Research on NLP for RE at Fraunhofer FKIE: a Report on Grouping Requirements

Daniel Toews
Fraunhofer FKIE
Wachtberg 53343
daniel.toews@fkie.fraunhofer.de

Timm Heuss
Fraunhofer FKIE
Wachtberg 53343
timm.heuss@fkie.fraunhofer.de

Abstract

In this report we describe the previous research done by our institute in the field of requirement analysis using different natural language processing methods. To represent the different degrees of similarity between words we implemented different methods that make use of synonyms and hyperonyms. We present the strengths of our methods and identify their weaknesses. For our future research we want to incorporate Word Embeddings as they solve most of the difficulties we faced with synonyms and hyperonyms.

1 Team Overview

In our interdisciplinary research team at the Fraunhofer FKIE, we combine practical expertise in requirement engineering of large-scale commercial projects as well as knowledge in the field of Natural Language Processing (NLP), Data Science and Machine Learning, Software Architecture, Systems Engineering and Linguistics. It is our goal to provide novel approaches, algorithms and prototypical implementations to our customers to help them with their daily businesses, such as large commercial requirement engineering projects.

2 Past Research on NLP for RE

Over the past year our team at Fraunhofer FKIE conducted research in clustering and analysing requirements. This research was started because of the challenges that requirements engineers and managers faced at our institute, which include finding duplicate requirements as well as requirements formulated with different words for the same semantic context. This leads to requirements stocks that are hard to maintain.

One way to address these challenges is to introduce a grouping for requirements, providing requirements engineers and managers a better overview over the data. Since this is an extensive and time consuming task, especially when looking at hundreds of requirements, an automatic system would improve the work flow of those people significantly. Thus a combination of approaches and algorithms is desired to maximize the gain for the people working on those requirement sets.

2.1 General Setup

To identify such a combination we implemented multiple algoriths to improve the clustering of their methods (such as k-Means [Llo82], ClusterART [Mas03] and Neural Gas [MS+91]) and distance functions (such as euclidean distance and manhattan distance). Following the common Bag of Words approach, the requirements were translated into vectors that represented how often certain words appeared. This resulted in over a thousand of dimensions and thus leaving us vulnerable for the curse of dimensionality. This "curse" describes problems that arise when trying to calculate the distance between objects high dimensionality. With higher dimensionality distances grow more alike and thus it is hard to differentiate between similar and not similar objects [SEK04].

We tried to reduce the number of dimensions by using stopwords filtering and lemmatization of words. Additionally we used tf-idf as suggested by the literature [SM86].

2.2 Using Synonyms for Requirements Clustering

We tried to mitigate the effects that using different words for similar semantic intentions (e.g. using *broadcast* instead of *send*) had on the distances of requirements by using synonym lists. The idea is to replace each word with a specific synonym, such that words like *transfer* and *broadcast* would be replaced by the word *send*. This reduced the number of dimensions by a small amount, as well as putting requirements with those similar words closer together. However, this makes words that are similar into the same word, thus having no distance between them. But since those words are not the same, a more nuanced method, in which words can be similar without being the same word, would be preferred.

2.3 Using Ontologies for Requirements Clustering

In a next step we enriched the text by using an ontology. This allowed us to determine hyperonyms of a word and then enrich the text with this information. The hyperonym of a word is a word describing the more general concept. For example, the hyperonym of *car* would be *vehicle* and the hyperonym of *red* would be *color*. It is noteworthy that words can have multiple hyperonyms. For instance, the word *watch* is a *clock* as well as an *accessoire*. For our method we enriched the text by adding all hyperonym information available to this word (meaning the different hyperonyms as well as the hypernyms of those words and so on) to the sentence. This information can contain more then ten or twenty words and thus increases the size of the sentences, as well as more than double the dimension size. However, similar words like *car* and *truck* will then share some words in this tree, increasing the similarity between both sentences, while also describing the differences between the words with another subtree. For our tests we made use of the GermaNet ontology.[1]

2.4 Difficulties with Synonyms and Hyperonyms

Even though both approaches have theoretical advantages, in our experience they proved to be not perfect. While our synonyms approach reduced the dimensionality, it provided only a binary similarity between words, either they are the same or not. In contrast, the hyperonym approach with an ontology was able to provide more differentiated information on words, but in doing more than doubled the number of different words in our corpus.

Additionally, we faced another problem with those approaches. In general, words can have multiple meanings, for example homonyms, like the word *bear*, which can describe a mammal, as well as tolerating or enduring something. But other words can also have slightly different meanings. The word *group* can describe multiple things, for example a criminal group (*gang*) or a musical group (*band*). In both cases a collective of persons is described, but both would have different synonyms and hyperonyms. Currently we designed an algorithm that tries to solve this problem by previously determining the context of the requirements (is the text about criminals or musicians?) and then chooses the appropriate meaning of the word.

2.5 First Evaluation

In a first evaluation we tried to determine the impacts of different parts on the clustering quality. We decided to use external metrics to determine the quality of our algorithms. As we wanted to measure the potential support our software could have on requirements engineers, we measured the algorithms against a grouping provided by our own experts. They grouped the technical requirements by their topic, meaning which technical subparts were specified. The grouping was done to get a better overview about the current status of the requirement stock.

This means that we had to use a fixed number of clusters to compare against the hand crafted grouping. Additionally this meant that we judged the quality of the algorithm on how well it was able to recreate a human made grouping, while dismissing potentially good results that were specific to the algorithm. Thus we wanted

[1] http://www.sfs.uni-tuebingen.de/lsd/tools.shtml, last access on 2018-01-25

requirements to be treated as similar, when the specified technical concept of the requirements was similar. Despite these restrictions on our evaluation, we were not able to determine a better way of measuring the quality of our algorithms, as we did not think that for our case internal metrics were suitable. How to improve the evaluation of our requirement analysis framework is thus an open question for our team.

We tested multiple combinations of algorithms, distance functions, modifiers (such as lemmatization and stop words filter), as well as our synonyms and ontology approaches. We calculated the F1 scores for all those combinations and found small indications showing good combinations. Multiple of those good combinations contained our synonyms as well as our ontology approach. We published our results[2] as well as parts of our algorithmic framework[3] on GitHub.

3 Research Plans on NLP for RE

Today sophisticated word embedding algorithms, like word2vec [MCCD13], GloVe [PSM14] and fasttext [BGJM16], are able to place words in a vector space that contains semantic information about the words. Thus similar words will be placed close to each other and form groups of words. This means, that the distance between two words in this space can describe how semantically similar they are. Meaning that synonyms as well as hyperonyms of a certain word will be placed close to each other, while non related words will have a higher distance. Additionally, the words have a fixed dimensionality (generally orders of magnitude lower than the number of different words in the corpus). Previous work in requirement analysis discusses the possibility of integrating Word Embeddings [FDE+17] or integrates them in classification approaches [WV16]. Additionally, Lucassen et. al also use the approach for clustering requirements [LDvdWB16].

Looking closer at word2vec and fasttext, these models are generated by reading lage sources of human text, such as Wikipedia, and trying to determine the word positions of word x by looking at words used around x. In short, the model is trained by looking at n words before and after x and is then encouraged to guess x using only the surrounding words (other methods like the Skip-Gramm Model [MSC+13], GloVe [PSM14] or fasttext [BGJM16] prove to be more precise, but would be too extensive in this report). Thus the position of x in the vector space is determined by the words often used with x. The assumption is (and proves to be correct), that similar words are used with the same surrounding words. Because of this training, we assume that the model trained with an appropriate dataset contains finer nuances of words like *group*, as the surrounding words will catch the different meanings and contexts (see the example in 2.4 the word can be used with).

Looking back again at the difficulties explained in section 2.4, word embeddings seem to solve exactly those problems. The fixed dimension size of the vectors around 200 to 500 dimensions (as recommended by Mikolov et. al [MCCD13]) is significantly less than our datasets containing thousands of requirements, with more than 1800 unique words, not counting the number of words that occur when using our ontology approach. This means that the number of dimensions in each vector is reduced significantly and thus the calculation of the distances is more meaningful [SEK04]. Words which are synonyms to each other will be placed close in the vector space and thus result in a low distance between those words, while still maintaining their identity. Further, the different meanings of the words are taken into consideration in the vector space due to the nature of the training. Thus, we hope to improve our clustering results even further when using this approach. Additionally, we want to try to use them together and evaluate these results.

Another open question we face is the training of the model. The fasttext framework provides pre trained models on 294 languages, using Wikipedia as training data[4], but requirements are mostly specific to a certain domain and so the words may not be presented well or at all in Wikipedia. Thus, using a model trained on a domain specific model may also improve the results.

Another research question is how to define similarity between requirements when using word embeddings. Previously, we defined a requirement by adding all words together into one vector which represented the words and their number of occurrences in this requirement. However the number of words in a requirement is now effecting the result in a more meaningful way [KDR15]. Averaging the resulting vector over the number of words could be one solution, but is not feasible, since the average of two sentences can be the same even though they do not share common words with each other. This is due to the fact that the position of words and the spaces between them carries semantic meaning.

[2]https://github.com/fkie/requirement-clustering-evaluation-2017
[3]https://github.com/fkie/requirement-clustering
[4]https://github.com/facebookresearch/fastText#enriching-word-vectors-with-subword-information

Because of these problems Kenter et. al suggest a different formula called *semantic text similarity (sts)* [KDR15]. The idea in short is that for the distance of two sentences (requirements) each word of sentence s_1 will be matched to the best corresponding word of sentence s_2. The distance of those words will be calculated and added together. The exact formula is:

$$f_{sts}(s_l, s_s) = \sum_{w \in s_l} IDF(w) * \frac{sem(w, s_s) * (k_1+)}{sem(w, s_s) + k_1 * (1 - b + b * \frac{|s_s|}{avgsl}}$$

Here s_l is the longer of the two sentences, while s_s is the shorter one. The parameter *avgsl* describes the average sentence length of the data set, while b as well as k_1 smooth the results and are set to $b = 0.75$ and $k_1 = 1.2$ on default. The function $sem(w, s)$ describes the semantic similarity of the word w w.r.t sentence s:

$$sem(w, s) = \max_{w' \in s} f_{sem}(w, w')$$

where $f_sem(w, w)$ describes the similarity of both words. For this, the cosine similarity is suggested.

In the next step of our research we will use pre trained fasttext models for clustering requirements and the f_{sts} formula to determine the similarity between requirements.

References

[BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[FDE+17] Alessio Ferrari, Felice DellOrletta, Andrea Esuli, Vincenzo Gervasi, and Stefania Gnesi. Natural language requirements processing: A 4d vision. *IEEE Software*, 34(6):28–35, 2017.

[KDR15] Tom Kenter and Maarten De Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM, 2015.

[LDvdWB16] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn EM van der Werf, and Sjaak Brinkkemper. Visualizing user story requirements at multiple granularity levels via semantic relatedness. In *International Conference on Conceptual Modeling*, pages 463–478. Springer, 2016.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[Mas03] Louis Massey. On the quality of art1 text clustering. *Neural Networks*, 16(5):771–778, 2003.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MS+91] Thomas Martinetz, Klaus Schulten, et al. A" neural-gas" network learns topologies. 1991.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[SEK04] Michael Steinbach, Levent Ertöz, and Vipin Kumar. *The Challenges of Clustering High Dimensional Data*, pages 273–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[SM86] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[WV16] Jonas Winkler and Andreas Vogelsang. Automatic classification of requirements based on convolutional neural networks. In *Requirements Engineering Conference Workshops (REW), IEEE International*, pages 39–45. IEEE, 2016.