Topic Modeling on the Crowd RE Dataset using Word Embeddings

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. We aimed to automatically derive topics from a dataset of 2966 requirement sentences. The requirements were previously collected, tagged and categorized by crowd workers as part of the Crowd RE project. We preprocessed the data in an NLP pipeline using state of the art NLP methods and applied topic modeling techniques to the results in order to cluster the data. This includes Latent Dirichclet Allocation, word embeddings using word2vec and the recently published Word Mover's Distance. We visiualized our findings in 2D scatter plots with the help of Principal Component Analysis and Stochastic Neighbor Embedding (t-SNE). All our programming work was done in Python and is strongly dependend on the nltk and gensim library. The requirement sentences had 5 different domains already assigned to them (including a domain called *Other*, which was neither of the other 4). Thus we expected to find 4 different clusters, with some noise between them, caused by the requirements of the *Other* domain. Having followed several approaches for topic modeling we found out that the Word Mover's Distance is probably the most promising, still we were only able to find 3 reasonably distinct clusters. The final verification, whether this means the pre-assigned categories are wrong or the dataset is too small for an automated topic modeling (or a combination of both) is a manual process and may be part of future work.

1 Introduction

In this paper, we aim to automatically analyze the Smart Home requirements collected Murukannaiah et al. for the Crowd RE project[15] 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" [15] 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 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 Background

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[5], a much larger generic corpus with words taken from books and news articles.

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

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

[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.

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

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 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[4]. 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[17][4]. The following list briefly describes the techniques we used in our research:

- **Tokenization** means separating a the data into tokens. Where tokens which are basically words. With tokenization whitespaces and all punctuation is removed from the data. As result a list of tokens is generated. The easiest tokenization is just splitting all alphanumeric characters.

- **Stopword-Removal** is removing common words from the data. They are often only required because of grammar or syntax.

- **Stemming** is a technique that reduce a word to just the root of the word. It eliminates duplicates that have the same meaning. This important in NLP as the as the conjugation of a word is not important if we are just interested in the semantic information that is contained in the words.

- **Bag-of-Words** is a technique that is used to simplify a sentence or document. The idea is to have a list of all containing words with the corresponding word-count in the text. It therefore only holds the word itself and the multiplicity (or in other words the frequency).

– **TF-IDF** can be separated into two different indices. TF: term frequency is the number of is a rating how often a specific term occurs in the data. IDF: inverse document frequency is a measure how much information a single term provides in relation to a document. The TF-IDF therefore is a rating how valuable a term for a document is.

3.2 Latent Dirichlet allocation

The Latent Dirichlet allocation is technique that can be used to observe groups of similar data. The LDA is a probabilistic model that can be used for discrete data. The LDA was supposed by Blei et. al in [1]. Within the LDA there are several terms that describe the data. The word is the basic unit of the discrete data. The collection of words is named as document and the set of documents is called corpus. The approach aims to find a limited number of topics that were latent inside of the documents of the corpus. To do so, the documents get *"represented as probability distributions over latent topics where each topic is characterized by a distribution over words"* [16]: The LDA uses all words 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 which is the basic assumption of this approach.

3.3 Word Embeddings

Being a probabilistic model, an LDA model describes the "statistical relationship of occurrences rather than real semantic information embedded in words" [16]. Without considering the semantic relationship between words, the similarity between words cannot be discovered, though [12]. This can result in too broad topics when performing topic modeling using LDA [16]. To overcome this shortcoming, continuous space neural network language models can be trained to capture both the syntactic and the semantic regularities of language. A defining feature of such models is that each word is converted into high-dimensional real valued vectors via learned lookup-tables [14].

Word2Vec In 2013, Mikolov et al. proposed two neural network architectures to calculate such vector representations of words: the continuous bag-of-words model (CBOW) and the continuous skip-gram model.

Word2Vec is an open-source project created by Mikolov et al. at Google Inc. for computing vector representations of words 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 continuous bag-of-words model (CBOW).

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

Introduced by the same authors, these architectures aimed at 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[12]. According to their research, "none of the previously proposed architectures has been successfully trained on more than a few hundred of millions of words"[12, 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 vectors by previous architectures is inherently limited for their "indifference to word order and their inability to represent idiomatic phrases"[13, 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 stop-words 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 bi-gram (or a 2-gram), but more words than just 2 could be involved making it a combination of an arbitrary 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[18]. 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[18].
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[13, p6].

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 [9]. 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"*[9, 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 Work

Using an Latent Dirichlet Allocation for topic modeling is not a new idea. Also Zhou et. al in [19] used this technique to automate a part of text mining. They used two different data for their research. At first they focus on articles from Wikipedia where they evaluated over 200.000 articles. They found out that from 50 topcis there are three topics with high probabilities compared to the others. They also used twitter messages from a set of 10.000 users. They found 30 topics five topics with the highest probabilities of the set of topics. They also mentioned that the processing time of their approach took quite long and might be improved in future works.

There are also researches using different techniques to get a topic model from a large amount of documents. George et. al in [7] proposed to use a 2D vector space model. They took over 300 unclassified document from 20 newsgroups dataset. As conclusion they mentioned that the performance of the algorithm need to be increased in future as the computation time consumes a lot of time.

Building up a pre-processing pipeline for our model approach was also performed in a similar way by other researchers. As Gemko et. al in [6] described they used also pre-processing for the automatic glossary term extraction. Their pipeline contains the steps of Tokenization, POS-Tagging, CHunking and Lemmatization. Additionally they apply some relevance filtering and specificity filtering afterwards. They also used the CrowdRE dataset and got well prepared data from their pre-processing pipeline to work with for their glossary term extraction. In they conclusion they figure out that the approach is generalized and not specific for the dataset so it can be used for any kind of data. Also it might be possible to add several filter stages to the pipeline to get reasonable data for the specific task that shall be performed afterwards.

In 2010, Řehůřek et al. wanted to automatically create a short list of similar articles to a given article[20]. They used Latent Semantic Analysis, as well as

LDA in their approach and created a Python library called *gensim*[5], which aimed at implementing these techniques in a clear, efficient and scalable way[6].

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[7]. 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.

To have a benchmark for the approaches we want to use we need a labelling at the dataset that we can use to rate how good the topic modelling worked. At first we checked if we can use the tags as soft labelling for the requirements but unfortunately most of them are only matched once (Total tags: 2116, tags that only occur once: 1562). We also checked the amount of requirements represented by the most common tags.

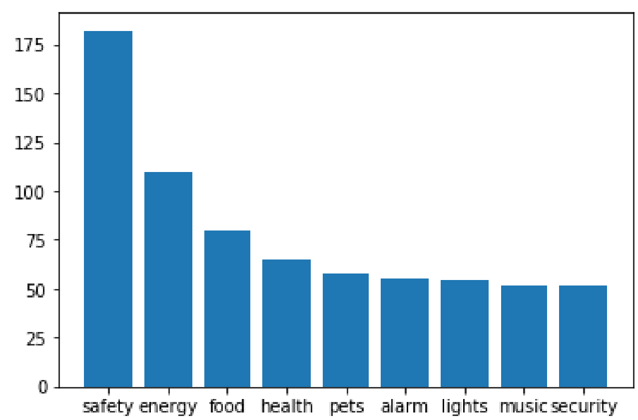


Fig. 1. Tag occurence and coverage of the requirements

In Figure 1 it is obvious that the high amount of tags that only occur once lead to the fact that the relation between requirements is not working good. The variety of tags that may be assigned to the same topic is very high and the low coverage of requirements with the top 9 tags makes the tags not suitable for the soft

[5]`https://radimrehurek.com/gensim/index.html`, last visited 2020-01-19

[6]`https://radimrehurek.com/gensim/about.html`, last visited 2020-01-19

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

labelling. So we checked the domains that were assigned to the requirements. The domains are separated into five groups: Health, Energy, Entertainment, Safety and Other. For the “Other“ there are again user defined specific domains, but we focus on the five top level domains for our labelling.

5.1 NLP Preprocessing Pipeline

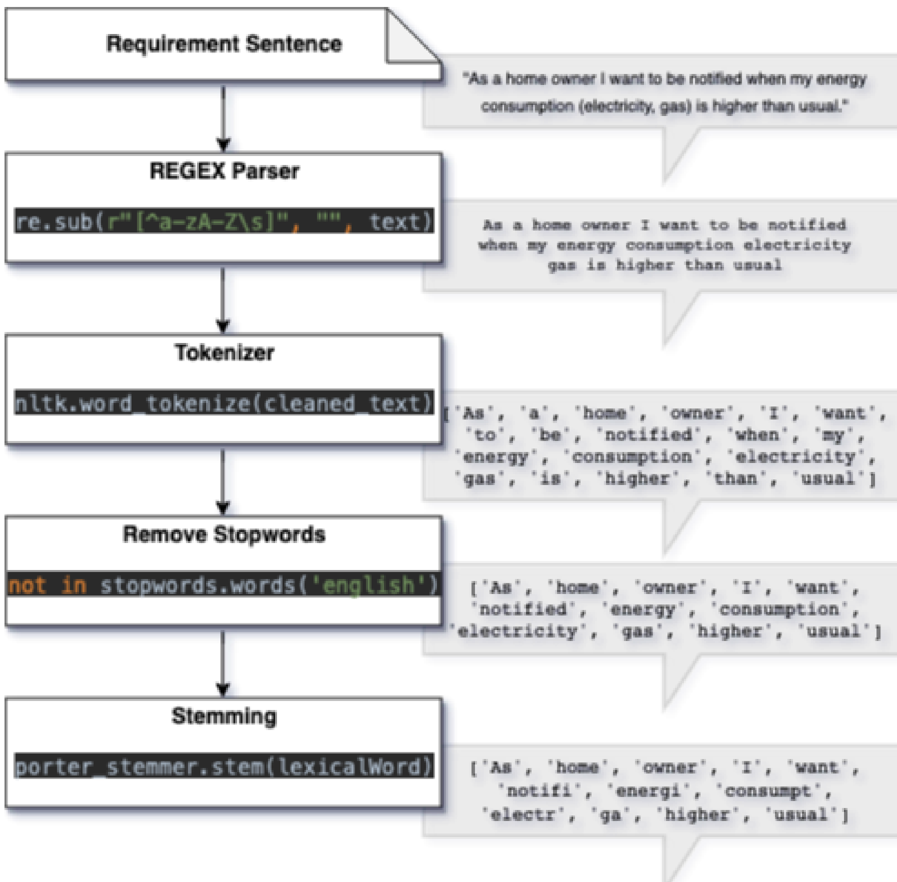


Fig. 2. Our NLP Preprocessing Pipeline and an exemplaric requirement sentence.

As initially described in subsection 3.1 we preprocessed our requirement documents using an NLP pipeline as shown in Figure 2. Implementing our solution in Python and following the common practice as suggested in [4], we made use of the NLTK library[8] to perform the NLP techniques we needed for our analysis.

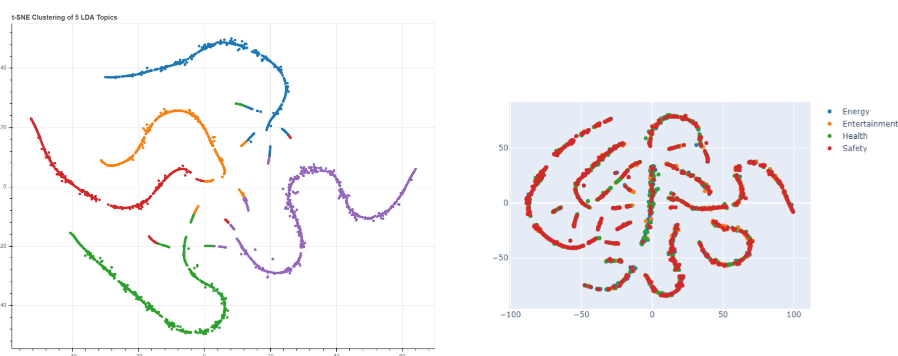
[8] `https://www.nltk.org/`, last visited 2020-01-18

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

After we developed our pre-processing pipeline for the dataset and some basic analysis on the data we have we decided to use the 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.

For our LDA approach we used our preprocessed requirements. To apply an LDA to the data we need an array of arrays where each inner array represents a single requirement (so in this dimension it has 2966 entries). each word in the requirements needs to be replaces by a number that can be processed by the LDA. So we first applied a bag-of-words which calculates a relative weight for the single words. the next step is rating the words with the TF-IDF. After these steps we have a prepared matrix that contains the data that now can be processed by the LDA.



(a) LDA Plot with colors to topics (b) LDA Plot with colors for domains

Fig. 3. LDA Result with TF-IDF (plotted with t-SNE)

In Figure 3 we can see the result of the LDA that is plotted using t-SNE. In figure 3(a) the colors are mapped to the found topics which of course looks better. But if we look for the expected mapping to the domains in figure 3(b) we can see that the found cluster doesn't match to the expected ones.

5.3 word2vec

For our word2vec approach, we made use of the gensim library[5], which we mentioned in section 4 already and which was also used by Solangi et. al in [17]. We mainly used the *gensim.models.Word2Vec* class, which implements both the CBOW and the skip-gram architecture of word2vec. We then followed different approaches, for the creation of our desired word embeddings. First, we tried to train our own model from the requiremnent senteces given the Crowd RE dataset both with and without prior processing of the requirements through our NLP pipeline and on both architectures.
Being unsatisfied with the outcome, in a second attempt we created our embeddings using a pre-trained word2vec model, which contained the word vectors of a model trained on about 100 billion words of the Google News dataset[9]. Even though the outcome was different, the underlying workflow for both approaches was very similiar once the trained model was available:

- For every tokenized requirement sentence, create a sentence matrix by replacing every word by its vector representation:
  $reqtokens = \{"As", "smart", "home", "owner", \ldots\}$
  $embeddings = \{\vec{as}, \vec{smart}, \vec{home}, \vec{owner}, \ldots\}$
- On the resulting matrices, reduce the different x-dimensions to the dimension of the shortest sentence using Principal Component Analysis
- Use K-Means to generate a number of clusters on these now equally shaped matrices
- Visualize the results by transforming the data into 2d space using t-SNE[11]

[9] `https://code.google.com/archive/p/word2vec/`, last visited 2020-01-19

```
plot_tsne(pcaed_arr.reshape(x, y*z), perplexity=50)
```

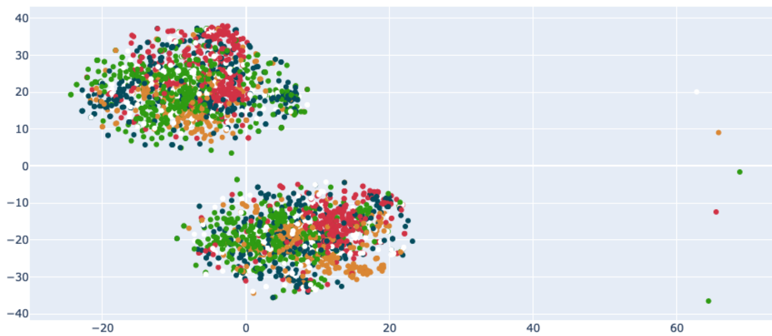


Fig. 4. word2vec result of the clustering with a pretrained model (plotted with t-SNE)

5.4 Word Mover's Distance

Finally, we used the Word Mover's Distance which was also implemented in the gensim library. To do so, it was also necessary to create word embeddings for our requirement sentences first. Again, we could use the word vectors of the aforementioned word2vec models. Instead of basing our clusters on the distance between word vectors though, we could now calculate a distance matrix which holds the calculated Word Mover's Distance from every sentence to every other sentence, as represented in Table 2. Note how the Word Mover's Distance is symmetric. So the distance to travel from s_1 to s_2 is the same as if your travelling vice versa.

	s_1	s_2	s_3	...
s_1	0	0.83	3.23	...
s_2	0.83	0	2.77	...
s_3	3.23	2.77	0	...
...	...	....	...	0

Table 2. Sentence Matrix containing the Word Mover's Distance from one sentence to another

Since by its nature, the resulting matrix already was a 2-dimensional array with equal dimensions, it was not necessary anymore to perform any further reduction. We could use this matrix instead to directly create some clusters using K-Means.

```
plot_tsne(distance_matrix)
```

Fig. 5. Distance Matrix of the Word Mover's Distance with a self-trained model (plotted with t-SNE)

6 Findings

For our approaches that have the task to generate topics for the given dataset we expect four different topics that match to the labeled domains: Energy, Entertainment, Health and Safety. The other data that was assigned to the topic Other is assumed as noise to the result. In our plots, we plotted the different requirement sentences. The coloring is based on the application domain they were associated with and is as follows: *Energy*, *Entertainment*, *Health*, *Safety*, *Other* (where requirements of the *Other* domain are represented by white markers).

6.1 LDA

Unfortunately the result of the LDA doesn't match the expected topics. The approach itself create 5 clusters that can't be mapped to the expected ones from the domain. But there is still some similarity between the requirements that are next to each other.

The calculaton of the LDA approach is very fast which means the performance of the algorithm fits good for that kind of task. But unfortunately the overall result is that we weren't able to gather topics from the given dataset.

6.2 word2vec

As shown in Figure 4 we could identify two clusters using the word2vec. Again, this did not match our expected 4 clusters (according to the different domains).

Any interpretation of the plotted clusters may be only speculative and highly subjective, which is why we did not make any assumptions with regard to the data quality, yet. Similar to the LDA, the preformance was relatively good and we did not wait for our results for much longer than a couple of minutes.

6.3 Word Mover's Distance

The best results we could achieve using Word Mover's Distance. What surprised us, as you can see in Figure 5, the results of the clustering with a word2vec model which we trained on our data was even better than those generated using the pre-trained word vecotrs of the the Google News model (see Figure 6). With our self-trained model we could distinguish three different clusters and we tend towards assuming, that clustering the requirements into these three clusters may even be more accurate than to sort them in 4 categories, as we initally intended to do. The higher quality of our results comes with a drawback of performance. On a current intel i5-9600K 6-core processor with 3.7 GHz and 32 GB of memeroy attached, the calculation of the Word Mover's Distance matrix took approx. 45 minutes (even after splitting up the calculation to be done in 12 parallel threads, where in each thread we would calculate the distance between two sentences). Some further improvements on the processing speed could still be made though, by not calculating the matrix as a whole, but taking advantage of the Word Mover's Distance symmetry.

```
plot_tsne(distance_matrix, perplexity=30, learning_rate=250)
```

Fig. 6. Distance Matrix of the Word Mover's Distance with a pre-trained model (plotted with t-SNE)

7 Conclusion

To sum up our research we can say that the used approaches work much better on large data sets and our dataset is compared to common used ones small. Also the soft labeling of the data that was done by the users themselves doesn't have a good quality. To increase the quality of the labeling manual checking with a lot of effort needs to be performed. The bad quality of the labeling might also result from a missing common understanding of the domains within the crowd workers.

In the end still a lot of manual work needs to be done to derive valuable results from the dataset regarding what features or feature categories are wanted the most in smart home application.

Finally we can summarize that the idea of CrowdRE works to some extend. One can gather a lot of requirements that are assigned to the smart home topic, but it is very difficult to analyze them automatically and requires a lot of effort to prepare them for further work.

8 Discussion

As expected, our dataset was probably 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[13, 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.

Also, all our word2vec approaches required to post-process the results using PCA. Though the PCA is a technique which is known for how well it can preserve the original information, some of the information will inevitably get lost. Our results may therefore have been impacted by the dimensionality reduction.

More time would have been needed for the evaluation of our results. Both the tags, as well as the application domains were set by the crowd-workers themselves. The quality of these assignemnts has not been proven yet and we used this data only for lack of proper testing data. For example, one of the requirements with the content "I want my smart home to sync with my biorhythm app and turn on some music that might suit my mood when I arrive home from work so that I can be relaxed" was related to the emphEntertainment domain. In our last approach this sentence was found to be in the *Health* domain using the Word Mover's Distance. We could not say that this assignement was definitely wrong, though. So it could be we find our approach a lot more successful after a thorough analysis of all the clusters.

Finally, we lacked prior knowledge of the field of topic modeling and machine learning in general. Though we performed our research with technical and professional care in all conscience and under consideration of commonly accepted principles, there may be a lot of potential for further optimization (which goes beyond changing the hyper-parameters of our models). Future works could be done on an improved set of data, by only analyzing those requirements which have clearly defined domains. Such dataset could be achieved, by cleaning the current Crowd RE dataset by the means of manually labeling the requirement sentences. This includes verifying the currently assigned application domains, reassigning some domains and also creating new domains, which may not have been in the domain-selection when the requirements were created.

When manually reviewing the dataset, our results could also be improved through cleaning the dataset. In accordance with [3] and [8] cleaned datasets have a much higher impact on the training results of ML models than the optimization of hyper parameters.

- Li et al. also created a classifier using WMD [10]. Future works could use their approach for a comparison of the generated clusters on the Crowd RE dataset

Acronyms

CBOW	Continous bag-of-words
CSV	Comma Separated Value
LDA	Latent Dirichlet allocation
ML	Machine Learning
PCA	Principal Component Analysis
RE	Requirements Engineering
TF-IDF	Term Frequency, Inverse Document Frequency
WMD	Word Mover's Distance

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. Chu, X., Ilyas, I.F., Krishnan, S., Wang, J.: Data cleaning: Overview and emerging challenges. In: Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16. pp. 2201–2206. ACM Press, http://dl.acm.org/citation.cfm?doid=2882903.2912574

4. 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
5. Francis, W.N.: A standard corpus of edited present-day american english 26(4), 267–273, https://www.jstor.org/stable/373638
6. Gemkow, T., Conzelmann, M., Hartig, K., Vogelsang, A.: Automatic Glossary Term Extraction from Large-Scale Requirements Specifications. In: 2018 IEEE 26th International Requirements Engineering Conference (RE). pp. 412–417. IEEE, Banff, AB (Aug 2018), https://ieeexplore.ieee.org/document/8491159/
7. George, M.: Unsupervised Topic Detection based on 2d Vector Space model using Apriori Algorithm and NLP. In: 2018 Thirteenth International Conference on Digital Information Management (ICDIM). pp. 279–283 (Sep 2018), iSSN: null
8. Krishnan, S., Wang, J.: Data cleaning: A statistical perspective - overview and challenges part 2, https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxkYXRhY2xlYW5pbmd0dXRvcmlhbHNpZ21vZDE2fGd4OjJhMzc4ZWExM2U3MzA3MGE, ACM SIGMOD/PODS Conference
9. 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
10. Li, C., Ouyang, J., Li, X.: Classifying extremely short texts by exploiting semantic centroids in word mover's distance space. In: The World Wide Web Conference. pp. 939–949. WWW '19, Association for Computing Machinery, https://doi.org/10.1145/3308558.3313397
11. Maaten, L.v.d., Hinton, G.: Visualizing data using t-SNE 9, 2579–2605, http://www.jmlr.org/papers/v9/vandermaaten08a.html
12. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. pp. 1–12
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality 26
14. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 746–751. Association for Computational Linguistics, https://www.aclweb.org/anthology/N13-1090
15. 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/
16. Niu, L.Q., Dai, X.Y.: Topic2vec: Learning distributed representations of topics http://arxiv.org/abs/1506.08422
17. 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
18. Suen, C.Y.: n-gram statistics for natural language understanding and text processing PAMI-1(2), 164–172
19. Zhou Tong, H.Z.: A TEXT MINING RESEARCH BASED ON lda TOPIC MODELLING. Computer Science & Information Technology (CS & IT) pp. pp. 201–210 (2016)

20. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: New Challenges for NLP Frameworks. ELRA, http://is.muni.cz/publication/884893/en