LONTAR KOMPUTER VOL. 10, NO. 1 APRIL 2019 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2019.v10.i01.p01 **e-ISSN 2541-5832**
Accredited B by RISTEKDIKTI Decree No. 51/E/KPT/2017

Classification of Mobile Application Reviews using Word Embedding and Convolutional Neural Network

I Made Mika Parwita[1], Daniel Siahaan[2]

Informatics Department, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
[1] mika.parwita@gmail.com
[2] daniel@if.its.ac.id

Abstract

The app reviews are useful for app developers because they contain valuable information, e.g. bug, feature request, user experience, and rating. This information can be used to better understand user needs and application defects during software maintenance and evolution phase. The increasing number of reviews causes problems in the analysis process for developers. Reviews in textual form are difficult to understand, this is due to the difficulty of considering semantic between sentences. Moreover, manual checking is time-consuming, requires a lot of effort, and costly for manual analysis. Previous research shows that the collection of the review contains non-informative reviews because they do not have valuable information. Non-informative reviews considered as noise and should be eliminated especially for classification process. Moreover, semantic problems between sentences are not considered for the reviews classification. The purpose of this research is to classify user reviews into three classes, i.e. bug, feature request, and non-informative reviews automatically. User reviews are converted into vectors using word embedding to handle the semantic problem. The vectors are used as input into the first classifier that classifies informative and non-informative reviews. The results from the first classifier, that is informative reviews, then reclassified using the second classifier to determine its category, e.g. bug report or feature request. The experiment using 306,849 sentences of reviews crawled from Google Play and F-Droid. The experiment result shows that the proposed model is able to classify mobile application review by produces best accuracy of 0.79, precision of 0.77, recall of 0.87, and F-Measure of 0.81.

Keywords: *Convolutional Neural Network, mobile applications, Natural Language Processing, review classification, word embedding.*

1. Introduction

Mobile application store like Google Play, IOS AppStore, and Windows Phone Store provides features for users to search, download, and give a rating in text form [1], [2]. The developer uses reviews as information to maintain application development [3], [4]. The reviews can also be used as a reference for allocating development efforts, maintenance, and application quality improvement [5]–[7].

The rapid development of mobile application increases the number of reviews. For example, the facebook app receives more than 4275 reviews per day [3]. This challenging task for developers in analyzing and classifying app reviews regularly. The number of reviews is simply too large for manual checking and extremely consume a lot of cost, time, and effort [5]. Moreover, user reviews tend contains unstructured and informal sentences [6], [7]. User reviews might contain semantic sentence structure, i.e. synonym, homonym, and polysemy words contained in the review sentences. There are also useless reviews for developers, known as non-informative reviews [8], [9]. In other cases, non-informative reviews are also called spam reviews. These type of reviews tend not to be related to the content being discussed [10].

Research on the software reviews classification from application store has been done by many researchers, especially for mobile application. Maalej & Nabil use probability techniques based on the features of review metadata, keyword frequencies, linguistic rules, and sentiment analysis [1]. Review data is converted into Bag-of-word (BOW) then classified using three classification methods, Naive Bayes, Decision Tree, and MaxEnt. Other studies use a combination of Natural Language Processing (NLP), Sentiment Analysis (SA) and Text Analysis (TA) which are classified using five machine learning classification methods, namely Naive Bayes, Support Vector Machine, Logistic Regression, J48 and ADTree [5]. Puspaningrum et.al uses lexical similarity by utilizing term list to classifying the mobile application review. Three categories are used, i.e. bug report, feature request, and non-informative. This is the basis of this study to use these three categories. However, previous research did not consider semantic sentences for the classification. The Convolutional Neural Network (CNN) which is combined with word vector as the input, produces higher accuracy than linear classification methods [11]. In addition, CNN does not require term list to classify data in textual form. Another advantage of CNN is it can determine features automatically [11], [12]. The use of word embedding can handle semantic problems because each word is converted into vector based on the word relation in the sentence [13].

This research proposes a framework to classify reviews automatically using word embedding and binary classifier. Review data in sentences are converted into vectors using word embedding to handle semantic problem. The sentence vector is used as input for classification using CNN. The classification process is conducted twice, where the first classifier uses to classify the informative reviews and the second classifier to classify bug and feature request categories. The output of the first classifier is a collection of informative and non-informative reviews. Furthermore, informative reviews are reclassified using the second classifier to determine the category of review (bug report or feature request). As shown in the experiment result, the proposed model is able to classify mobile application review.

2. Review Categorization

Previous research describes reviews into four categories, i.e. bug report, feature request, user experience, and rating [1], [14], [15]. Other research describes five categories, i.e. Feature Request (FR), Problem Discovery (PD), Information Seeking (IS), Information Giving (IG), and Other (OT) [5], [7], [16]. User experience and rating categories are considered as non-informative reviews because they do not provide significant benefits for developers in maintenance and software evolution. Moreover, reviews in the experience category sometimes still overlapping to rating [8]. Therefore, reviews included in IG, IS, and OT categories are considered as non-informative reviews.

This study uses three categories, i.e. bug report, feature request, and non-informative. **Bug report** describes problems related to applications that must be corrected such as errors in functional or application performance issues. **Feature request** describes functionality or missing application content. Users may give ideas to improve application performance by adding or replacing application features. **Non-informative** describes reviews that do not provide significant benefits for developers in maintenance and software evolution process.

3. Research Methods

Proposed model consists of three modules, i.e. pre-processing, word embedding, and classification as shown in the figure 1. The pre-processing module processes review document using NLP technique so that ready to be used for the classification process. The word embedding module maps each word into vector. The classification module categorizes review sentences into informative and non-informative documents. The collection of informative sentences is classified into bug report or feature request categories using CNN.

3.1. Pre-processing

Pre-processing includes extraction process for textual data so that it can be used for classification [17]. In this research, there are five steps for the pre-process stage, i.e. lowercase, tokenization, stop-words removal, and spelling correction. **Lowercase** step changes reviews into standard form by converting all letters in review sentences into lowercase. **Tokenization** step aims to split sentences into words called token. Sentences are separated into tokens based on the space character. After that, tokens that are considered less relevant for the classification process will be removed during **stopwords removal** step. The list of stopwords used in this research is Google standard stopwords list.

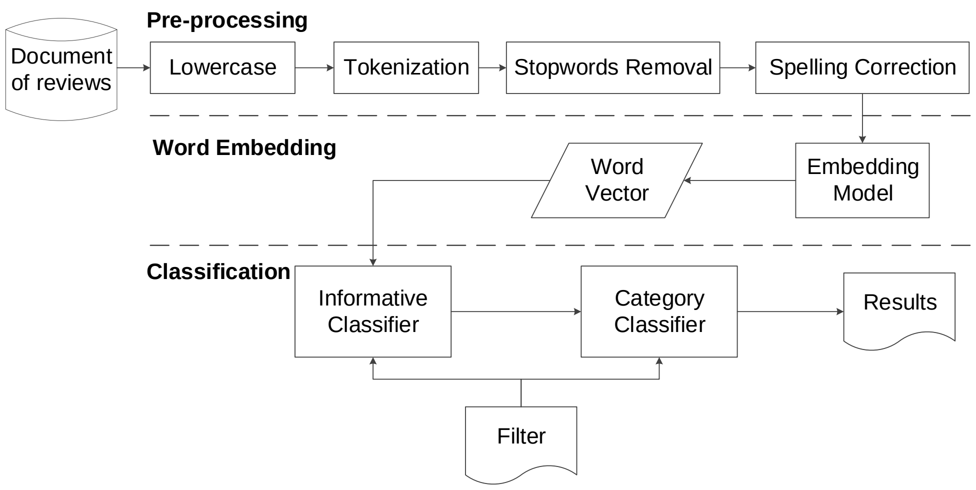

Figure 1. Proposed Model.

The final step is **spelling correction**, which aims to throw and change the abbreviation words into proper words. The habit of people in writing a review is to abbreviate the words, e.g. “don't” means “do not”, “isn’t” means “is not”, “can’t” means “cannot”, etc. This can affect the results of the classification if not corrected. The spelling correction words used in this research follow the spelling correction list used in [8].

3.2. Word Embedding

Word embedding is a language modeling technique on Natural Language Processing (NLP) where each word or phrase in vocabulary will be mapped into real number vector. The advantages of word embedding are being able to reduce dimensions of words vector and increase computing performance [11], [18]. Furthermore, word embedding can handle semantic sentence problems. This is because the process of forming words into vectors is based on the closeness of the word used in the sentence. The vector formed is a real number.

The most popular word vector is GloVe because it provides vectors in various dimensions, i.e. 50, 100, 200, and 300 dimensions. GloVe was developed by Pennington at Stanford University. This word vector is based on an unsupervised algorithm for tracing the representations of the word in vectors. GloVe is basically a log-bilinear method by giving values to the least squares that are generated from 6 billion corpus tokens from the Wikipedia data 2014 and English Gigaword Fifth Edition. The result of this process is vectors that represent the information of words. These word vector can produce high probability for words that are contexts of sentences and low probability for words that are not context. Furthermore, this vector becomes input for the classifier. The position of tokens that do not exist in the vocabulary model (out of vocabulary) is determined as a random vector. This research uses GloVe that has been trained and can be accessed publicly in the study [19].

3.3. Classification

The classification process consists of two modules, i.e. (1) informative review classification, detecting informative and non-informative reviews and (2) classification of review categories, specifying categories (bug reports or feature requests) from each review sentence. This study uses CNN as classifier. CNN is a type of neural network development that can be used for text classification. CNN consists of neurons that have weight, bias and activation functions. The architecture of CNN is divided into three parts, i.e. the convolutional layer, pooling layer, and Fully-Connected layer (FC-layer).

Convolutional layer produces filters with length and height (pixels). The initialized filters are shifted to all parts of reviews word vector. Each shift will be performed with a *dot* operation between word vector and value of filters. The output is called an activation map or feature map. The filter is shifted based on stride and padding that was previously determined as a parameter [20].

Pooling layer consists of filters with certain size and stride that will shift across the feature map. This research uses max pooling as pooling layer. Max pooling collects maximum value to generate a new matrix from feature maps. The main purpose of pooling layer is to reduce the dimensions of the feature map without losing important information from the matrix. This process is able to accelerate computation because the parameters that are processed further are smaller and can overcome *overfitting* [21].

FC layer consists of the hidden layer, activation function, an output layer and a loss function. The output from FC layer is processed using softmax function with the aim to specify the category of input review. The output of softmax function represents a category distribution, i.e. probability distribution of a number of possible K results. Given input vector x, weight vector w, and $x^\neg w$ denotes the inner product of x and w, softmax function is defined in equation (1).

$$P(y - j|x) = \frac{e^{x^\neg w_j}}{\sum_{k=1}^{K} e^{x^\neg w_k}} \tag{1}$$

$$H(y, \hat{y}) = \sum_i y_i \, log \frac{1}{\hat{y}_i} = -\sum_i y_i \, log \, \hat{y}_i \tag{2}$$

The difference between softmax output and ground truth H is calculated using cross entropy objective function. Cross entropy is defined in equation (2). Parameter y denotes ground truth and $\hat{y}$ denotes softmax output. The softmax function is used twice in this research, (i) function in the first classifier indicates input review sentence into informative or non-informative. (ii) The second function is used in the second classifier to indicates bug or feature request. The quality of experiment is examined by using accuracy, precision, recall, and F-measure. The value of performance is declared in decimal units.

4. Dataset and Experiment

4.1. Dataset

This research uses the dataset that was obtained from [16]. The dataset is obtained by crawling on the Google Play mobile application store that is associated with Android F-Droid application provider. The number of data is 288,065 reviews from 395 different applications. Reviews are broken down into a collection of sentences. The total number of sentences is 451.293 sentences. The sentences classified into three categories, i.e. feature request, bug report, and non-informative. The final number of sentences per category is shown in table 2.

4.2. Data Cleaning

This research applies data cleaning to minimize noise. The data cleaning process removes non-latin characters, reviews that only consist of punctuation, reviews without the label, blank reviews, and duplicate reviews. The removal of non-latin characters and punctuation uses Regular Expression (ReGex). Punctuation marks to be removed i.e. comma (,), period (.), exclamation point (!), question mark (?), quotation mark/inverted comma ("), colon (:), semicolon (;), ellipsis (…), hyphen (-), n-dash (–), and m-dash (—). Furthermore, blank review, reviews without the label, and duplicate reviews are removed by Weka application.

Table 1. Details of Data Cleaning Process.

Cleaning Process	Number of Initial Sentences	Number of Final Sentences	Removed Sentences

Remove non-latin character	451,293	450,137	1,156
Remove full punctuation reviews	450,137	448,022	2,115
Remove reviews without labels	448,022	447,955	67
Delete blank reviews	447,955	435,484	12,471
Remove duplicate reviews	435,484	306,849	128,635
Total Removed Sentences			**144,444**

The number of initial data from [16] is 288,065 reviews that are consisted of 451,293 sentences. Data cleaning process eliminates 144,444 sentences, so the number of final data is 306,849 sentences. Table 1 shows the number of data that were removed for each data cleaning process. The number of clean sentences for each category is shown in table 2.

Table 2. Number of Sentences Each Category.

Category	Number of Sentences
Feature Request	16,212
Problem Discovery	30,369
Non-informative	260,268
Total Sentences	**306,849**

In addition, the collected data is divided into two parts, i.e. training and testing data with a ratio of 80:20. Data for training as much as 80% and testing data as much as 20%. The data separation is determined randomly. Furthermore, the experiment uses cross validation to increase the relevance of experiment data.

4.3. Experimental Setup

Conversion of reviews into vector uses four variants of GloVe as word embedding, i.e. 50, 100, 200, and 300 dimensions. This aims to determine the performance of different dimensions of word embedding. The CNN parameters used for classification in this study based on [22]. Some parameters are used for CNN application for text classification, i.e. zero padding (set to 0), the stride of 1, mini-batch size of 128, and one epoch. ReLU refers to Rectified Linear Unit and 1-max pooling as commonly used in CNN also used in this experiment. Region value of 1 with 100 feature maps each. Some parameters are tuned based on the number of words per sentence in dataset. In Giovanni's dataset, the average number of words per sentence is 15 words. Tuning process is carried out with a variant of certain values to obtain values for regularization and kernel parameters based on [22]. The basis for determining the best parameters is the value of parameter that produces the highest F-measure for classification. So that the best kernel for Giovanni’s dataset is 1 and best regularization parameters include dropout rate 0.5 and l_2 constraint $1x10^{-1}$.

5. Result and Discussion

The experiment result is shown in table 3. It can be seen that the use of 200-dimension of GloVe word vector produces the highest F-measure compared to other dimensions for informative and category classifier. F-measure by 0.671 for the informative and non-informative classifier (Informative classifier) and 0.819 for bugs and feature requests classifier (Category classifier). This is because of the number of vector dimensions is correspond to the used parameters.

Table 3. Classifier performance.

Word Vector	Informative Classifier	Category Classifier

Dimension	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
50	0.887	0.639	0.625	0.632	0.543	0.554	0.610	0.581
100	0.885	0.734	0.596	0.658	0.732	0.733	0.803	0.766
200	**0.890**	**0.738**	**0.681**	**0.671**	**0.793**	**0.772**	**0.871**	**0.819**
300	0.888	0.754	0.605	0.671	0.815	0.831	0.607	0.556

The 300-dimension results close to 200-dimension. The first classifier's precision in 300-dimension produces a higher value, but not significant. The results of this experiment support research in [22] that discusses the number of words per input vector and the word vector dimension affecting the classification results. So, the selection of the dimensions of the word vector depends on the number of words in the review sentence. A sentence has 15 words on average in the dataset which used in this research. Moreover, experiment using 100-dimension always produces the lowest value when implemented for category classifier. This may be affected by the amount of test data for the classification. The number of test data for category classifier is around 15,016 sentences and dropout rate by 0.5. The dropout rate affects the final results depending on the dataset [12], [22].

Figure 2 shows the performance of final accuracy for informative and category classifier. The final accuracy is obtained by calculating the results of informative classifier followed by category classifier. The informative reviews that are predicted as informative (true positive) on informative classifier are classified using category classifier to determine the category (bug report or feature request). In this way, performance informative classifier combined with category classifier can be obtained. The final accuracy It can be seen 200-dimension produces a higher accuracy compared to other dimensions which produces best accuracy value by 0.53. This is due to result of informative and category classifier where 200-dimension always produces the best performance for recall and accuracy. However, the performance accuracy of each classifier (shown in table 3) decreases compared to the accuracy of the combined two classifiers. This is due to the high false positives obtained from the informative classifier. False positives from informative classifier are non-informative reviews that are predicted as informative reviews by the system. The number of false positives is added as a divider to calculate the final accuracy.

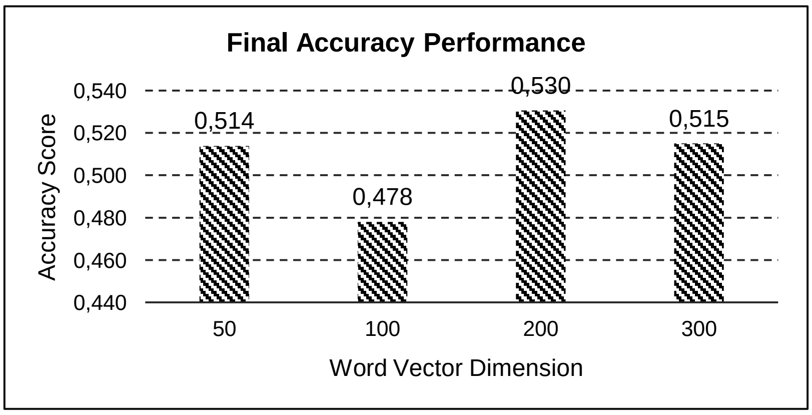

Figure 2. Final Accuracy Performance.

Based on the experiment result, the proposed model is able to classify mobile application review. Compared to the LSTM as classifier in Puspaningrum et al. [8], the proposed model produces higher precision, recall, and F-measure. The precision, recall, and F-measure produced by Puspaningrum et al. are 0.564, 0.507, and 0.491 respectively. The proposed model produces 0.772, 0.871, and 0.819. One possible factor that may affect the different result is that the number of sentences in the dataset is different. The experiment of the proposed model used more data than Puspaningrum et al. It means more vocabulary are captured by word vector. From the comparison result, the CNN combined to word embedding as input is able to handle the review classification.

6. Conclusion

This research proposed CNN which was built on top of GloVe word vector to handle mobile application review classification. The classification model classifies review into three categories, i.e. bug report, feature request, and non-informative. The experiment uses 306,849 sentences of mobile application reviews. The best performance is produced by using GloVe 200-dimension as word vectors in word embedding process. Two classifiers were used to classify reviews, (i) classifier to classify informative and non-informative sentences and (ii) classifier to detect the category of informative sentences (bug report or feature request). The result shows that the proposed model is able to classify reviews by F-measure values 0.671 for the informative and non-informative classifier. Furthermore, the category classifier produces F-measure by 0.819 and the best final accuracy by 0.53.

However, we found an issue that may affect the overall performance. The issue is the effect of the number of words per sentences on the word vector dimension. To solve this problem, tuning parameters for CNN may be needed for different types of datasets. For the future work, word position in a vector can be improved by using other word vectors, e.g. Word2Vec, Senna, or non-static word vector.

References

[1] W. Maalej and H. Nabil, “Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews,” *2015 IEEE 23rd international requirements engineering conference (RE)*, pp. 116–125, 2015.

[2] E. Guzman, M. El-halaby, and B. Bruegge, “Ensemble Methods for App Review Classification : An Approach for Software Evolution,” *30th IEEE/ACM International Conference on Automated Software Engineering*, pp. 771–776, 2015.

[3] M. Lu and P. Liang, “Automatic Classification of Non-Functional Requirements from Augmented App User Reviews,” *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pp. 344–353, 2017.

[4] A. E. Hassan, S. Mcilroy, N. Ali, H. Khalid, and A. E. Hassan, “Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews issues that are raised in mobile app reviews,” *Empirical Software Engineering*, no. July, 2016.

[5] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, “How Can I Improve My App ? Classifying User Reviews for Software Maintenance and Evolution,” *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 281–290, 2015.

[6] D. Galih, P. Putri, and D. O. Siahaan, “Software Feature Extraction using Infrequent Feature Extraction,” *6th International Annual Engineering Seminar (InAES)*, pp. 165–169, 2016.

[7] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. Gall, “ARdoc : App Reviews Development Oriented Classifier,” *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 1023–1027, 2016.

[8] A. Puspaningrum, D. Siahaan, and C. Fatichah, “Mobile App Review Labeling Using LDA Similarity and Term Frequency-Inverse Cluster Frequency (TF-ICF),” *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE). IEEE*, 2018.

[9] K. Giannakopoulos, “Informative vs . Non-informative Short Message Detection in Social Networks,” *International Conference on Big Data Computing and Communications Informative*, pp. 165–171, 2017.

[10] A. R. Chrismanto and Y. Lukito, “Identifikasi Komentar Spam Pada Instagram,” *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 8, no. 3, p. 219, 2017.

[11] Y. Goldberg, “A Primer on Neural Network Models for Natural Language Processing,” *Journal of Artificial Intelligence Research 57*, vol. 57, pp. 345–420, 2016.

[12] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *arXiv preprint arXiv:1408.5882*, 2014.
[13] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, “Semantic Clustering and Convolutional Neural Network for Short Text Categorization,” *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 352–357, 2015.
[14] W. Maalej, Z. Kurtanovic, H. Nabil, and C. Stanik, “On the Automatic Classification of App Reviews,” *Requirements Engineering*, pp. 311–331, 2016.
[15] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, “Towards Data-Driven Requirements Engineering,” *IEEE Software SI - FUTURE OF SOFTWARE ENGINEERING*, vol. 33, no. 1, pp. 48–54, 2015.
[16] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella, “Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement,” *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, pp. 8–11, 2017.
[17] N. N. E. Smrti, “Otomatisasi Klasifikasi Buku Perpustakaan dengan Menggabungkan Metode K-NN dengan K-Medoids,” *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 4, no. 1, pp. 201–214, 2013.
[18] A. Risteski, “RAND-WALK : A latent variable model approach to word embeddings,” *ArXiv preprint arXiv:1502.03520*, pp. 1–33, 2015.
[19] J. Pennington, R. Socher, and C. D. Manning, “GloVe : Global Vectors for Word Representation,” *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.
[20] B. Jan, H. Farman, M. Khan, M. Imran, I. Ul, A. Ahmad, S. Ali, and G. Jeon, “Deep learning in big data Analytics : A comparative study,” *Computers and Electrical Engineering*, pp. 1–13, 2017.
[21] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, no. October 2016, pp. 11–26, 2017.
[22] B. C. Wallace and Y. Zhang, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification,” *arXiv preprint arXiv:1510.03820*, 2016.