Simple App Review Classification with Only Lexical Features

Faiz Ali Shah, Kairit Sirts and Dietmar Pfahl
Institute of Computer Science, University of Tartu, J. Liivi 2, 50409, Tartu, Estonia

Keywords: App Review Classification, Convolutional Neural Networks, Linguistic Resources, Bag of Words.

Abstract: User reviews submitted to app marketplaces contain information that falls into different categories, e.g., feature evaluation, feature request, and bug report. The information is valuable for developers to improve the quality of mobile applications. However, due to the large volume of reviews received every day, manual classification of user reviews into these categories is not feasible. Therefore, developing automatic classification methods using machine learning approaches is desirable. In this study, we compare the simplest textual machine learning classifier using only lexical features—the so-called Bag-of-Words (BoW) approach—with the more complex models used in previous works adopting rich linguistic features. We find that the performance of the simple BoW model is very competitive and has the advantage of not requiring any external linguistic tools to extract the features. Moreover, we experiment with deep learning based Convolutional Neural Network (CNN) models that have recently achieved state-of-the-art results in many classification tasks. We find that, on average the CNN models do not perform better than the simple BoW model—it is possible that for the CNN model to gain an advantage, a larger training set would have been necessary.

1 INTRODUCTION

App marketplaces such as PlayStore or AppStore offer apps to its users supporting virtually all kinds of services and businesses (Chen et al., 2014). These marketplaces provide to users a central place to download apps and submit their feedbacks on them in the form of ratings and reviews. The app market is highly competitive. Therefore, app developers constantly look for information that helps them improve the quality of their apps (Villarroel et al., 2016). User reviews contain information such as feature requests, bug reports, and feature evaluations, making them an extremely valuable source for app developers to improve the quality of their apps (Maalej and Nabil, 2015).

Developers receive a large number of reviews every day making manual classification of reviews an arduous task. In past research, supervised machine learning methods have been used for automatic classification of app reviews into different categories (Gu and Kim, 2015; Maalej and Nabil, 2015). Maalej et al. (2015) performed automatic classification at review level. However, multiple types of information can be mentioned in a single review or a review can contain information that is not informative for app developers. Therefore, other studies have performed automatic classification of reviews at sentence-level (Gu and Kim, 2015; Chen et al., 2014).

Gu et al. (2015) use natural language processing (NLP) tools, such as taggers and parsers, to extract features for classifying review sentences. However, the review-level classification results of Maalej et al. (2015) suggest that extracting such complex features might not be necessary and comparable classification results could be obtained by using only simple lexical Bag-of-Words (BoW) features. The BoW model, if its performance is on par with more complex feature sets, is an attractive approach for a non-expert because it does not require using any dedicated natural language processing tools. This perspective motivates us to find an answer to the following research question:

RQ1: When classifying app review sentences, how does a model with simple BoW features compare with the model using more complex linguistic features extracted via external NLP tools?

To answer RQ1, we use the dataset of Gu et al. (2015) and train a Maximum Entropy (MaxEnt) model using both feature sets: BoW features and the set of linguistic features proposed by Gu et al (2015).

Our results show that the simple BoW is very

Shah, F., Sirts, K. and Pfahl, D.
Simple App Review Classification with Only Lexical Features.
DOI: 10.5220/0006855901120119
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 112-119
ISBN: 978-989-758-320-9

competitive, both in terms of feature extraction and computational complexity, for review sentence classification.

Recently, deep learning based models have gained popularity among researchers as they have an ability to learn useful feature representations automatically from a large corpus of labeled data without manual feature engineering effort.

Specifically, a deep learning model known as *Convolutional Neural Network* (CNN) has recently achieved encouraging results for various text classification tasks (Kim, 2014). A recent study of Fu and Menzies (2017) suggest researchers to always compare computationally expensive models with their simple and efficient counterparts. For this objective, we are interested in comparing the powerful deep learning CNN model with the simple BoW model. We formulate the second research question (RQ2) as follows:

RQ2: How does the deep learning based CNN classifier compare with the simple BoW model for app review sentence classification?

To answer RQ2, we experiment with CNN-based models for review sentence classification. For that, we adopt the model proposed by Kim (Kim, 2014). A comparison of CNN model performance with MaxEnt model with BoW features shows that on average, the CNN-based model performs slightly worse than the BoW model. However, for the review sentence types *feature request* and *bug report*, which are some of the most informative sentence types to the developers, CNN-based models obtain the highest precision.

The rest of the paper is structured as follows. Section 2 summarizes the related work. In Section 3, we describe the dataset used for this study. In Section 4, we provide the description of the features and models used in this study.

Section 5 details the experimental setting. Section 6 discusses the results. In Section 7, threats to validity are examined. Conclusions are presented in Section 8.

2 RELATED WORK

The system "SUR-Miner" proposed classifying review sentences into *feature evaluation*, *praise*, *bug report*, *feature request*, and *other* (Gu and Kim, 2015). They used a MaxEnt model for the classification task with a rich set of lexical and structural features extracted with NLP tools. We adopt their feature set and compare it to the BoW model. However, our results are not directly comparable to theirs because they trained a separate model for each app while we train a single model incorporating sentences of all apps in the dataset, thus having a larger training set.

Table 1: Definition of five review sentence types used in the study of Gu and Kim (2015)

Sentence type	Definition	Examples
Praise	Expressing emotions with specific reasons	Excellent! I love it! Amazing!
Feature Evaluation	Expressing opinions about specific features	The UI is convenient. I like the prediction text.
Bug Report	Reporting bugs, glitches or problems	It always force closes when I click the ".com" button.
Feature Request	Suggestion or new feature requests	It's a pity it doesn't support Chinese.
Other	Other categories defined in (Pagano and Maalej, 2013)	I've been playing it for three years.

Maalej et al. (2015) experimented with different classification models to classify reviews into *feature request*, *bug report*, *rating*, and *user experience*. They experimented with various features, including BoW. However, they evaluated their models on review-level and not on sentence level as we do in this work. Similarly to us, they trained their models on the whole dataset of different apps.

Chen et al. 2014 proposed the system "AR-Miner" to help developers filter out informative reviews. Their system classifies review sentences into two classes: *informative* and *non-informative*.

The study of Panichella et al. (2015) assigned a different set of categories to reviews based on user intentions, i.e., *opinion asking*, *problem discovery*, *solution proposal*, *information seeking*, and *information giving*, and trained a learner to automatically classify reviews into those categories.

All these previous studies have used manual feature engineering for their classification models. According to our knowledge, this the first study that also experiments with features automatically learned with a deep neural network to classify app reviews. Moreover, none of the previous studies has established the BoW baseline for review sentence classification, which is one of the simplest feature sets that does not require any feature engineering or external tools, and which despite of its simplicity can be very effective.

3 DATASET AND PREPROCESSING

For this study, we used the app review dataset contributed by Gu and Kim (2015). The dataset contains labeled review sentences of 17 apps belonging to dif-

Table 2: App-wise distribution of sentence types in the dataset of (Gu and Kim, 2015).

App Name	App Category	Review types					
		Feature Evaluation	Feature Request	Bug Report	Praise	Other	Total
chase mobile	finance	372	152	120	304	1051	**1999**
duolingo	education	370	20	121	614	874	**1999**
swiftkey	productivity	385	98	177	463	876	**1999**
google playbook	books	254	152	198	413	982	**1999**
yelp	food	435	44	54	348	1118	**1999**
google map	map	354	273	141	312	919	**1999**
text plus	social	354	138	75	537	1013	**2117**
wechat	social network	231	132	71	612	953	**1999**
google calender	productivity	466	119	463	109	842	**1999**
spotify calender	music	231	87	90	714	877	**1999**
yahoo weather	weather	493	71	85	508	842	**1999**
temple run 2	game	234	48	17	877	877	**2053**
medscape	medical	464	82	83	522	848	**1999**
espn	sports	472	287	128	161	951	**1999**
camera360	photography	178	67	24	928	928	**2125**
imdb	entertainment	361	115	194	363	966	**1999**
kakotalk	communication	220	69	77	768	865	**1999**
Total		**5874**	**1954**	**2118**	**8553**	**15782**	**34281**

ferent app categories, such as games, communication, books, and music. Each review sentence is assigned a label from a set of mutually exclusive types, which are: a) *feature evaluation*, b) *praise*, c) *feature request*, d) *bug report*, and e) *other* (Gu and Kim, 2015). Table 1 presents the definition and a sample of review sentences for each type (Gu and Kim, 2015).

Table 2 shows the distribution of sentence types in each app category. It is apparent that the distribution of sentence types is highly skewed. The highest number of sentences belongs to the type *other* followed by *praise*. The numbers of other three sentence types—*feature evaluation*, *bug report* and *feature request*—are significantly smaller. However, these are the sentence types we are most interested in because they more likely contain useful information that help developers to improve their app.

The user review texts contain many typos and contractions that can make automatic classification of app review sentences a difficult task. To address this issue, we used a collection of 60 types and contractions[1] identified by Gu and Kim (2015) to correct the words in the dataset. During this cleaning process, we replaced the common typos and contractions, e.g. "U" is replaced with "you" and "Plz" is replaced with "Please" etc.

[1]https://guxd.github.io/srminer/appendix.html

4 CLASSIFICATION MODELS

This section describes the models designed to answer our research questions (RQ1 and RQ2). We describe in detail the textual features used to train MaxEnt models for review sentence classification. Then, we explain the CNN architecture that combines the automatic feature extraction and classifier to classify the same set of review sentences.

4.1 BoW Features

Bag-of-Words (BoW) is a very simple feature extraction method without much manual effort. In this approach, first a dictionary is created from all lexicons occurring in the training corpus. Then, a feature vector for each review sentence is created that stores the frequency of each lexical term in that sentence (Maalej and Nabil, 2015).

The lexical features are important in characterizing review sentence types. For instance, the words "awesome" and "great" are mostly used to praise the app. Similarly, the words "bug" and "crash" represent bug reports.

4.2 Linguistic Features

We extract the same set of linguistic features as was proposed by Gu et al. 2015.

Linguistic features can be useful because review sentences in each category often follow a distinct

structural pattern. For instance, for aspect evaluation, the sentence structure tends to have a pattern like "The search (noun) works pretty nice (adjective)" or "Its perfect (adjective) for storing notes (noun)". While for feature request, sentence structure often follows the patterns such as "please add look up feature" or "it could be improved by adding more themes".

Part of Speech (POS): POS tags indicate the type of each word in a sentence. For example, POS tags for the sentence "The user interface is elegant." are "Determiner Noun Noun Verb Adjective". We extracted the PTB POS tags[2] with NLTK[3] and used the concatenation of POS tags of all the words in a sentence as a feature.

Constituency Parse Tree: Constituency parse tree represents the grammatical structure of a sentence. Figure 1 shows the constituency parse tree for a sample review sentence generated using Stanford CoreNLP library.[4] The parse tree shows that the sentence (S) consists of a noun phrase (NP) and a verb phrase (VP). The VP is further decomposed into an adjective phrase (ADJP). The parse tree of a sentence is traversed in breadth first order and the first five nodes are stored. The concatenation of non-terminal labels of these five nodes is then used as a feature.

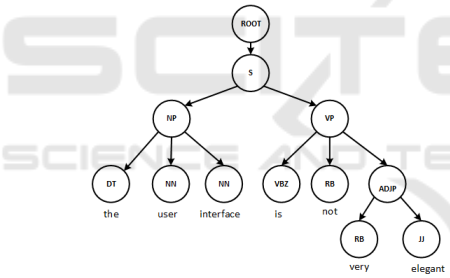

Figure 1: Constituency parse tree for a review sentence "the user interface is not very elegant". The feature extracted from this tree is "ROOT-S-NP-VP-DT-NN".

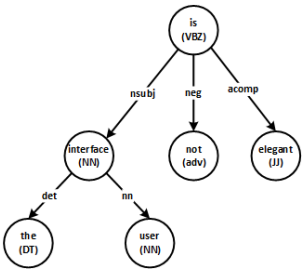

Figure 2: Semantic Dependence Graph of a sample review sentence "the user interface is not elegant". The feature extracted from this SGD is "VBZ-nsubj-NN-neg-ADV-acomp-JJ".

[2]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[3]http://www.nltk.org/

[4]https://stanfordnlp.github.io/CoreNLP/

Semantic Dependency Graph (SDG): SDG is a directed graph that shows the dependency relations between words in a sentence. Nodes in the graph represent words labeled with POS tags and edges represent dependency relations between words. Figure 2 shows the dependency graph of a sample sentence generated using spaCy[5] library. The word *is* is the ROOT node of the sentence as it does not have any incoming edges. The root has three dependents with the following relationships: a noun subject (nsubj) *interface*, a negation modifier (neg) *not*, and and adjectival complement (acomp) *elegant*. The child node *interface* has two children: a determiner (det) *the* and a noun compound modifier (nn) *user*. To extract the feature, the SDG is traversed in a breadth first order and the dependency relations labeling the edges and the POS tags of the words in the nodes are concatenated. Leaf nodes that are not directly connected to the ROOT node are ignored. For example, the textual feature extracted from SDG of a sentence shown in Figure 2 is "VBZ-nsubj-NN-neg-ADV-acomp-JJ".

Trunk Words: The trunk word feature is simply the root word of a SDG. For instance, the trunk word of the sentence "The user interface is not elegant" is *is*.

Character N-Grams: Finally, character N-grams, similar to BoW, are simple lexical features.

They have been used successfully in many applications such as malicious code detection and duplicate bug report detection (Gu and Kim, 2015). Character N-gram features of a sentence are all N-consecutive letter sequences (without spaces) in the tokens of the given sentence. For example, the 3-grams for the sentence "The UI is Ok" are *The*, *heU*, *eUI*, *UIi*, *Iis*, *isO*, and *sOk*. We use 2-4 grams as features in our classification model.

4.3 Convolutional Neural Networks (CNNs)

CNN-based classification models have shown encouraging results on various textual classification tasks (Kim, 2014; Collobert et al., 2011). We adopt the CNN architecture proposed by Kim (2014) to classify the review sentences.

The architecture of the model is illustrated in Figure 3. The first layer of the network embeds words into low dimensional vectors. The second layer performs convolutions over the embedded word vectors using multiple filter sizes. The output of these convolutions are max pooled into a long feature vector in the third layer. The fourth layer is a dense layer with dropout applied. Finally, the results are classi-

[5]https://spacy.io/

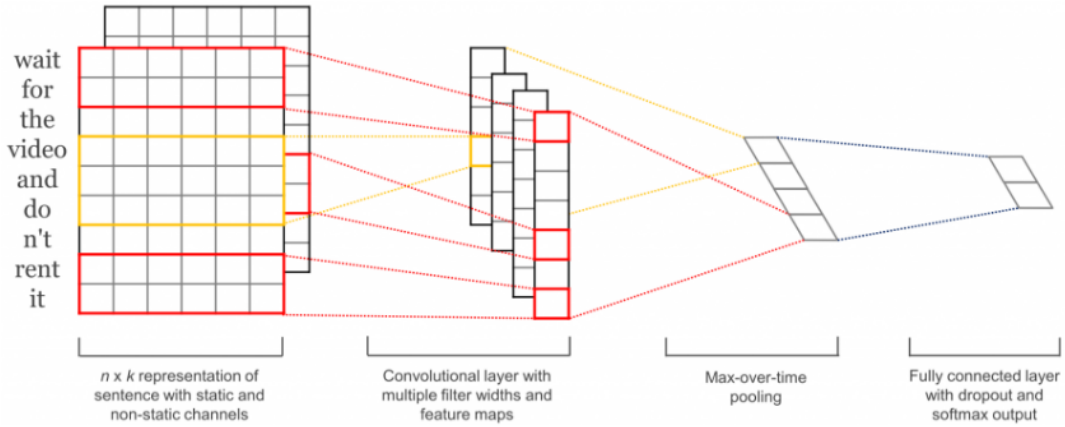

Figure 3: CNN model architecture for sentence classification (Figure taken from (Kim, 2014)).

fied using a softmax layer. For more details see (Kim, 2014).

Because neural network models have a large number of trainable parameters, they typically require large training sets to learn properly. However, when the available training sets are not that large, as is the case in this study, initializing CNN-based model with pre-trained word embedding vectors, obtained from a unsupervised neural language model might help to improve the model performance (Kim, 2014; Socher et al., 2011).

Therefore, we train CNN-models both with and without pre-trained word embeddings to assess the effect of using the externally trained word vectors for classifying app review sentences. We use the 300-dimensional Word2Vec embeddings (Mikolov et al., 2013) trained on 100 billion words from Google News.[6]

The words that are absent in the vocabulary of pre-trained embeddings are initialized randomly. In particular, we experiment with three different models:

- **CNN (rand):** The CNN model in which all word vectors in the embedding layer are randomly initialized and then modified during training.
- **CNN (static):** The CNN model is initialized with the pre-trained word vectors but all words including the ones that are randomly initialized are kept static and are not updated during training.
- **CNN (non-static):** Same as CNN (static) but the pre-trained vectors are fine-tuned during model training for our classification task.

[6]https://code.google.com/archive/p/word2vec/

5 EXPERIMENTAL SETUP

We train and test all models on the dataset described in Section 3. We compare all classification models on the test set by computing precision, recall, and f1-score for each review sentence type.

For all experiments, labeled review sentences of all apps were merged into one dataset (see Table 2). We trained 10 instances of each model to ensure that the impact on accuracy due to variation in the data has been taken into account.

For each training instance, 80% of the data was randomly sampled as training set and 20% as test set without fixing the seed value. During each run, a model was trained on the training set and evaluated on the test set. The prediction accuracy of the ten evaluations were averaged and reported as the final performance.

All the experiments were run on a CPU cluster (2 x Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz) with resources of one compute node and 16 GB RAM. We used the scikit-learn python library[7] to train, tune and evaluate the MaxEnt models. The regularization hyperparameter C was fine-tuned separately for both MaxEnt models by performing 5-fold cross-validation on 80% of the randomly sampled data. For the BoW model, the regularization weight was fixed to .856 and for the linguistic features model, C was fixed to .09.

For the CNN model, we used a freely available implementation of Kim (2014)[8] based on TensorFlow[9] library in python. The hyperparameters used in the CNN model are: rectified linear units (ReLU), filter windows of sizes 2, 3 and 4 with 128 feature maps

[7]http://scikit-learn.org/stable/
[8]https://github.com/dennybritz/cnn-text-classification-tf
[9]https://www.tensorflow.org/

for each filter. The dropout rate of 0.6 and L2 regularization parameter of 0.1 was chosen by performing 5-fold cross-validation on a training set. We used a batch size of 256 and trained the models for 50 epochs.

6 RESULTS AND DISCUSSION

This section presents the results regarding our research questions (RQ1 and RQ2). The classification accuracies of our models are shown in Table 3. The first two rows present the results of the MaxEnt models. The bottom three rows give the results of the three CNN-based models. The best result in each column is in bold, the best neural model result is underlined. We also give the average results of *feature evaluation*, *feature request* and *bug report* sentence types, as these categories are expected to give the most information about how improve the app.

In the following, we answer the research questions and discuss the results.

RQ1 concerns with the performance comparison of the two MaxEnt models (the first two rows in Table 3). The first model uses simple BoW features and the second model leverages linguistic resources (see Section 4.2). On average, the MaxEnt model with linguistic features achieves better performance than the MaxEnt model with BoW features but the difference is only .01 points for precision, and .02 points for both recall and f1-score. Both models demonstrate roughly the same performance for the class types *feature evaluation* and *bug report*. Only for the class type *feature request*, the model with linguistic features is clearly better than the BoW model. In relation to our RQ1, we conclude that the simple MaxEnt BoW model that does not require linguistic resources and is computationally the fastest (see Table 4), is almost as competitive as the MaxEnt model with complex linguistic features.

The RQ2 studies the performance of deep learning based CNN in comparison with the MaxEnt model with BoW features. On average, the neural model with best f1-score (non-static CNN) is worse than the model with BoW features but the difference is only less than 0.01. In terms of precision, the best neural model is the CNN with randomly initialized embeddings. For all three relevant sentence categories, this model obtains the best or close to best precision among all models at the cost of lower recall for *feature evaluation* and *feature request* sentence types. In terms of recall, the CNN (non-static) is the best, obtaining competitive performance for all three relevant sentence types.

Hence, we conclude with regards to our RQ2 that the CNN-based neural networks can achieve competitive performance in comparison to the MaxEnt BoW model.

However, as the best precision and recall were obtained with the different configuration of the CNN model (random vs non-static embeddings), the superiority of one or the other approach is not clear. It is possible that with a larger training set, the CNN model would gain a clear advantage over the simple MaxEnt BoW model.

Previous studies have shown that tuning the word vectors specific to the classification task (non-static CNN) improves model performance (Kim, 2014). Although the average F1-score as well as the recall of the CNN (non-static) model is the best among the neural models, the difference at least in F1-score is non-significant. On the other hand, the precision is the best for the model with randomly initialized embeddings (rand CNN) and the difference between the other two neural models is almost 5%. One possible reason for this can be that the textual domain of Google News is too different from the texts of app reviews and thus embeddings trained on that will not give a good starting point for our model. It is possible that word embeddings pre-trained on a large amount of app reviews would perform better in our case.

Training neural network models is generally computationally more costly (see Table 4) than training MaxEnt models due to the larger number of trainable parameters. Still, the fact that they do not require external linguistic tools (parsers, taggers etc) to extract features and they can be trained offline, might make the neural models an attractive alternative in case they display superior performance over simpler models. The CNN-based models are expected to perform better than the traditional machine learning models, i.e, MaxEnt, when large amounts of labeled data is provided. However, this is rarely the case in the software engineering community. Moreover, such models also require specialized knowledge and expertise to use them. Therefore, researchers who are not expert in deep learning nor have the knowledge of using NLP tools can safely use the simple BoW model for classification of app review sentences, which yields results very close to the more complex models.

7 THREATS TO VALIDITY

The review dataset used in this study is collected from PlayStore and manually labeled by Gu et al. (2015). We do not know the extent to which the results of our study are sensitive to the annotators and annotation

Table 3: Performance of classification models for different types of review sentences. We also show the mean precision, recall and f1-score averaged over the most relevant sentence types: *feature evaluation*, *feature request* and *bug report*. The best result in each column is in bold, the best CNN model result is underlined.

Model	Feature Evaluation			Feature Request			Bug Report			Average			Praise			Other		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
MaxEnt (BoW)	**.783**	.633	.700	.715	.589	.646	.736	.562	.637	.745	.595	.661	.835	.853	.844	.775	**.863**	.817
MaxEnt (L+S)	.767	.645	.700	.766	**.619**	**.684**	.733	**.588**	**.652**	.755	.617	**.679**	.851	**.873**	**.861**	.784	.860	.820
CNN (rand)	.771	.613	.681	**.781**	.492	.602	**.755**	.569	.646	**.769**	.558	.643	**.883**	.816	.781	**.861**	.847	**.854**
CNN (static)	.738	.638	.685	.722	.522	.604	.698	.547	.612	.719	.569	.634	.751	.864	.802	.781	.763	.770
CNN (non-static)	.726	**.679**	**.702**	.679	.607	.640	.655	.584	.617	.687	**.623**	.653	.788	.822	.804	.842	.855	.848

Table 4: Runtime of different classification models.

Model	Runtime for one run
MaxEnt (BoW)	9 mins
MaxEnt (L+S)	22 mins
CNN (rand)	252 mins
CNN (non-static)	376 mins
CNN (static)	554 mins

guidelines used to label this data. Moreover, the nature or language characteristics of the reviews in other app marketplaces may be different to that of PlayStore. Therefore, we do not claim the generalizability of our results to reviews from other platforms like, e.g., AppStore.

The CNN-based model has a large number of hyperparameters that can be tuned to potentially improve the performance. This set of hyperparameters includes the size of the embeddings, number and sizes of filters, the choice of the optimizer with its parameters, various options for regularization, etc. Tuning all these hypermarameters is infeasible in practice. Thus, we tuned the drop-out rate and the strength of the L2-regularization. Still, tuning other hyperparameters as well might improve the model performance. The number of examples for each sentence type in the dataset are imbalanced. To tackle this imbalance, we experimented with random oversampling and random undersampling techniques in MaxEnt models but did not observe any improvements in F1-score. Many other techniques exist to handle class imbalance and thus it is possible that using one of those would have made a difference. Also, we did not apply the class balancing techniques to neural models where they potentially could have improved the results.

8 CONCLUSION

We explored the power of simple lexical features in classifying app review sentences. For that, we compared the simple Bag-of-Words feature representation with a more complex feature set proposed in previous work extracted using various NLP tools. We found that on average, the simple BoW model performs almost as well as the model with complex linguistic features. Considering that software developers and software engineering researchers are typically not experts in NLP tools, this is a desirable result. We also experimented with deep learning based CNN models which have become very popular due to their ability to learn complex feature representations from simple lexical inputs as well as their good performance in many tasks. In our study, we did not observe any advantage of using computationally more expensive CNN over its simpler BoW counterpart. Thus, we conclude that the simple lexical BoW model is very competitive and offers a simple method even to the non-experts, both in terms of feature extraction and computational complexity, for review sentence classification.

ACKNOWLEDGMENTS

We are grateful to Xiaodong Gu for sharing the review dataset for this study. This research was supported by the institutional research grant IUT20-55 of the Estonian Research Council.

REFERENCES

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., and Zhang, B. (2014). AR-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the ICSE 2014*, pages 767–778. ACM Press.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Fu, W. and Menzies, T. (2017). Easy over hard: A case study on deep learning. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, pages 49–60, New York, NY, USA. ACM.

Gu, X. and Kim, S. (2015). "what parts of your apps are loved by users?". In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 760–770.

Johann, T., Stanik, C., B., A. M. A., and Maalej, W. (2017). SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 21–30. IEEE.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the EMNLP 2014*, pages 1746–1751. ACL.

Liu, P., Joty, S., and Meng, H. (2015). Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

Maalej, W. and Nabil, H. (2015). Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proceedings of RE 2015*, pages 116–125. IEEE.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pagano, D. and Maalej, W. (2013). User feedback in the appstore: An empirical study. In *Proceedings of RE 2013*, pages 125–134.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., and Gall, H. C. (2015). How can i improve my app? classifying user reviews for software maintenance and evolution. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, ICSME '15, pages 281–290, Washington, DC, USA. IEEE Computer Society.

Shah, F. A., Sabanin, Y., and Pfahl, D. (2016). Feature-based evaluation of competing apps. In *Proceedings of the International Workshop on App Market Analytics - WAMA 2016*, pages 15–21, New York, New York, USA. ACM Press.

Socher, R., Lin, C. C.-Y., Ng, A. Y., and Manning, C. D. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 129–136, USA. Omnipress.

Villarroel, L., Bavota, G., Russo, B., Oliveto, R., and Di Penta, M. (2016). Release planning of mobile apps based on user reviews. In *Proceedings of the ICSE 2016*, pages 14–24. ACM.