natural Language Processing as a sub-discipline of machine learning has had major success in recent years on a broad range of natural language related tasks. There exist many not yet automated and potentially very useful NLP tasks in the automotive industry, ranging from automated scanners for the monitoring of (social) media with respect to potential quality or safety problems (a legal requirement for manufacturers), automated analysis of warranty and repair documents, systems for customer support, monitoring and analysis of patent applications to identify trends and many more. However, when confronted with highly specialized sub-domain languages like automotive engineering, commercial off the shelf (COTS) products decline in performance, often up to a point of being useless [2].

1.2 Problem Statement

One key element of many NLP pipelines are word embeddings, dense vecor representations of the vocabulary. Examples of these methods are word2vec [3], GloVe [4] or FastText [5]. These word embeddings are able to represent (surprisingly) much semantic and syntactic meaning, which makes downstream tasks in general much easier. However, to learn this, the embeddings have to be trained extensively, state of the art is basically a coyy of large chunks from the internet (a complete Wikipedia dump beeing at the lowest end). When it comes to sub-domain language, we face a problem: This data is in general not easily available, often proprietary or copyrighted and never as abundant as general text. The basic hypothesis for this capstone project is the assumption that the dominant problem of NLP algorithms operating on specialized domain languages is the fact that during training the algorithm hasn't seen any or enough domain specific text. For this reason I trained two different vector representations, the classic word2vec and the more recent FastText on a self-aquired domain specific data corpus and compared it with different metrics to a generic model, the Facebook German Fasttext Model.

1.3 Metrics

The evaluation of word embeddings is no easy task. To the best of my knowledge, there does not exist a single commonly used or agreed upon method on how to evaluate these models [9–11]. Instead a range of methods is used. These methods fall in two categories, extrinsic and intrinsic. Intrinsic evaluation takes into consideration only the embedding itself whereas extrinsic methods measure the embedding performance indirectly by a downstream NLP task with its own metric. For this project, I implemented the following metrics to compare the different embedings:

- M_1: Extrinsic measure: A real world multiclass classification task of software change requests (CRQ) for embedded control software. The classification separates the CRQs by topic to forward it directly to the right experts and change control boards.[1]

  For the classification task, I built an LSTM with the word embedding as a first (untrainable layer). The architecture itself is pretty basic using an attention layer. [2] The motivation for the attention layer is the fact, that the engineers are encouraged to use a certain structure to describe the CRQ. Therefore it might be possible, to focus the attention to certain areas of the CRQ for classification.

  The model itself can be found in the notebook "LSTM.ipynb".

[1] The CRQs themselfes are confidential and cannot be published or submitted.

[2] Adopted from `https://github.com/philipperemy/keras-attention-mechanism`

- M_2: Clustering by subsystem. A vehicle is built from subsystems like engine, transmission, exhaust system, etc.. For 5 subsystems we will pick representative 10 representative words that will represent this subsystem.

  The basic idea of this metric is then to leverage clustering or classification in the vector space of the embedding itself. As the metric for word embeddings is the consine distance,

  $$d(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| \cdot |\mathbf{b}|} \tag{1}$$

  the clustering or classifcation algorithms will have to use this as a measure of distance.

  To generate a metric out of this idea, I chose KNN classification with cosine distance and use the train(!) error as the metric. This is motivated by the intuition, that KNN classification only works if the subsystems are tightly grouped and have distance to other clusters.

- M_3: Word analogy. Another intrinsic metric for word embddings is to assess semantic meaning. We look for relations of the form

  $$a : b :: c : d$$

  In particular, we will assess the semantic representation of technical concepts. Two examples are

  $$\text{engine} : \text{engine control unit} :: \text{transmission} : \text{transmission control unit}$$

  $$\text{camshaft} : \text{camshaft bearing} :: \text{crankshaft} : \text{crankshaft bearing}$$

  We can extract the word d from words a,b and c from the embedding simply with

  $$d = \underset{i}{\operatorname{argmax}} \frac{(x_b - x_a + x_c)^T \cdot x_i}{|x_b - x_a + x_c|} \tag{2}$$

  For this to become a metric, I simply count the number of word analogies the embeddings get right in their Top-n list.

- M_4**: Word similarity**. A well known metric is the correlation with human similarity scores (like wordsim353...) pairs of words from technical jargon and ask colleagues to rate their similarity. This measure of similarity will then be compared to the cosine distance the embeddings produce for these word pairs. This metric can also be adopted to include relatedness instead of similarity.

- M_5: As a qualitative measure I visualized the embeddings using t-SNE [12] to get a comparative qualitative measure.

2 Analysis

2.1 The Dataset

For a domain specific data corpus I collected scraped the text from the following sources:

- technical / specialist books: 60 titles, 28507 pages, 5950125 words in total
- internal technical documentation: 17763 change requests for embedded control software with a total of 960921 words[3]
- patents: 500 patents from the domain vehicle powertrain, containing a total of 9871 pages with a total of 3929993 words.

This data corpus would in principle b easy to scale up. As a company, we have access to electronic versions of books from almost all major publishers in the field. Furthermore we have electronic access to the major journals in the field. The patent database is public and with a paid account we can access an unlimited number of documents. The internally available data is of course limited, however only a fraction of the available data was used in this project.

2.2 Algorithms and Techniques

2.3 Benchmark Models

As a benchmark I chose the pretrained German Facebook FastText embedding. [4]

As a benchmark, I propose taking an existing trained word embedding. For all the mentioned word embeddings (Word2vec, GloVe, FastText) there exist extensively pretrained models for many languages in NLP Toolkits like spacy [6], gensim [7] or NLTK [8] or GitHub (cf. Section 5.3). For details on how to get from a word embedding to an evaluation metric, please refer to section 1.3 Evaluation Metrics.

2.4 Related Work

I found one paper with a similar goals [2] from the Oil and Gas Industry. The paper describes a similar problem and focusses on the embeddings as well. As a novel element they define word similarity by analysis of knowledge database instead of relying on human judgement.

3 Methodology

As a general remark, NLP Toolkits like spacy [6], gensim [7] or NLTK [8]

3.1 Data Preprocessing

For the preprocessing, the following steps were taken:

- Scraping war text from pdf. This is done with the Python packages "textract" and "PyPDF2". Used on the patents and the books, the CRQs were scraped directly to raw text via SQL from a database.
- The structure of the data is a list of lists, the inner lists being the scope over which the windows run while training.

[3] proprietary, cannot be submitted

[4] `https://fasttext.cc/docs/en/crawl-vectors.html`

- The raw text is serialzied to disk using pickle.
- The raw text is stripped of punctuation, numerics, multiple whitespace, words shorter than 3 letters, and German stop words. It is lower-cased and split. A lemmatizer was integrated but not used due to runtime considerations. The lsit of German stop words was imported from the python package "get_stop_words", the preporcessing was mainly done with the package "gensim" preprocessing functions.
- The preprocessed text is serialized to disk using pickle,

The scraping and preprocessing is implemented in the notebook "Data Collection.ipynb".

3.2 Implementation

Training the embeddings was done using the package "gensim".

The training of the embeddings is implemented in the notebook "Embeddings.ipynb"

3.3 Optimization

Optimizing the embeddings To optimize the embeddings, a few options are available.

- Increasing the embedding size. I generously started with an emedding size of 300, which is often used. It might be worthwhile to experiment with this parameter, especially to see if reducing the size does lower the performance in a measurable way.
- More training data: As the training data is unlabelled, it is feasible to massively increase the amount of domain specific training data. I would assume that at least a factor of 100 would be possible.
- Using different embedding algorithms. So far, I used only 2 widely used algorithms, word2vec and FastText, but there are many more established algorithms with will documented strengths and weaknesses. It might be worthwhile to see if improvements could be made by choosing more wisely.
- Retraining a pretrained network. For some embedding algorithms it is possible to retrain them on additional data. It would be interesting to measure the performance of a retrained embedding and compare the results to the exclusively trained embeddings used in this project. In theory it might be possible to somehow bring the good general performance and the domain specific training together.

Optimizing the classifier Even though the classifier is mainly used as a metric to evaluate the different embeddings, some very basic optimization was done before using it. The overall goal is to prevent overfitting and increase test accuracy.

The networt currently still seems to overfit. This is not surprising considering the very small number of available labelled samples and the relatively large capacity of the network.

To prevent overfitting, the following steps were taken:

- Early termination was implemented in the learning.
- Dropout was applied extensively through the network (between layers and recurrently inside the LSTM layer). Dropout rate used is 0.4 everywhere.
- Model Capacity was reduced by decreasing the size of the LSTM internal state to 80)

Other options that were not thoroughly explored in this project include

- Regularization. Not yet included.
- Data augmentation. Cropping might be an option, but was not explored.
- Acquisistion of more data. Always an option :-)

4 Results

4.1 Visualizations

To get a qualitative impression of what the embeddings learned to represent I produced t-SNE plots for the 20 nearest neighbours of two different word pairs: ('Motor', 'Getriebe') (meaning engine and transmission, respectively) and ('Benzin', 'Diesel') (meaning gasoline and diesel, respectively).

From an engineers point of view these groups of words should be clearly separated as they refer to different subsystems or different classes of internal combustion engines.

The following plots mainly show the degree of separation the embeddings assigned to these groups of words. The specifically trained FastText clearly stands out here as it seems to be able to distinguish between these clusters of words.

4.2 Quantitative Results

The metrics M_1–M_4 produce qunatitavie results that are summarized in the following table

	M_1	M_2	M_3	M_4
Word Embedding	Test Acc [%]	Train Acc [%]	Top-10	Pearson Corr
Generic German FastText	80.0	67.4	0	-0.28
domain specific word2vec	83,6	88.4	1-Top1,2	0.42
domain specific FastText	84.5	90.7	1-Top5	0.41

M_1 Downstream Classification Before we discuss the results of the remaining measures, a caveat is in order: The process to aqcuire and score the list of word pairs would not hold up to any scientific standard whatsoever, so every interpretation of the results is speculative. Plus the number of word pairs used is tiny, from the originally planned 60 only 16 were used, as the others are not in all embeddings (especially the generic FastText).

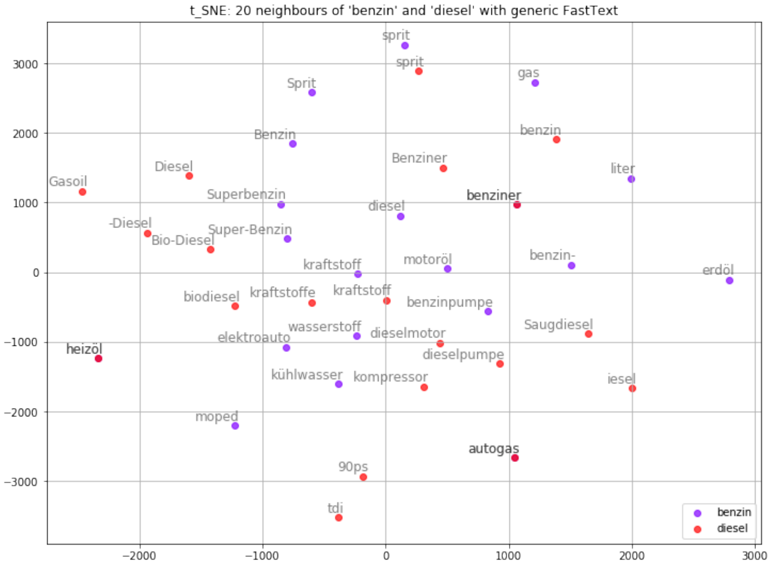

Figure 1: t-SNE for generic FastText on word pair ('Benzin', 'Diesel')

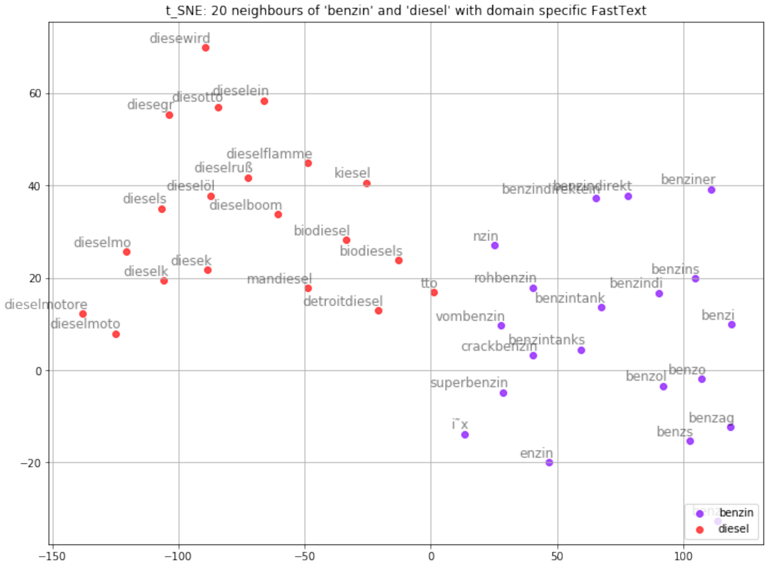

Figure 2: t-SNE for domain specific FastText on word pair ('Benzin', 'Diesel')

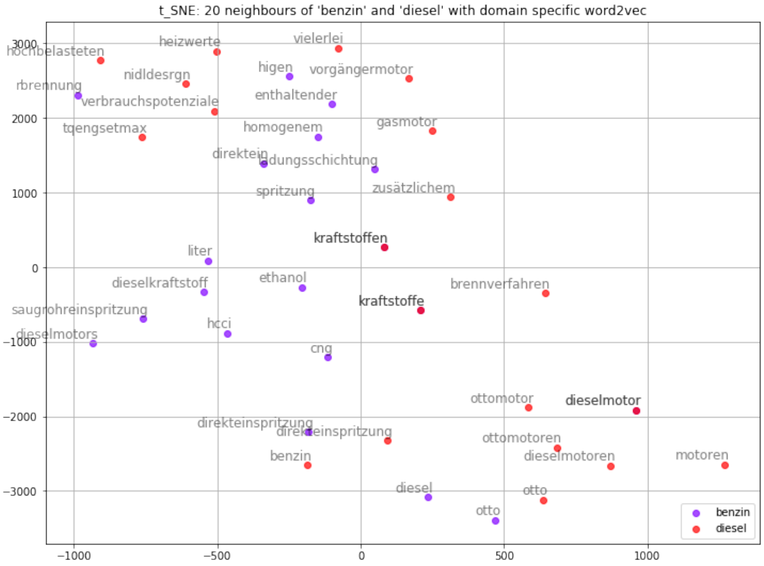

Figure 3: t-SNE for domain specific wor2vec on word pair ('Benzin', 'Diesel')

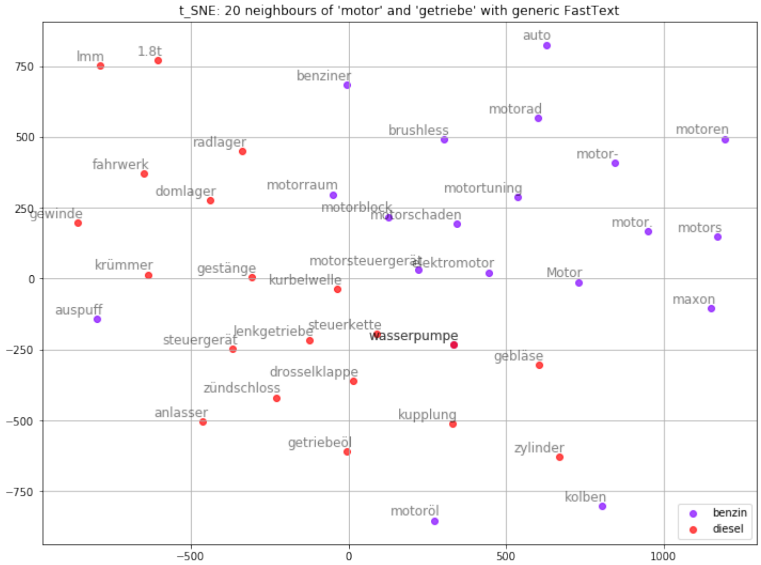

Figure 4: t-SNE for generic FastText on word pair ('Motor', 'Getriebe')

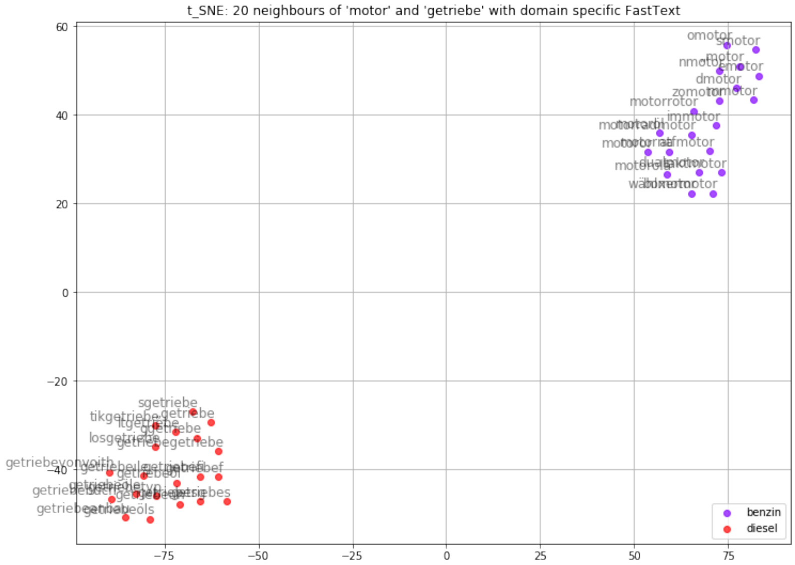

Figure 5: t-SNE for domain specific FastText on word pair ('Motor', 'Getriebe')

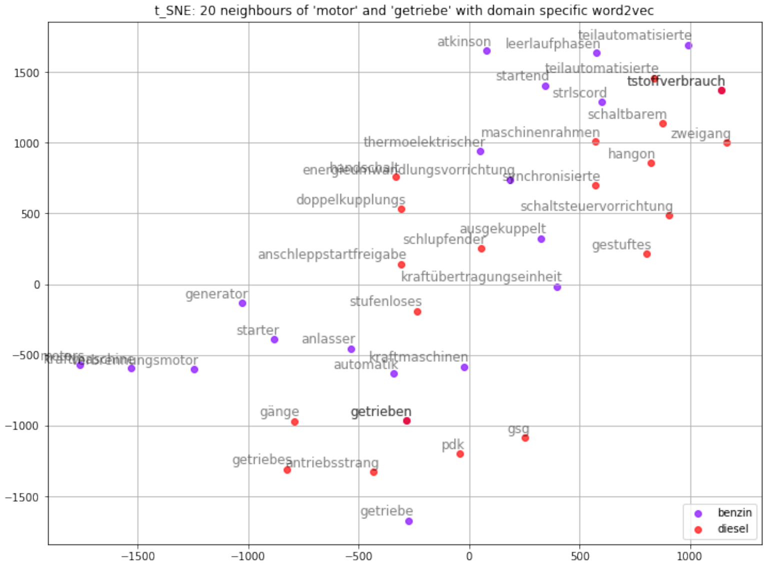

Figure 6: t-SNE for domain specific wor2vec on word pair ('Motor', 'Getriebe')

***M_2* Subsystem Classification in Embedding Space** The results suggest that the embeddings trained on domain specific data can separate the subsystems of the vehicle quite well, whereas the generic model performs significantly worse in this regard. FastText might have an edge over word2vec due to it's property of using character n-grams as the smallest entity, whereas word2vec only hat complete words as entity. This might help to correctly classify compund nouns.

***M_3* Word Analogy** 6 analogies with automotive specific meaning were randomly picked. A point is given for every correctly found analogy in the top-10. Both specifically trained embeddings get one analogy right, whereas the generic embedding does not produce any analogy. The word2vec produces the analogy with Top-1 and Top-2 whereas the fasttext embedding gets it with Top-5 only.

***M_4* Word Similarity** Before we discuss the results of this measure, a caveat is in order: The process to aqcuire and score the list of word pairs would not hold up to any scientific standard whatsoever, so every interpretation of the results is speculative. Plus the number of word pairs used is tiny, from the originally planned 60 only 16 were used, as the others are not in all embeddings (especially the generic FastText). The similarity measure used is a Pearson.coefficient between the human similarity score and the cosine similarity of the embedding. The results would suggest, that only the domain specific embeddings capture the similarities, whereas the generic model does not represent any domain specific semantics. This result is to be expected, but would have to be shown in a more rigorous manner.

5 Conclusion

By all metrics employed, the generic Facebook FastText model performed worse than the domain specific models. This is remarkable, because the data corpus used to train the generic embedding is of the size 6-16 billion words, whereas we trained the domain specific embeddings on a corpus of a little less than 11 million words, three orders of magnitude smaller.

5.1 Reflection

5.2 Improvement/Future Work

Here I will mainly mention the program that I laid out in the proposal and did not yet finish. Plus I draw conclusions from the findings. Overview over the word embeddings used:

Word Embedding	pretrained	trained	retrained
word2vec	GenSim Standard	w/ GenSim Implementation	done w/ GenSim implementation
GloVe	GenSim Standard	w/ GenSim implementation	n.a., needs global cooccurence matrix
FastText	from Facebook-Page,	w/ Facebook implementation	done w/ undocumented feature in Facebook implementation

5.3 Additional Research Questions

Due to time contraints, I didn't touch the additional research questions I raised in the proposal. I still think it would be interesting to follow up on them.

Extra Dimesions. If we retrain a word embedding for a specialized subdomain, do we need to add a few dimensions to the vector space "to make room" for additional semantic concepts? Intuitition would suggest that all existing dimensions are already somehow occupied and new technical concepts would require "new space". So an additional question would be: Do extra dimesions improce the embedding quality? And if so, what semantic meaning can we assign the new dimensions?

Compund Nouns. A distinctive feature of the German language are compound nouns. This holds especially true for technical terms like Nockenwellenlagerungskonzept (cam shaft bearing method). I expect this to be of particular importance to be reflected in the embedding.

Web Resources

- `https://github.com/kudkudak/word-embeddings-benchmarks/blob/master/web/datasets/similarity.py`
- `https://github.com/Hironsan/awesome-embedding-models`
- `https://github.com/philipperemy/keras-attention-mechanism`

References

[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," 2017.

[2] F. Nooralahzadeh, L. Ovrelid, and J. T. Lonning, "Evaluation of Domain-specific Word Embeddings using Knowledge Resources," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (N. C. C. chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, eds.), (Miyazaki,

Japan), European Language Resources Association (ELRA), May 7-12, 2018 2018.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013.

[4] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[6] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.” To appear, 2017.

[7] R. Řehůřek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.

[8] E. Loper and S. Bird, “Nltk: The natural language toolkit,” in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP ’02, (Stroudsburg, PA, USA), pp. 63–70, Association for Computational Linguistics, 2002.

[9] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. J. Kuo, “Evaluating word embedding models: Methods and experimental results,” 2019.

[10] T. Schnabel, I. Labutov, D. M. Mimno, and T. Joachims, “Evaluation methods for unsupervised word embeddings.,” in *EMNLP* (L. MÃ rquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, eds.), pp. 298–307, The Association for Computational Linguistics, 2015.

[11] A. Bakarov, “A survey of word embeddings evaluation methods,” 2018.

[12] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.