From Bag-of-Words to Pre-trained Neural Language Models: Improving Automatic Classification of App Reviews for Requirements Engineering

Adailton F. Araujo, Marcos P. S. Gôlo, Breno M. F. Viana, Felipe P. Sanches, Roseli A. F. Romero, Ricardo M. Marcacini

Institute of Mathematics and Computer Sciences - University of São Paulo (USP)
PO Box 668 – 13.560-970 – São Carlos – SP – Brazil

{adailton.araujo,marcosgolo,brenov,fpadula}@usp.br

{ricardo.marcacini,rafrance}@icmc.usp.br

Abstract. *Popular mobile applications receive millions of user reviews. These reviews contain relevant information, such as problem reports and improvement suggestions. The reviews information is a valuable knowledge source for software requirements engineering since the analysis of the reviews feedback helps to make strategic decisions in order to improve the app quality. However, due to the large volume of texts, the manual extraction of the relevant information is an impracticable task. In this paper, we investigate and compare textual representation models for app reviews classification. We discuss different aspects and approaches for the reviews representation, analyzing from the classic Bag-of-Words models to the most recent state-of-the-art Pre-trained Neural Language models. Our findings show that the classic Bag-of-Words model, combined with a careful analysis of text pre-processing techniques, is still a competitive model. However, pre-trained neural language models showed to be more advantageous since it obtains good classification performance, provides significant dimensionality reduction, and deals more adequately with semantic proximity between the reviews' texts, especially the multilingual neural language models.*

1. Introduction

Mobile applications (app) users can evaluate their usage experience by providing a numerical score and textual comments. Such text reviews describe bug reports, new features requests, or just the evaluation of app specific features. In fact, app reviews are a valuable knowledge source for requirements engineering and software development innovation. This knowledge has been explored in several tasks, such as feedback-driven improvements to increase the user engagement [Pagano and Maalej 2013], strategic decisions related to development, validation and product improvement [Maalej et al. 2016] and, consequently, obtain a competitive advantage in the market [Shah et al. 2019].

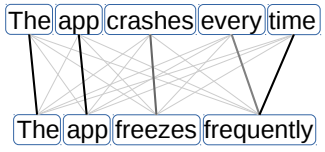
Popular apps usually receive hundreds of thousands or even millions of reviews and the manual analysis of so many reviews is an impracticable task. Several machine learning-based solutions have been proposed to solve this problem [Pagano and Maalej 2013, Guzman et al. 2015, Maalej et al. 2016, Wang et al. 2018, Al Kilani et al. 2019, Messaoud et al. 2019, Dabrowski et al. 2020], in which relevant information is extracted from the texts to automatically classify a review.

An app review classifier can be defined as a function $f : X \rightarrow Y$ that maps a review $x \in X$ to a set of c classes $Y = \{1, ..., c\}$, where $y \in Y$ usually indicates useful labels for requirements engineering, such as "bug", "app feature", "user experience", "opinion / rating", among others. In this case, $X \in \mathbb{R}^d$ represents the reviews feature space. Supervised machine learning classification aims to learn a function f^* from a training set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ that approximates the unknown mapping function f. A well-known issue is that the learning process is directly influenced by the feature space, i.e., by the technique used to represent the texts (unstructured data) in a vector space model. Considering the various techniques for text pre-processing, from traditional Bag-of-Words (BoW) models to the most recent Neural Language Models (NLM), we investigate the following research question: *what is the impact of different text representation models for app reviews classification in the context of software requirements engineering*?

Most existing studies use the traditional Bag-of-Words model for textual representation, where word histograms are used as the features in app reviews classification [Aggarwal 2018]. Although BoW is simple and intuitive, this model has limitations that hinder classification. First, BoW assumes independence between words, so different sentences can have the same representation. Even when we use Bag-of-Ngrams to consider word order, the classifier usually suffers from high dimensionality and sparse representation. Second, there is a lack of semantics, in which similar (or semantically related) words or sentences are considered to be distinct features.

Neural language models are currently state-of-the-art for many tasks in natural language processing (NLP) [Mulder et al. 2015, Belinkov and Glass 2019]. The first word embeddings models, such as Word2vec [Mikolov et al. 2013], were promising for computing proximity between words by using word vectors, but without solving the word order drawback. More recently, neural language models based on transformer architectures and attention mechanisms enabled a rich textual representation learning [Otter et al. 2020], allowing to identify semantic and syntactic relations, as well as to consider the sequential structure of the texts. Besides, these models are pre-trained in large textual corpus, thereby enabling their use for new tasks with smaller training corpus [Zhou et al. 2020]. Figure 1 presents a simple example of how the similarity between two sentences can be obtained in pre-trained language models, in which the darker lines indicate greater semantic proximity. Note that a traditional BoW model fails to determine the proximity between these sentences.

Figure 1. Example of the similarity between two sentences in pre-trained NLM.



In this paper, we investigate and compare Bag-of-Words models and pre-trained neural language models for app reviews classification. We analyzed the main term weighting strategies for Bag-of-Words models, such as binary weighting, term frequency (TF) and TFIDF, as well as the use of unigrams and bigrams. Regarding pre-trained neural language models, we analyzed three state-of-the-art deep neu-

ral networks: BERT [Devlin et al. 2018], DistilBERT [Sanh et al. 2019] and RoBERTa [Liu et al. 2019]. BERT is a representation model that achieved promising results in the General Language Understanding Evaluation (GLUE) benchmark, both for sentence-level and token-level tasks. DistilBERT is a smaller (distillate) general-purpose language representation model, which obtains competitive results with lower computational costs. RoBERTa is an optimized (robust) version of BERT, in which the authors investigated new strategies for training the deep neural network. All textual representation models were evaluated in classification tasks, in which app reviews are mapped into requirements engineering classes. To mitigate the classification method's bias, we evaluated the representation models considering four different classifiers: Support Vector Machines (SVM), k-Neighbors Nearest (kNN), Multilayer Perceptron Neural Networks (MPNN), and Multinomial Naive Bayes (MNB).

We carried out an experimental evaluation on a real-world dataset containing 3691 app reviews organized into four requirements engineering classes. Experimental results show that app reviews classification using pre-trained language models obtains higher F1-Measure values than BoW-based classification. However, our analysis found no statistically significant differences between these textual representation models when considering the various classification methods. We discuss practical advantages in favor of pre-trained neural language models for app reviews classification. While BoW models require a careful feature selection, stopwords removal and term weighting, the pre-trained neural language models can be used as a ready-to-use module that achieves a reduced, high-quality vector embedding space directly from texts. Moreover, we also show that semantic similarities from neural language models combined with explainable AI tools provide interesting insights involving the extraction of important words from app reviews and their relationship with requirements engineering classes.

2. App Review Classification

App developers understand that the user's opinion about their apps' usage is important to the app's development and maintenance to meet the users' expectations [Pagano and Maalej 2013]. This opinion, called *app review*, is an important communication channel between developers and app users [Messaoud et al. 2019].

An app review can be classified into different classes related to requirements engineering, such as a **fault report** describing a found problem in the software which must to be fixed, e.g. "*The order history does not show my last order*"; a wish of a **new feature**, e.g. "*It would be wonderful if I could be able to pay with Bitcoin*" or "Too bad it isn't available of IOS platform"; a description of the **usage experience of a specific feature**, e.g. a positive experience "*It's sensational I can see my order location in real time*" or a negative experience "*Impossible to find a good restaurant with this awful search*", as well as other types of **ratings** in general. In this context, it may become strategic for app developers to analyze each review aiming to identify the app review class and plan actions for each situation. These actions may involve: to fix a critical fault that has affected a wide number of users; to design a new feature or module for users satisfaction improvement; to prioritize opportunities and urgent demands from the perspective of a wide number of users [Al Kilani et al. 2019].

Several works have been developed in the last years aiming to automatize this

app review classification [Maalej et al. 2016, Wang et al. 2018, Aralikatte et al. 2018, Messaoud et al. 2019, Al Kilani et al. 2019]. This automation aims to discard the irrelevant reviews and to group truly useful reviews. Supervised Machine Learning algorithms have been used to perform the automatic review classification where the important review groups are mapped for the target-classes. First, human experts must manually analyze each review to assign one of the target-class (e.g. *Rating*, *User Experience*, *Bug* e *Feature*) or discard the review if it does not fit any class. Second, machine learning algorithms are trained from the labeled reviews. Thus, the obtained classification model are able to automatically assign a target-class to new reviews.

[Maalej et al. 2016] created a dataset with 3.691 app user reviews that were manually classified by human experts into four classes: Rating, User Experience, Bug, or Feature. In the pre-processing stage, authors used the Bag-of-Words (BoW) model for text representation, as well as stopwords removal, stemming, bigram generation. For app review classification, they used the Naive Bayes, Decision Tree, and MaxEnt algorithms. In our work, as well as other works found in the literature [Wang et al. 2018, Aralikatte et al. 2018, Messaoud et al. 2019, Al Kilani et al. 2019], we observed the use of just traditional BoW models for text representation. Thereby, we also used the dataset created by [Maalej et al. 2016] to compare the performance of BoW models with pre-trained neural language models.

3. Textual Representation

Machine learning algorithms for text classification require a structured data representation. In the natural language context, it is necessary to apply textual data pre-processing to build a structured representation of these data. One of the main ways to represent textual data is the traditional frequency-based representation, such as the Bag-of-Words model. More recently, pre-trained neural language models have become popular due to the competitive performance in text classification for different domains. Thereby, this section describes the derivations of frequency-based representations and three neural language models: BERT, RoBERTa, and DistilBERT.

3.1. Frequency-Based Representations

In the vector space model (VSM), each textual document is represented by a vector and each vector dimension by a characteristic (attribute) of the textual document. Table 1 illustrates a matrix representation of the vector space model for a text collection of m documents and n attributes [Tan et al. 2013]. The set of documents is denoted by $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ and the set of attributes by $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$. The matrix cells, or vector dimensions, quantify the attribute occurrence in a document.

Table 1. Matrix representation of VSM (m documents and n attributes).

	$\mathbf{t}_1$	$\mathbf{t}_2$	$\mathbf{t}_3$	...	$\mathbf{t}_n$
$\mathbf{d}_1$	w_{d_1,t_1}	w_{d_1,t_2}	w_{d_1,t_3}	...	w_{d_1,t_n}
$\mathbf{d}_2$	w_{d_2,t_1}	w_{d_2,t_2}	w_{d_2,t_3}	...	w_{d_2,t_n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{d}_m$	w_{d_m,t_1}	w_{d_m,t_2}	w_{d_m,t_3}	...	w_{d_m,t_n}

If the vector dimensions represent simple terms, the representation is called bag-of-words [Aggarwal 2018]. The terms weights of BoW may correspond to: (1) Bi-

nary that indicates just if there is or not occurrence of the term in the document; (2) Term Frequency (TF) that defines an occurrence frequency of a term in a document; Term Frequency - Inverse Document Frequency (TFIDF) that ponders the TF with the inverse of all documents frequency. Equation 1 defines the TFIDF term weighting,

$$TFIDF = TF_{d,t} \cdot log\left(\frac{|D|}{df_t}\right) \quad (1)$$

where $TF_{d,t}$ corresponds to the frequency of the term t in the document d, $|D|$ to the number of documents and df_t to the number of documents that contain the term t.

The traditional representation defined by BoW model may be considered as a set of unigrams, i.e., a set of unique words. Bigrams, in turn, are all the combinations of two terms of a set of terms of documents. Bigrams cause a significant increase in the number of attributes and can impair the classification process due to the so-called curse of high dimensionality. In fact, the pre-processing step using BoW models requires careful analysis of the textual dataset and the correct choice of techniques for attribute selection, stopword removal and term weighting.

3.2. Language Model-Based Representations

Language modeling aims to obtain a function to predict a word (or sentence) given previous words (or sentences) [Mikolov et al. 2013]. Recent methods use the idea of distributed representations, in which texts (e.g. words and/or sentences) are represented by low-dimensional (real-valued) vectors [Mulder et al. 2015]. Thus, texts can be compared through the correlation or distances between these vectors, which capture syntactic and semantic relationships. Learning good distributed representations is a challenge and neural networks are currently used for this task, also called Neural Language Models (NLM) [Otter et al. 2020].

NLM may be context-free or context-dependent. Context-free NLMs, such as word2vec [Mikolov et al. 2013], transform each text word into a vector of characteristics which is independent of the context in which the documents are being analyzed (e.g. word order). Context-dependent NLMs, on the other hand, group a vector of characteristics for each sentence or document, rather than each word, and also consider the context in which the words are written [Devlin et al. 2018]. Context-dependent NLMs perform better than the traditional language models [Devlin et al. 2018, Otter et al. 2020] and were used in our study involving app reviews classification.

NLMs may be undirected or bidirected. Undirected models analyze the text in a single way, like the reading way, for example. Bidirected models, on the other hand, analyze the text in both ways, from left to right and vice-versa. Bidirected NLMs provide a better performance than undirected NLMs [Devlin et al. 2018, Liu et al. 2019, Sanh et al. 2019]. Examples of bidirected models are: BERT [Devlin et al. 2018], RoBERTa [Liu et al. 2019] and DistilBERT [Sanh et al. 2019]. BERT-based models use a masking strategy for training, in which some words in a sentence are replaced by a special token called [MASK]. The model is trained to predict the masked words, using the context of non-masked words and their word sequence. In this case, the model uses pairs of sentences as input, in which it aims to predict whether the second sentence is subsequent to the first sentence. During the training stage, BERT-based models use an attention mechanism that learns contextual relations between words [Vaswani et al. 2017].

The RoBERTa, DistilBERT and Multilingual DistilBERT models are derivations of the BERT model. All of these derivations were proposed aiming to improve the results and to decrease the computational cost of the BERT model [Devlin et al. 2018]. RoBERTa model uses greater sequences of textual data than original model [Liu et al. 2019]. In addition, this model also removes the next sentence prediction goal and changes dynamically the masking pattern [Liu et al. 2019]. The DistilBERT model, on the other hand, differs of BERT in the pre-training stage by using a smaller number of parameters, a knowledge distillation technique (training of a compacted model) and the triple loss technique [Sanh et al. 2019]. This strategy grants the DistilBERT model to save 40% of memory and become 60% faster than BERT, besides preserving 97% of understanding resources of idioms [Sanh et al. 2019]. The Multilingual DistilBERT is a DistilBERT model that considers different idioms.

4. Experimental Evaluation

We used the textual collection created by [Maalej et al. 2016] for experimental evaluation. This collection contains the characteristics shown in Table 2.

Table 2. Textual Collection Characteristics.

Class Name	Number of Reviews	Average Review Tokens	Total Tokens
Bug	370	19	7220
User Experience	607	17	10303
Rating	2462	7	18421
Feature	252	18	4564

In these reviews, the users detailed their experience of mobile applications usage. The dataset contains 3691 app reviews, in English language, organized into four classes. The class Rating presents the user explanation for a given score. The class User Experience refers to the experience of using a specific feature. The class Bug describes a found problem in the app which must be fixed. The class Feature describes a wish or request for a new feature. Besides the free text and the class, each instance of the collection contains metadata information, such as the review date and score. To generate this app review dataset, the authors followed a strict protocol of annotation, involving 10 annotators and criteria for selecting a subset of a total of 1.3 thousand of reviews, of several categories, gathered from the Apple Store and Google Play Store.

4.1. Experimental Setup

Our work applies BoW models and pre-trained NLM models for textual representation. In BoW model, we used three term weighting techniques: TF, TFIDF and Binary. We also considered unigrams and bigrams versions of each one of these term weighting[1]. In BoW models, we applied a text cleaning process to decrease the data dimensionality, as well as to increase representation quality. This process, according to Aggarwal [Aggarwal 2018], improves the quality of the classification algorithms. The cleaning steps were: (1) converting words to lowercase and removal of accents; (2) removal of punctuation marks and alphanumeric characters; (3) removal of stopwords; and (4) word stemming [Aggarwal 2018].

[1]We generated the BoW model with bigrams by using the bigram generator of the scikit-learn library.

In the pre-trained models, we do not use any text cleaning techniques to maintain the original text structure, which is important for context-dependent NLMs. We used four pre-trained neural language models: BERT, RoBERTa, DistilBERT, and a multilingual version of DistilBERT (M. DistilBERT). Details on training parameters and textual corpus used to obtain pre-trained models are available in the work of [Reimers and Gurevych 2019]. Table 3 presents an overview of the dimensionality (number of features) obtained in each textual representation.

Table 3. Overview of the dimensionality of each textual representation.

	BoW	BoW-bigram	BERT	RoBERTa	DistilBERT	M. DistilBERT
Number of Features	4035	26962	1024	1024	768	512

We used four traditional classification algorithms: k-Nearest Neighbors (kNN), Multinomial Naive Bayes (MNB), Support Vector Machines (SVM) and Multilayer Perceptron (MLP). We also considered two Fine-Tuning algorithms, which are Deep Learning approaches, for BERT and DistilBERT pre-processing. In the fine-tuning strategy we added new layers to the pre-trained models to consider the task target, i.e., the app reviews classification. Thus, the distributed representations of the texts are fine-tuned to consider specific relations of the classification task. The parameters of the ML algorithms we used in our experiments were:

- **kNN**: $k \in [1, 30]$ and cosine metric;
- **MNB**: None;
- **SVM**: kernels $= \{RBF, sigmoid, polinomial, linear\}$ and C $= \{1.0\}$;
- **MLP**: Layered architecture $= \{1, 3, 6\}$, neurons $= \{50, 100, 150\}$, moment $= \{0.9\}$ and learning rate $= \{0.001\}$.
- **Fine-Tuning**: training epochs $= \{1, 5, 10, 15\}$, learning rate $= \{0.00002\}$, max token length $= \{100\}$ and bach size $= \{32\}$.

For reproducibility purposes, we provide a repository[2] with the source code of the classification and fine-tuning algorithms, as well as the textual representations obtained.

We evaluated the Multi-Class ML algorithms performance by using the 10-Fold Cross-Validation strategy [Tan et al. 2013]. We use the F_1 evaluation measure that corresponds to the harmonic mean of Precision (Equation 2) and Recall (Equation 3), where TP (True Positive) refers to the number of documents of a class in which the algorithm correctly classified; FP (False Positive) refers to the number of documents that do not belongs of a class in which the algorithm wrongly classified as belonging; and FN (False Negative) refers to the number of documents of a class in which the algorithm wrongly classified as another class. Equation 4 defines the F_1 measure.

$$P = \frac{TP}{TP + FP}, \quad (2) \qquad R = \frac{TP}{TP + FN} \quad (3) \qquad F_1 = \frac{2 \times P \times R}{P + R}, \quad (4)$$

4.2. Results and Discussion

We conducted an experimental evaluation to investigate two aspects involved in the app reviews classification. In the first aspect, our objective is to analyze the impact of each

[2] https://github.com/adailtonaraujo/classify_app_review

textual representation model considering the four different classification algorithms. In the second aspect, we evaluate the impact of fine-tuning a neural language model for a specific app reviews classification task.

Concerning the first aspect, Table 4 presents the classification results of the algorithms kNN, MNB, SVM and MLP. The cell values correspond to the average of macro F_1 measure of each class of the textual collection. A row of the table represents the F_1 measure result for a certain algorithm. For each row, we highlight in blue and green the greatest values of the BoW e NLM models, respectively.

Table 4. Comparison (macro F_1 measure) of bag-of-words models and pre-trained neural language models in different classifiers.

	Bag-of-Words						Neural Language Models			
	Unigram			Bigram						
	TF	TFIDF	Binary	TF	TFIDF	Binary	BERT	DistilBERT	RoBERTa	M. DistilBERT
SVM	**0.38**	0.35	0.36	0.34	0.28	0.34	0.47	0.47	0.46	**0.50**
MLP	0.37	0.34	0.37	**0.38**	0.31	0.38	0.47	0.46	0.47	**0.50**
MNB	0.41	**0.42**	0.41	0.39	0.38	0.39	0.53	0.51	**0.54**	0.53
KNN	0.29	0.28	0.26	**0.30**	0.26	0.26	0.47	0.46	0.46	**0.48**

Regarding the classification algorithms, we observe that the MNB obtained the greatest values of the F_1 for all text representation models. We also note that the kNN algorithm usually obtains the lowest F_1 values for the most text representation models.

Concerning the BoW-based models, we observe that the bigrams deteriorate the classification performance when used in conjunction with Binary and TFIDF term weighting techniques. For TF term weighting, on the other hand, bigrams improved the performance of the MLP and kNN algorithms. We also notice that, for unigrams, the TFIDF term weighting obtained the best F_1 value using MNB classifier.

Regarding the pre-trained NLM models, the Multilingual DistillBERT obtained the greatest values for the most ML algorithms. The only exception was the MNB algorithm in which the greater F_1 value was obtained by the RoBERTa model. We also identified that the results of the language model-based techniques were always greater than BoW-based for all the classification algorithms.

Figure 2 presents a graphical comparison of the BoW and NLM models based on Friedman's non-parametric test with Nemenyi post-test through a critical (statistical) difference diagram [García et al. 2010]. The diagram presents the average F_1 ranking of each model, and those which are connected by lines did not present statistically significant differences to each other. We observe that the only statistical difference is the one presented between three NLMs and one BoW: BERT, RoBERTa, Multilingual DistilBERT and TFIDF-bigram. Although there is no critical difference, we note that the NLMs present better average ranking than the BoW-based approaches. Regarding classification runtime, the NLMs present better performances than the traditional approaches due to the reduced dimensionality of the representation.

Figure 3 presents the best F_1 results of the BoW and NLM models for each classification algorithm, regardless of BoW's pre-processing technique and pre-trained language model choice. We observe that the NLM models obtained the best F_1 values in all classification algorithms.

Figure 2. Diagram of critical difference of Friedman test with Nemenyi post-test.

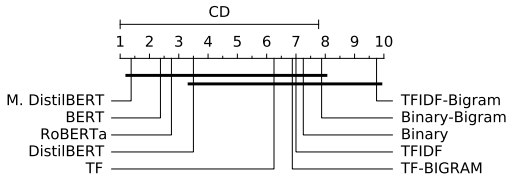
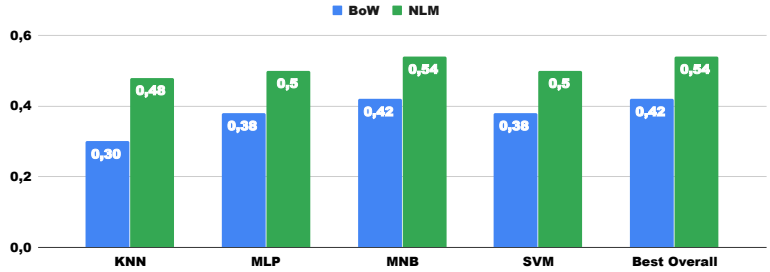


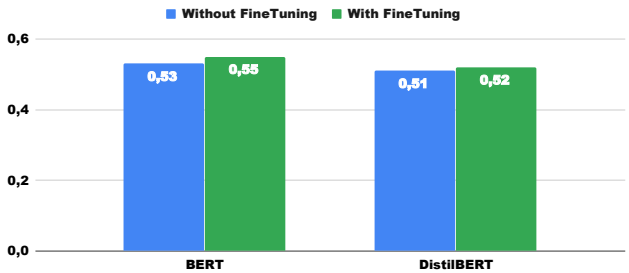
Figure 3. Comparison of the best F_1 values for BoW and NLM.



Besides of obtaining better F_1 values, the NLM models present other advantages. The first one is the absence of text tokenization and feature selection. Since the NLM models were pre-trained on a large textual corpus, such representations are relatively robust in relation to the writing errors commonly found in reviews. Furthermore, some NLM models are able to deal with several idioms written in a single textual document, e.g. Multilingual DistilBERT.

Concerning the second aspect of our experimental evaluation, we also consider the Fine-Tuning step in the BERT and DistilBERT models. Figure 4 shows a comparison of F1 results. For the classification results without Fine-Tuning, we chose the best results of both that were obtained by the MNB algorithm. We observe that the models that use the Fine-Tuning step obtained better F_1 results. However, considering the small improvement in classifier performance and the higher runtime and computational costs of the fine-tuning step, we argue that this strategy should only be used for large training sets.

Figure 4. Comparison of NLM approaches with and without Fine-Tuning.



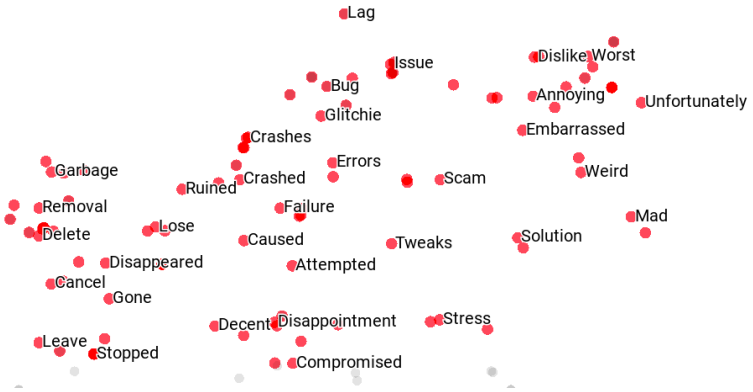
Although the fine-tuning strategy has not improved significantly in F_1 measure, it is useful to identify patterns (words and expressions) more related to the target classes. For example, we can highlight text excerpts that most influence the classifier's decision by using the LIME method — a method that infers the relative importance of attributes by using an interpretable linear model to generate explainings of classification predic-

tions [Ribeiro et al. 2016]. Figure 5 presents an example of a review of the class Feature extracted from our review dataset. The excerpts highlighted in green influenced more the classification, where the words "missing", "add" and "should" are related to the texts that express the user's desire for new app features. On the other hand, in red, the irrelevant excerpts for the classification, i.e., general words or less related to the context of the class, as "think", "you" e "app". Furthermore, the fine-tuning also improved the similarity relation between words. To illustrate this behavior, Figure 6 shows a two-dimensional projection of words that are semantically similar to the word "Error" (a relevant word for the class Bug). We note that the fine-tuned NLM models identified related words, which is potentially useful for exploratory analysis tasks.

Figure 5. Inference of important words for the class Feature (green and red for the most and less important words, respectively).

This is a very useful and helpful app In my opinion, the only thing it's missing is video handling You can add pictures and audio to your notes, and I think you should be able to add videos, too

Figure 6. Two-dimensional projection of semantically similar words.



5. Conclusion

We compared BoW-based models and pre-trained NLM models for textual representation. In the experimental evaluation, we used a textual collection of mobile app reviews and classification algorithms of different paradigms. The NLM models obtained greater classification F_1 values than BoW models. The Multilingual DistilBERT presented the greatest F_1 means for most classification algorithms. We consider this pre-trained model to be a reasonable choice for classifications tasks of app reviews, because, in addition to competitive F_1 results, it also supports multiple languages and uses less computational resources (it is a distilled version of BERT).

We also evaluated the fine-tuning usage in the BERT and DistilBERT models. The fine-tuning step presented an average improvement of 0.01 in the F_1 measure. This improvement may not be enough to justify fine-tuning due the runtime and high use of computational resources, especially for small datasets. However, it is important to emphasize that fine-tuning is potentially useful for exploratory analysis and interpretation of classification models, as well as the analysis of semantic relations between words.

The directions for future work involve investigating the impact of different fine-tuning strategies for app reviews classification. We intend to explore unsupervised fine-tuning methods that allow the use of large collections of unlabeled reviews, as well as cross-domain transfer learning learning strategies [Marcacini et al. 2018]. Thus, we intend to provide pre-trained BERT-based language models for text reviews that will be useful for various tasks related to the opinion mining and sentiment analysis. In addition, it is intended to analyze all the described pre-processing techniques with more than one new textual collections in order to make a better and richer evaluation of these techniques.

Acknowledgements

This work was supported by National Council for Scientific and Technological Development (CNPq) [process number 426663/2018-7], CAPES and FAPESP.

References

Aggarwal, C. C. (2018). *Machine Learning for Text*. Springer Publishing Company, Incorporated, 1st edition.

Al Kilani, N., Tailakh, R., and Hanani, A. (2019). Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 541–548.

Aralikatte, R., Sridhara, G., Gantayat, N., and Mani, S. (2018). Fault in your stars: an analysis of android app reviews. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 57–66.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Dabrowski, J., Letier, E., Perini, A., and Susi, A. (2020). Mining user opinions to support requirement engineering: An empirical study. In *Advanced Information Systems Engineering*, pages 401–416, Cham. Springer International Publishing.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064.

Guzman, E., El-Haliby, M., and Bruegge, B. (2015). Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 771–776.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Maalej, W., Kurtanović, Z., Nabil, H., and Stanik, C. (2016). On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331.

Maalej, W., Nayebi, M., Johann, T., and Ruhe, G. (2016). Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54.

Marcacini, R. M., Rossi, R. G., Matsuno, I. P., and Rezende, S. O. (2018). Cross-domain aspect extraction for sentiment analysis: A transductive learning approach. *Decision Support Systems*, 114:70–80.

Messaoud, M. B., Jenhani, I., Jemaa, N. B., and Mkaouer, M. W. (2019). A multi-label active learning approach for mobile app user review classification. In *International Conference on Knowledge Science, Engineering and Management*, pages 805–816.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mulder, W., Bethard, S., and Moens, M.-F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98.

Otter, D. W., Medina, J. R., and Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*.

Pagano, D. and Maalej, W. (2013). User feedback in the appstore: An empirical study. In *IEEE International Requirements Engineering Conference (RE)*, pages 125–134.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Shah, F. A., Sirts, K., and Pfahl, D. (2019). Using app reviews for competitive analysis: Tool support. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics*, WAMA 2019, pages 40–46, New York, NY, USA. ACM.

Tan, P., Steinbach, M., and Kumar, V. (2013). *Introduction to Data Mining: Pearson New International Edition*. Pearson Education Limited.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wang, C., Zhang, F., Liang, P., Daneva, M., and van Sinderen, M. (2018). Can app changelogs improve requirements classification from app reviews? an exploratory study. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–4.

Zhou, X., Zhang, Y., Cui, L., and Huang, D. (2020). Evaluating commonsense in pre-trained language models. In *AAAI*, pages 9733–9740.