Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text

S Tiun, U A Mokhtar, S H Bakar and S Saad

1ASLAN Lab, Centre of Artificial Intelligence (CAIT), Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
2Information Governance Lab, Center for Cyber Security (CYBER), Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
3Center for Software Technology and Management (SOFTAM), Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
4ONTOS Lab, Centre of Artificial Intelligence (CAIT), Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

sabrinatiun@ukm.edu.my

Abstract. One of the needs in adopting a crowdsourcing approach in software requirement system (SRS) is to be able to perform text analytics to gain insight or knowledge from the crowd's feedback. One of the expected text analytic tasks is to be able of analyzing the feedback automatically; such as, whether the feedback concerns about the functional requirement (FR) and non-functional requirements (NFR). To automatically do the FR and NFR identification, one can treat the problem as a text classification task. Performing automatic identification requires features (word representation) that contain sufficient information which can be used to do the identification. Thus, the types of features can be considered as an important step in performing automatic identification of FR and NFR. In this study, the RE’17 dataset challenge is used as the dataset. Using the dataset, we will like to find out the effect of word embedding against traditional features (such as bag-of-words) in NFR and FR classification. In addition, we also want to find out whether is necessary to use a complex neural classifier to obtain the best performance of NFR and FR classification. Based on the obtained results, using fastText seems to be the promising classification model since the model obtained the highest F1-score of 92.8%.

1. Introduction

The use of natural language processing (NLP) in the study of software requirements is not new[11]. In fact, with the rising of more advanced document representation viz. word embedding[1,2,10], NLP is seen to be more adapted in the study of software engineering. For example, one of the needs in adopting a crowdsourcing approach in software requirement system (SRS) is to be able to perform text analytics to gain insight or knowledge from the crowd (users)'s feedback. One of the expected text analytic tasks is to be able to identify what kind of feedback given by the users (crowd).

Software requirements can be classified as functional (FR) e.g. explicit features, or functions, of the product or non-functional (NFR) e.g. implicit quality criteria for the product [5]. Identification of the
FR and NFR allows software specialists to immediately locate which requirements interest them without
needed to peruse through the entire SRS (e.g., the UX designer is likely interested in look-and-feel
requirements)[3]. However, the manual task of labeling what category a requirement falls under is
tedious and requires domain experts. In other words, manual labeling is expensive and time-consuming.
Thus, this is a strong motivation for the need to have an automated method with high-performance
accuracy of FR and NFR identification.

Automatic of FR and NFR identification can be seen as the task of a text classification problem.
Performing automatic identification requires features that contain sufficient information that can be used
to do the identification. In this study, we revisit the RE 17 Dataset Challenge [5]. One of the challenges
of RE17 Dataset is to automatically categorize FR and NFR of a software product. There are a few
works that have worked on the challenge, such as [3] and [4]. In [3], Word2vec was used as a feature
with the convolutional neural network (CNN) as the classifier. In their work, the NFR classification
achieved more than 90% F1-score. In [4], they classified FR and NFR using supervised machine learning
with various types of lexical features. The best obtained result was a recall and a precision of ~92%
using the SVM classifier on lexical features.

In this paper, we will study a few word embeddings as the text feature on traditional (linear)
supervised machine learning classifiers, namely SVM, Naïve Bayes (NB) and Logistic Regression (LR).
The detail on our research method will be explained in section 2, then the obtained results and discussion
will be presented in section 3. In section 4, we will conclude our study and present future works.

2. Research Design

In this study, we will see the effect of complex features representing the requirement text in RE’17
Dataset. As for feature benchmark, we use bag-of-words (BOW) and term frequency-inverse document
frequency (TFIDF). To see the effect of complex document features, we implement two types of word
embeddings; Doc2vec (extension of Word2vec)[8] and fastText [6]. We also use three types of
traditional machine learning classifiers (SVM, NB, and LR), as the benchmark classifier against the
CNN classifier used in [3]. Thus, the experiment design was setup based on the following questions:

RQ 1: How is the classification performance of NFR and FR, if different word embeddings with
traditional classifiers are compared against deep learning classifiers (the benchmark)?
RQ2: How is the performance of the traditional classifier with common statistical text features
(e.g. BOW or TFIDF) when compared against more complex representations and classifiers (i.e.
Word2vec and CNN) in NFR and FR classification?

2.1. Dataset

The dataset in this study is a RE ’17 Data Challenge Area 2[5], which focuses on identifying functional
(FR) and non-functional (NFR) requirements. In the dataset, requirement text of FR category is
labeled with 'F' and NFR as 'A, L, LF, MN, O, PE, SC, SE, US, FT, PO' (see figure 1). Since this paper
aims to only categorize requirement text into FR and NFS, thus for categories besides the 'F' label, we
wrote the requirement text as NFR. In our experiment, a total of 625 requirement text is used as the
dataset, in which 56% is in NFR class and the rest is FR class.

Figure 1. Excerpt from the RE’17 Dataset Challenge [5].

2.2. Method

To answer the research questions (RQ1 and RQ2), we had set up an experiment consisted of two types
of features: word embeddings and traditional (statistical) text features. The method of experimenting is
the same as a text classification problem. The experiment started by pre-processing the requirement
text, which is, removing symbols and applying lowercasing letters. Afterward, feature extraction was
conducted. The extracted features were BOW and TFIDF and the trained embedding models as
features were Doc2vec and fastText. BOW and TFIDF were extracted using CountVectorizer and
TfidfVectorizer, respectively, of the Scikit-learn package[7]. The detail on Doc2vec and fastText will
be explained in the following subsections.

2.2.1 Doc2Vec

The Doc2vec[9] feature is a paragraph embedding model, an extension of the Word2vec embedding
model. Thus, instead of word representation vector, Doc2vec is a representation of a text. The text can
be sentences, phrases or documents[8]. Doc2vec was built based on a concept known as paragraph
vector [8]. The idea is simply adding another paragraph ID on the Word2vec model. Analogous to the
Word2vec model, Doc2vec consists of two models; paragraph vector with distributed memory (PV-
DM) and paragraph vector with distributed bag-of-words (PV-DBOW)[8]. As seen in figure 2, to do the
prediction of a label or class given a text, the average of the concatenated words vectors will be used as
the feature[8] for text classification.

Figure 2: Illustration of PV-DM - Paragraph vector of distributed memory analogous to Word2vec-
skip-gram model[8]

In our study, a combination of PV-DM and PV-DBOW of Doc2vec is used as our text embedding
model. The combination of the two Doc2vec is more stable than using either, and eventually will
increase the FR and NFR classification performance[8][9]. In the training of the Doc2vec model for
our FR and NFR classification, the set of requirement text was used as the text. After being trained, the
Doc2vec can be used as features in classifying whether the requirement text belongs to FR or NFR
classes.

2.2.2 fastText. We include the fastText embedding model in our experiment. This is due to fastText is
claimed not only simple but more efficient than the far complex classifier model e.g. deep learning
classifier[8]. Adopting the same concept as [8], the document/sentence representation in the fastText
model is formed based on the average of N-gram features (fastText is an N-gram based vector and not
word vector as in Word2vec) embedded in a hidden variable. The hidden variable then can be used as
features (input) for any linear classifier model[8], and in [8], fastText uses a logistic regression
classifier. Figure 4 illustrates how fastText model represents a sentence (requirement text) with a
hidden variable where N-gram features of the sentence, x1..xN are embedded and averaged[8].
Figure 3: Illustration of the fastText model represents a sentence where the N-gram features \( x_1..x_N \) are embedded and averaged at a hidden variable. The hidden variable is the representation of the sentence [8].

Once all the needed features have been prepared, each of the features was used (except fastText) on the three traditional supervised classifiers: SVM, NB, and LR. Since obtained results needed to be compared with the benchmarks [3-4], F1-score, precision, and recall were used as the measurement metrics.

3. Results and Discussion

Given the split of the dataset as of NFR and FR into 80/20. The following are the obtained results. For the RQ1, Table 1 presented the results and Table 2 for the answer of RQ2.

Table 1. Results of the NFR and FR classification for RQ1.

| Features and Classifiers | F1-score | Precision | Recall |
|--------------------------|----------|-----------|--------|
| Combination-Doc2vec+LR   | 82.66    | 82.25     | 84.00  |
| Combination-Doc2vec+NB   | 82.55    | 82.11     | 83.58  |
| Combination-Doc2vec+SVM  | 73.47    | 83.46     | 72.12  |
| Word2Vec+CNN[3]          | 92.16    | 92.68     | 91.87  |
| fastText                 | **92.8** | **92.8**  | **92.8** |

Table 2. Results of the NFR and FR classification for RQ2.

| Features and Classifiers | F1-score | Precision | Recall |
|--------------------------|----------|-----------|--------|
| BOW+LR                   | 89.26    | 88.72     | 90.82  |
| BOW+SVM BOW+NB           | 61.17    | 84.57     | 62.76  |
|                          | 91.40    | 91.80     | 91.05  |
| TFIDF+LR                 | **92.41**| **91.99** | **92.96** |
| TFIDF+NB                 | 91.13    | 93.48     | 89.78  |
| TFIDF+SVM                | 38.42    | 31.2      | 50.00  |
| Lexical features +SVM[4] | ~92      | ~92       | ~92    |
| Word2Vec+CNN[3]          | 92.16    | 92.68     | 91.87  |
In Table 1, the results clearly show that given Doc2vec as the feature, the classifier performance is not very encouraging compared to fastText. This can be seen with the lowest F1-score with only 73.47% on the SVM classifier. The highest F1-score is 82.66% using Doc2vec feature on the LR classifier. One possible reason behind the poor performance of Doc2vec compared to fastText is due to Doc2vec only great when it is used with large number of vocabularies. When small size data or vocabulary is used, the model will experience the sparseness problem. In [3] the Word2vec is performing well (F1-score = 91.6%) and at par with fastText. The explanation for the impressive result is due to the two enhancements when Word2vec is used as text representation. The first one is, they use the vocabulary from Google News to fill in any zero vector of the FR and NFR dataset. Second, the used classifier is based on a more competitive and powerful neural network classifier, the CNN. CNN is a deep learning classifier that can learn more features than a simple classifier like SVM, NB or LR. However, fastText is considered a favor embedding model for FR and NFR classification in this paper since with a simple N-gram representation and just using a linear classifier, the result obtained (F1 = 92.8%) was superior compared to the combination of Doc2vec or the benchmark[3]. Furthermore, to obtain a good result, it does not require complex classifiers, like CNN. Therefore, for the classification of FR and NFR, fastText is the most suitable embedding feature and classifier model.

In Table 2, we want to find the answer of RQ2, which is, what if traditional (statistical) features with traditional (linear) classifiers are used in NF and FR classification compared to more complex features. In Table 2, using SVM as the classifier, the traditional feature of TFIDF seems to have the worst performance with only 38.42% for F1-score, whereas in [4], they obtained a high result with F1-score of 92% (using far more complex features). TFIDF is great with a lengthy document, and since the requirement text of FR and NFR is very short, which mostly contains a single sentence, this can explain the poor performance of TFIDF with SVM. However, when a different classifier was used on the TFIDF feature, the model does not seem to generate the same result. Using TFIDF on LR, the result was quite impressive with an improvement of around 0.24% than the obtained results of [3]. Thus, from Table 2, we can conclude that using TFIDF and LR is the best model if one wants to only consider traditional (statistical) feature and traditional (linear) classifier model.

4. Conclusion and Future Works
Overall, we can conclude that fastText is the best model for classifying the FR and NFR. The superior result given by fastText compared to Word2vec feature with deep learning classifier concludes that using deep learning classifier does not necessarily will outperform the linear classifier in text classification problem. For a binary text classification with a very short length of a document and least number of vocabularies, fastText can do a better job. Besides, using just traditional features and classifiers will be sufficed to classify the FR and NFR problem if one prefers a simple text classification model. Thus, from our study, we suggest, if one prefers to use traditional features and classifiers on a similar FR and NFR classification, they should consider using TFIDF with NB as their model. For future work, we will try to see if there is any increase of performance if more enhancement is done on TFIDF for FR and NFR classification due the impressive obtained result of TFIDF with NB. In addition, we also interested to see how the rising text representation like Bidirectional Encoder Representations from Transformers (BERT) will perform on FR and NFR classification.

Acknowledgment
This research is funded by Universiti Kebangsaan Malaysia under Research Code: GGPM-2017-025.

References
[1] Efstathiou V Chatzilenas C and Spinellis D 2018 Word Embeddings for the Software Engineering Domain 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), Gothenburg pp 38-41
[2] Tiun S Nor N F M Jalaludin A and Rahman A N C A 2020 Word Embedding for Small and
Domain-specific Malay Corpus Computational Science and Technology - 6th ICCST 2019 (Lecture Notes in Electrical Engineering Vol 603) eds R Alfred Y Lim H Haviluddin and C K On (Springer Verlag) pp 435-443

[3] Dekhtyar A and Fong V 2017 RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow IEEE 25th International Requirements Engineering Conference (RE) Lisbon pp 484-489

[4] Kurtanović Z and Maalej W 2017 Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning IEEE 25th International Requirements Engineering Conference (RE) Lisbon pp 490-495

[5] Cleland-Huang, J Settimi R, Zou X and Sole P 2006. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects IEEE 14th International Requirements Engineering Conference (RE) Minnesota pp 36-45

[6] Joulin A Grave E Bojanowski P and Tomas Mikolov T 2017 Bag of Tricks for Efficient Text Classification, 15th EACL Spain pp 427–431

[7] Pedregosa F Varoquaux G Gramfort A et al., 2011 Scikit-learn: Machine Learning in Python Journal of Machine Learning Research 12, pp 2825-2830,

[8] Le Q and Mikolov T 2014 Distributed Representations of Sentences and Documents Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2) pp 1188-1196

[9] Rehurek R, and Sojka P 2010 Software framework for topic modelling with large corpora Proceeding of LREC 2010 Workshop on New Challenges for NLP Frameworks Malta pp 45-50

[10] Liza F F and Grzes M 2016 An Improved Crowdsourcing Based Evaluation Technique for Word Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP Berlin August pp 55–61

[11] Ernst M D 2017 Natural Language is a Programming Language: Applying Natural Language Processing to Software Development 2nd Summit on Advances in Programming Languages - SNAPL 2017 (Leibniz International Proceedings in Informatics) eds Lerner B S Rastislav Bodík R and Krishnamurthi S Article 4 pp 1–14