Topic Modeling on the Crowd RE Dataset using Unsupervised Machine Learning

Nicholas Alan Andrew Patrick Ford, Kim Julian Gülle

Technische Universität Berlin
nicholas.ford@campus.tu-berlin.de, kim.j.guelle@campus.tu-berlin.de

Abstract. Hier kommt eine kurze Zusammenfassung der Arbeit.

1 Introduction

In this paper, we aim to automatically analyze the Smart Home requirements collected Murukannaiah et al. for the Crowd RE project[8] in 2016. We will put ourselves in the perspective of a fictitious product owner, who wants to answer the following question:

> *Given a set of requirement sentences, what kind of features are my potential customers interested in the most?*

We consider our product owner to be working in a company which builds smart home appliances and deem the Crowd RE requirements to be the result of a survey that company has performed. The collected requirements therefore are the foundation of our analysis.
Considering the number of requirements (2966), we want to automate our analysis using the Python programming language and a word2vec model to derive a set of categories where the collected requirements can be assigned to. Finally, we want to answer our initial question based on the categories we found and the number of requirements assigned to each of the categories.

2 The Crowd RE Dataset

In an attempt to "facilitate large scale user participation in RE" [8] 609 Amazon Mechanical Turk users[1] were asked to submit requirements for smart home applicances in the Crowd RE project. The result was a dataset containing 2966 requirements, related to the domains *Energy, Entertainment, Health, Safety* and *Other*. The requirements were collected in two phases.
In the first phase the crowd workers were asked for their requirements of a smart home. The phase comprised three stages in which the workers were given a number of requirements and they were asked to add 10 requirements which are distinct to what they have seen. The requirements had to be submit through a form

[1] https://www.mturk.com/, last visited 2020-01-15

to ensure the requirement sentences follow the user story format[2]. Furhtermore, one of the aforementioned domains had to be selected as the *application domain* of the requirement. Finally, a comma separated list of tags could be added to the requirement. The resulting requirement would then look as follows:

> *"**As a** pet owner, **I want** my smart home to let me know when the dog uses the doggy door, **so that** I can keep track of the pets whereabouts."*[3]

In the second phase, the crowd workers were presented with the requirements produced in the first phase and they were asked to rate the requirements with regard to their clarity, usefulness and novelty. Note that for our analysis though, we only rely on the results of phase one and we mentioned the second phase solely for the sake of completeness.

3 Used Techniques

3.1 Natural Language (Pre-)Processing

In order to successfully perform an analysis of the dataset, we first needed to better understand the composition of the data. In a first step we therefore created and analyzed a corpus of requirements and compared the results to the Brown Corpus[4], a much larger generic corpus with words taken from books and news articles.

Indicator	Crowd RE	Brown
Number of Tokens (unique)	90,844 (5,024)	1,034,378
Number of Lexical Words	52,266	542,924
Vocabulary Size (Lexical Words)	4,906	4,6018
Vocabulary Size (Stems)	3,398	29,846
Average Sentence Length (Tokens)	31	18
Average Sentence Length (Lexical Words)	18	10
Lexical Diversity	0.011	0.054

Table 1. Data from the analysis of the Crowd RE dataset

In Table 1 we can see the number of tokens and lexical words is much larger in the Brown dataset which is a result of a wider variety of words in this kind of texts and is also because the brown dataset contains approximately 10 times more lexical words than the Crowd RE dataset. Even though the requirement sentences tend to be much longer, which may have also been caused by the

[2] As a [role] I want [feature] so that [benefit].

[3] The keywords marekd in bold text represent the placeholders which were already provided by the form to preserve the user story format.

prescribed user story format, the lexical diversity is lower. Requirements use domain-specifc expressions, so the same or similar words appear more often in the written requirements[3]. And it is also necessary to use unique words for the description of the same feature to avoid ambiguity. To sum up we can say that the results are as expected from a dataset that contains only requirements.

In order to derive meaningful data from a dataset which is as small as ours, we had to perform some Natural Language Processing (NLP) first, before further analyzing the data. A range of NLP techniques exist, which can be used to prepare the data for our kind of analysis[9][3]. The following list briefly describes the techniques we used in our research:

- **Tokenization** is...
- **Stopword-Removal**
- **Stemming**
- **Bag-of-Words**
- **TF-IDF**

3.2 Latent Dirichlet Allocation

After we developed our pre-processing pipeline for the dataset and some basic analysis on the data we have we decided to use the Latent Dirichlet Allocation (LDA) for a first topic modelling. The idea was to have another approach in the first step that we can use as intermediate result for the data and also to compare it to the result of the neural network to have some kind of benchmark or basis for a performance comparison.
The LDA is a probabilistic model that can be used for discrete data. It is a statistical approach that can be used to generate a topic model for text corpora [1]. The technique starts with selection a number of expected topics. The LDA then use all terms that are inside of the collection of documents and generates a polynomial distribution over all terms inside of the documents. Afterwards for each document a dirichlet distribution is performed which assumes that each document only contains a limited amount of topics. Target of the approach is to get the latent topcis that are are core of the document collection.

3.3 Word2Vec

Word2Vec is an open-source project for learning word embeddings and was created by Google Inc. in 2013[4]. The project incorporates the word2vec tool, which can be used to generate word embeddings from a given text corpus using two neural network architectures - the skip-gram model and the continous bag-of-words model (CBOW). Introduced by the same authors, these architectures aimed at

[4] `https://code.google.com/archive/p/word2vec/`, last visited 2020-01-17

optimizing the learning quality of the word vectors, while at the same time reducing the learning time to be able to train the model on data sets with billions of words[6]. According to their research, "none of the previously proposed architectures has been successfully trained on more than a few hundred of millions of words"[6, p1] and these architectures (which also includes the previously mentioned LDA) become computationally very expensive with larger data sets. Furthermore, the quality of the learned vectures by previous architectures is inherently limited for their "indifference to word order and their inability to represent idiomatic phrases"[7, p1]. This limitation was also important for us to consider during our analysis.
As a consequence of the user story format imposed to our requirement sentences a larger number of the requirements contained the phrase "I want my smart home to..." ($416/2966 \approx 14.03\%$). Also, the requested role description induced some of the participants to start their requirements with "As a smart home owner..." (8 requirements). Even though the latter example may be less relevant in its impact on our findings, it illustrates the problem of idioms just perfectly. Because when calculating the word vectors for these phrases using an LDA, the words "smart", "home" and "owner" would be represented by the same vectors. Hence, the phrase "a smart home owner" would always be represented with the same vectors and the vector distance of this phrase would be similar to both of the phrases "a clever home owner" and "an owner of a smart home". Especially after the stopwords were removed. It is rather obvious though, how these phrases could change the meaning of a statement completely.
A combination of two words which appear often together is called a bigram (or a 2-gram), but more words than just 2 could be involved making it a combination of an arbitraty number of N words and thus an n-gram. To detect these n-grams usually is done using statistical probability models, where one would analyze a given corpus of words to understand what words frequently occur close to each another[10]. The task of finding n-grams also is not just limited to finding related words, but started as a task of finding related characters which would follow one another[2] and was a measurement to improve the performance of automatic character recognition systems[10].
In their word2vec library, Mikolov et al. focused on the application for phrase detection though, to maximize the accuracy on the phrase analogy task. Using a dataset with about 33 billion words, they were able to train a model that reached an accuracy of 72% for the detection of phrase analogies[7, p6].

3.4 Word Mover's Distance

While word2vec is very sophisticated when it comes to generating quaility word embeddings, the CBOW method still has its weaknesses. Consider the two documents: *"My smart home should turn on my favorite music when I come to my home."* and *"My smart home shall play my most favored songs when I arrive at my place."* Even though the information is the same, the word vectors of these sentences will be different. Even though a word-wise similarity will be given (e.g. of the pairs $< music, songs >$, $< come, arrive >$) the closeness of the sentencens can not be represented by the CBOW model. To overcome this shortage, Kusner et al. introduced the Word Mover's Distance (WMD) in 2015 [5]. The WMD is a distance function which can be used to calculate the distance between these kind of text documents. Based on previously created word embeddings (as for example using the word2vec), the *"distance be- tween[sic!] two text documents A and B is the minimum cumu- lative[sic!] distance that words from document A need to travel to match exactly the point cloud of document B"*[5, p2]. Using this method, the WMD reaches a high retrieval accuracy, while being completely free of hyper-parameters and therefore straight-forward to use.

4 Related Works

ToDo: Add references to papers that have a similar approach...

- Paper about LDA for topic modelling

5 Analysis / Our approach

The Crowd RE dataset is available in form of a MySQL database dump, but the tables can also be downloaded separated into several *.csv* files[5]. For our research, we were only interested in the pure requirement sentences (without any ratings, or user characterization added to the data). We could therefore reconstructed the sentences from the *requirements.csv* file only, which is included in the downloaded data.

ToDo: Give the approach a name as title!

5.1 NLP Preprocessing Pipeline

As initially described in section 3.1 we preprocessed our requirement documents using an NLP pipeline as shown in figure 1. Implementing our solution in Python and following the common practice as suggested in [3], we made use of the NLTK library[6] to perform the NLP techniques we needed for our analysis. As some of the requirements sentences contained special characters, some initial data cleansing was necessary, to remove these special characters (i.e. spaces, dots, apostrophes, slashes) as they would have otherwise been ranked in the later used bag of words. We used regular expressions as provided by the Python standard library in order to do so. For the tokenization, the stop-word-removal and the stemming we used the functions provided by the NLTK API.

5.2 LDA Approach

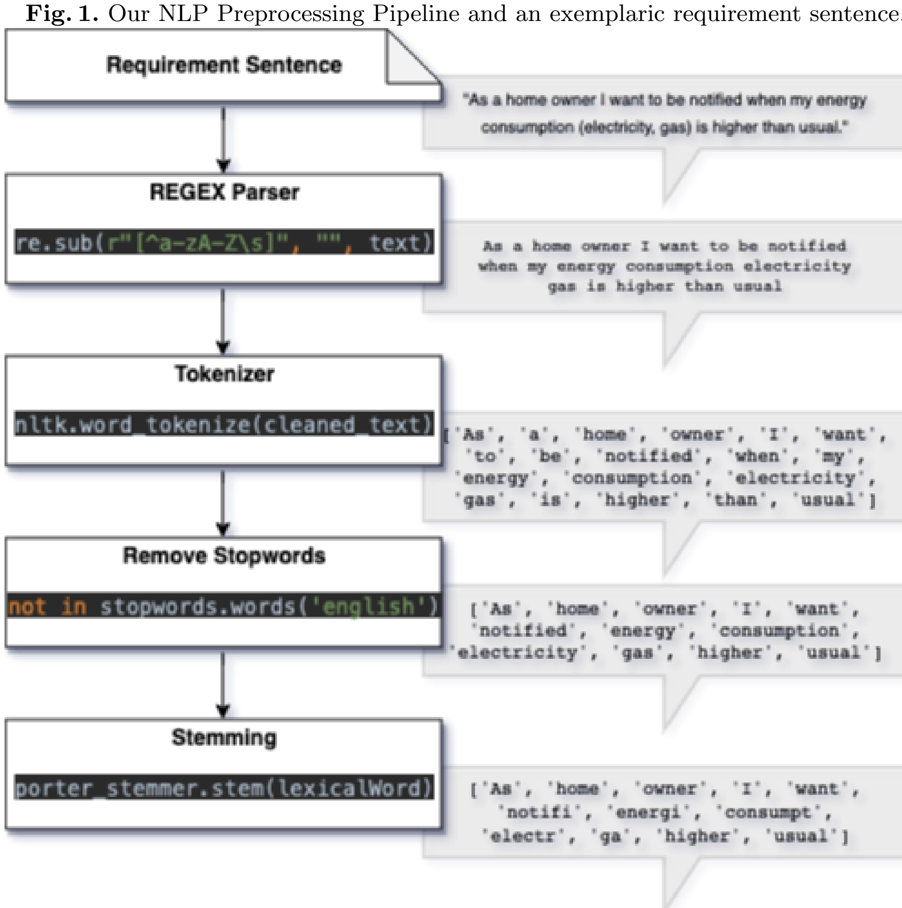
- how do we process the LDA on our dataset

- Bag of words - TF/IDF

[5] `https://crowdre.github.io/murukannaiah-smarthome-requirements-dataset/`, last visited 2020-01-15

[6] `https://www.nltk.org/`, last visited 2020-01-18

Fig. 1. Our NLP Preprocessing Pipeline and an exemplaric requirement sentence.



5.3 Neural Network

ToDo: How does our approach with the Neural Network looks like?

6 Findings

ToDo: Our results

6.1 LDA

ToDo: How good does the LDA perform?

6.2 Neural Network

ToDo: How good does our approach with the Neural Network perform?

6.3 Comparison

ToDo: Compare the results of the both methods

ToDo: Finally, our initial questions can be answered as follows:

ToDo: Discuss the results (maybe a new chapter for that?

As expected, our dataset was too small to achieve any better results. In this context, it is important to know how the accuracy of the word2vec phrase detection dropped to 66% when Mikolov et al. trained their model on a "smaller" dataset of 6 billion words[7, p7]. *Smaller* at least in comparison to their final training set, but this is still a lot larger than our dataset by a factor of almost 120.000.

Acronyms

CBOW	Continous bag-of-words
CSV	Comma Separated Value
LDA	Latent Dirichlet allocation
RE	Requirements Engineering

References

1. Blei, D.M.: Latent Dirichlet Allocation p. 30
2. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization p. 14
3. Ferrari, A.: Natural language requirements processing: from research to practice. In: Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18. pp. 536–537. ACM Press, `http://dl.acm.org/citation.cfm?doid=3183440.3183467`
4. Francis, W.N.: A standard corpus of edited present-day american english 26(4), 267–273, `https://www.jstor.org/stable/373638`
5. Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. pp. 957–966. ICML'15, JMLR.org
6. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. pp. 1–12
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality 26
8. Murukannaiah, P.K., Ajmeri, N., Singh, M.P.: Toward automating crowd RE. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). pp. 512–515. IEEE, `http://ieeexplore.ieee.org/document/8049175/`

9. Solangi, Y.A., Solangi, Z.A., Aarain, S., Abro, A., Mallah, G.A., Shah, A.: Review on natural language processing (NLP) and its toolkits for opinion mining and sentiment analysis. In: 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS). pp. 1–4. ISSN: null
10. Suen, C.Y.: n-gram statistics for natural language understanding and text processing PAMI-1(2), 164–172