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

data.camp097@audi.de

October 22, 2019

1 Definition

1.1 Overview

or Introduction or Context

1.2 Problem Statement

1.3 Metrics

2 Analysis

2.1 The Dataset

The dataset chosen for this project consists of three different 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[1]
- patents: 500 patents from the domain vehicle powertrain, containing a total of 9871 pages with a total of 3929993 words.

2.2 Algorithms

2.3 Benchmark Models

As a benchmark I chose the pretrained German Facebook FastText embedding. [2]

2.4 Related Work

Overview over the word embeddings used:

[1] proprietary, cannot be submitted
[2] `www.spiegel.de`

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 FacebookPage,	w/ Facebook implementation	done w/ undocumented feature in Facebook implementation

3 Methodology

3.1 Data Preprocessing

For

3.2 Implementation

Training the embeddings

3.3 Optimization

One problem seems to be overfitting to the small number of currently available labelled samples. The learning curves suggest this.

To prevent overfitting, the following steps were taken:

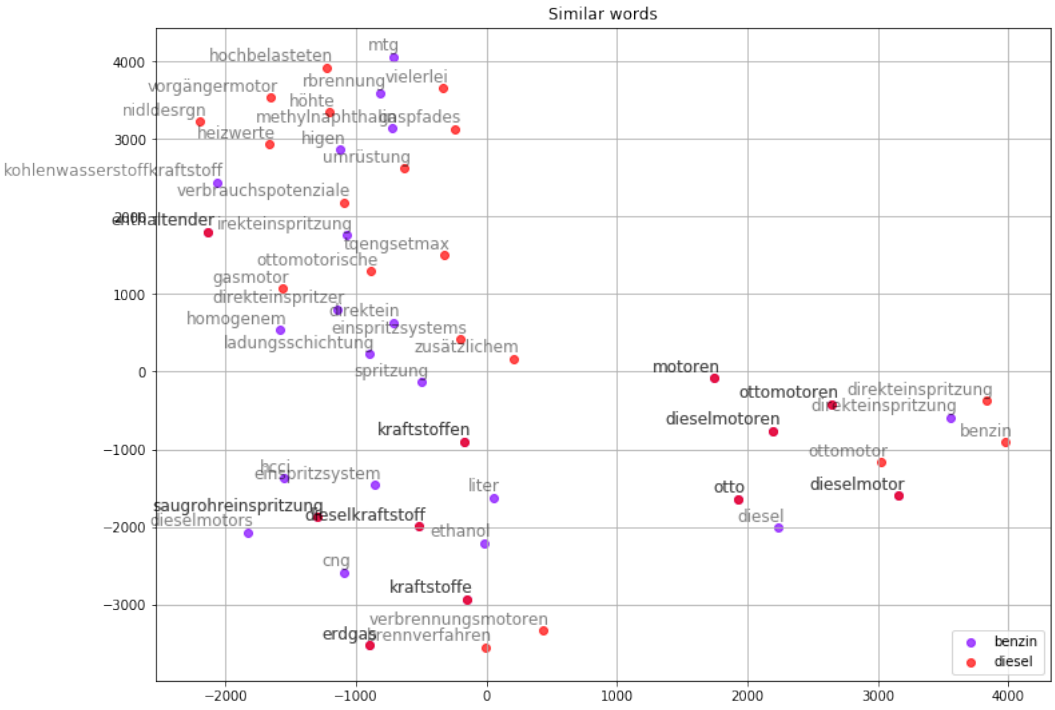
- Early termination.
- Dropout.
- Model Capacity.

Other options that were not thoroughly explored in this project include

- Regularization.
- Data augmentation
- Acquisistion of more data.

4 Results

4.1 Visualizations



Conic Section

My proposed evaluation is best summarized in an empty results table:

Word Embedding	M_1	M_2	M_3	M_4
word2vec pretrained	•	•	•	•
word2vec retrained	•	•	•	•
word2vec trained from scratch	•	•	•	•
GloVe pretrained	•	•	•	•
GloVe retrained	•	•	•	•
GloVe trained from scratch	•	•	•	•
FastText pretrained	•	•	•	•
FastText retrained	•	•	•	•
FastText trained from scratch	•	•	•	•

5 Conclusion

5.1 Reflection

5.2 Improvement/Future Work

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. An overview of recent trends is this field can for example be found in the paper [1] "Recent Trends in Deep Learning Based Natural Language Processing" by Young et al.. However, when confronted with highly specialized sub-domain languages, commercial off the shelf (COTS) products decline in performance, often up to a point of being useless [2]. As an example from personal experience, I would like to mention the use of Google Translate, a benchmark NLP product, on translating text from climbing guidebooks. The results are funny, but utterly useless.

Another example more relevant for this work is the author's area of expertise, automotive engineering. There exist many potentially 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. As in the example of climbing guidebooks, the results of COTS products when confronted with this form of specialized sub-domain language are usually poor. The topic of the capstone project is to make the first steps to remedy this situation. In particular, we will try to improve the word embeddings for our subdomain of interest.

6 Problem Statement

The basic hypothesis for this capstone project is the assumption that the dominant problem of NLP algorithms operating in highly specialized domain languages is the fact that during training the algorithm hasn't seen any or enough domain specific text. One important element of most modern NLP pipelines is a dense vector representation, also referred to as word embedding, instead of sparse representations like one-hot-encoding or bag-of-words. Examples of these methods are word2vec [3], GloVe [4] or FastText [5]. I assume that in particular these representations are not trained well enough on domain-specific language.

7 Datasets and Inputs

The dataset consists of 3 different types of technical documents in German:

- 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 will have to be preprocessed. Preprocessing includes cleaning the documents from things like stopwords, interpunctuation and capitalization. After this a tokenizer and lemmatizer will be used. The preprocessed text will then be used for training of the word embedding.

[3] proprietary, cannot be submitted

8 Solution Statement

In a word: Train word embeddings on text from the subdomain of interest. First I propose to review the different existing word embeddings like Word2vec, GloVe or FastText and pick one to work with. An important part of the research will be the question of how to make sure the embedding performs well on both text in general **and** the domain language. To achive this, I aim at using some sort of pretraining. How this is done best to balance the performance is probably one of the key research questions. The implementation and training itself should be pretty straightforward as there are packages for many word embeddings available that I would use for this project.

9 Benchmark Model

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 13). For details on how to get from a word embedding to an evaluation metric, please refer to section 10 Evaluation Metrics.

10 Evaluation Metrics

To the best of my knowledge, there does not exist a single commonly used or agreed upon method on how to evaluate 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. In this project I will use the following metrics:

- M_1: Extrinsic measure: A multiclass classification task of change requests for embedded control software will be used a an extrinsic metric. In particular, the Top 1 and Top 2 accuracy will be used to compare the word embeddings performance.[4]
- M_2: Clustering by subsystem. A vehicle is built from subsystems like engine, transmission, chassis, interior, etc. For a few different subsystems, we will each pick a list of words that belong to the subsystem. On this set of words, we can then perform clustering or classification. For word embeddings, we use the consine distance

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

  With this distance measure, we can do a KNN classification and will take the resulting classification accuracy as an intrinsic measure for the word embedding.

[4]The classification task itself relies on proprietary text that cannot be published or submitted.

- 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

  engine : engine control unit :: transmission : transmission control unit

  camshaft : camshaft bearing :: crankshaft : crankshaft bearing

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

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

- M_4: Word similarity. I will compile a list of 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.

- As a qualitative measure I intend to visualize the embedding using t-SNE [12] to get a comparative qualitative measure.

My proposed evaluation is best summarized in an empty results table:

Word Embedding	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$
word2vec pretrained	•	•	•	•
word2vec retrained	•	•	•	•
word2vec trained from scratch	•	•	•	•
GloVe pretrained	•	•	•	•
GloVe retrained	•	•	•	•
GloVe trained from scratch	•	•	•	•
FastText pretrained	•	•	•	•
FastText retrained	•	•	•	•
FastText trained from scratch	•	•	•	•

where M_i are the metrics as described above.

11 Additional Research Questions

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.

12 Project Design

The project design seems pretty straightforward. The steps themselves and the whole process will most likely be highly iterative. In a first step I will collect text for a particular specialized subdomain by collecting existing documents and scraping additional text off the internet. Where possible, I will try to leverage existing Python packages like pypatent etc. for the scraping In a second step, I will preprocess the collected texts as described above. For this task, I will rely on existing NLP toolkits like NLTK or gensim. In a third step I will use three popular word embeddings: word2vec, GloVe and FastText. These word embeddings will be used pretrained, trained from scratch and retrained. The results will be compared with the described evaluation metrics.

13 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.