Text analysis and classification: Reddit dataset

Mohammadreza Sadeghi, Bahareh Nikpour, Negar Hassantabar

Abstract

Reddit is a social media platform containing several communities. In this work, our goal is to train a classifier on a bunch of comments from 20 different societies with their associated categories in such a way that it can classify unseen test text samples. Our work can be split to two phases: First, the features of the data are extracted and the feature vector are transformed to numerical features. Secondly, a suitable machine learning approach is selected to reach a good performance. Also, Bernoulli naïve Bayes algorithm is implemented from scratch to be tested on our data. During our work, we evaluated the performances of several classifiers such as Multinomial naïve Bayes, Decision tree, Multi-layer perceptron, k nearest neighbors, Support vector machine, to define the best one. Afterward, we applied the ensemble of the results of these algorithms to reach a potential performance. The results have shown the ensemble method performs better.

1. Introduction

In this project the main task was text analysis and classification on comments from Reddit website (https://www.reddit.com/) using a suitable machine learning model. The main objective is to analyze each comment and assign it to its desired category from 20 different classes. The train data set in hand contains 70000 comments with their ID and target class. As this dataset is a raw, it contains the original comments with no specific features; thus, at the very first step, analyzing and evaluating the dataset in detail is a must to identify features for the machine learning model. We have realized that this data set contains a lot of unrelated comments e.g. “hi”, “wow”, etc which are associated with a label that is meaningless and confuses the classification algorithm. These samples can be considered as outliers. Moreover, the length of the comments of each samples are very different that brings more difficulty to our model. Overall, we have figured out that our data hard to classify for the classical machine learning models.

After vectorising the data and transforming it to a numerical format, the next step is to implement different algorithms such as Bernoulli Naïve Bayes, which we have implemented it from scratch, and Logistic Regression or Decision Tree from SciKit Learn packages models to be applied on the preprocessed datasets; however, as was mentioned, the most challenging part is not related to the algorithms, but to the data preprocessing. To vectorize the data, each word can be considered as a feature, but it will lead to very high dimensional vectors and some of them may be unnecessary or even mislead the algorithm, so we intend to remove some of them that are called stop words to improve the performance of the algorithm. In the following, we first provide a brief literature review in section 2. Afterward, we briefly explain the preprocessing approaches we used for our data set in section 3. The proposed approach as well as the obtained results are given in sections 4 and 5 respectively. Finally, in section 6, our conclusion is drawn.

2. Related works

Sentiment analysis has become an interesting subject in the past few years for academia and industry as it is a challenging area of text classification in the research fields of Natural language processing (NLP), data mining (DM) and information retrieval (IR) [1, 2].

The most commonly used techniques in the sentiment analysis area are old text classification approaches such as mapping a Bag of words (BOW) to a feature vector and classifying by the Naive Bayes, Maximum entropy or Support vector machines models; however, the experimental results show that important characteristics are not met in BOW technique since it is not holding the information of word order, syntactic structures and semantic relationships between words. As a result, ensemble techniques are proposed to be used in sentiment classification [2].

In general, there are three automatic text summarization methods: surface methods, entity level and discourse level methods. In the surface method we consider shallow text features to process the document which means the most frequent and best positioned words are considered as the most relevant while in the entry level method, the relations in the text is also important. Finally, in the discourse method the structure of the whole text is considered important [3].

Additionally, deep learning techniques have attracted a great deal of attention as these techniques provide automatic feature extraction that have better performance than traditional manually extracted features like surface methods. Moreover, the performance of deep learning techniques has been improved by proposing ensemble techniques in which surface methods are integrated with deep learning methods [4].

3. Dataset and setup

As was mentioned before, our data include comments in a social media platform, called reddit, from 20 different communities. Our goal is to train a machine learning model in such a way that in can predict the categories of comments from an unseen test data with a good performance. To be able to feed the train data to the classifier, two steps are needed. Firstly, in the feature extraction step, the training comments should be parsed and important words should be found which is called tokenizing. At the second step, the tokens must be transformed to numerical feature vectors. so that in can be in a proper format for the classifier.

For feature extraction, scikit-learn of Python (https://scikit-learn.org/) provides several functions the performance of which is explained in below:

CounterVectorizer: This is a function which provides a bag of word and makes a dictionary of features [5]. The length of the output feature vector is equal to the number of words appearing in the dictionary and it counts for the number of words existing in each data sample, i.e. a text string to transform the data to a feature vector. It is possible to use binary features 0 and 1 instead of counting which shows whether a word appears or not in our data sample. Also, it has a vocabulary of stop words that enables us to filter some unhelpful words like “the” which gives us no sentiment about the text and makes our model inaccurate and complicated. One can use either the provided stop word vocabulary of this function or his own stop words. Another thing that we can evaluate on our data is choosing different n for n-grams. n-grams count for co-occurrence of the words. For example, in one-gram, only one word is considered while in bi-gram, two subsequent words are considered together simultaneously. Choosing n depends on the text data and its nature, and should be defined per case. The output feature vector of this function is a very sparse long vector meaning it has a lot of zeros and some ones. As a result, Python shows the output dataset as a sparse matrix.

TfidfVectorizer: There may exist some words that appear very frequently in the data but gives no information [6]. To address this issue, some methods find the frequency one of which is Tf-Idf method. This method first finds the frequency of a word happening in a document and then analyzes whether that word has appeared a lot or not in all documents. If it occurs a lot, the corresponding weight is downscaled. These two parameters are calculated as:

$$TF(X) = \frac{Number\ of\ times\ the\ word\ X\ is\ seen\ in\ a\ document}{Total\ number\ of\ words\ in\ the\ document}$$

$$IDF(X) = log\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ word\ X}$$

Also a technique called stemming can be used to reduce the size of dictionary [7]. In this technique, each word is transformed to its root based on the concept that by doing so, the sentiment will not change. For example, all verbs in past, present and future (like give, gave, given, gives) will be the same after stemming (give).

4. The proposed approach

In this section, we first provide a brief explanation about the classifiers we used in our best performing approach as well as Bernoulli naïve Bayes method which we have implemented from scratch and afterward, we explain the methods we tried to reach our best model.

4.1. Classification approaches

a. Bernoulli and Multinomial Naïve Bayes

Naïve Bayes classifiers are set of classifiers that use Bayes theorem to classify data points [8]. Some of the Naïve Bayes classifiers such as Gaussian Naïve Bayes, Bernoulli Naïve Bayes, and Multinomial Naïve Bayes are popular for Text Classification. Bayes rule is described as follow:

$$P(c|x_i) = \frac{P(c)P(x_i|c)}{P(x_i)}$$

$$P(c) = \frac{N_c}{\sum_i N_i}$$

Where $P(c)$ is an estimated by dividing the total documents that belong to class c and $P(x_i|c)$ is the probability of document x_i belong to class c. For Bernoulli Naïve Bayes classifier, $P(x_i|c)$ using Laplace smoothing is defined as follow [9]:

$$P(n|c) = \frac{df_{n,c} + 1}{df_c + 2}$$

Where $df_{n_{ic}}$ is the number of documents that contain word n and belong to class c and df_c is the total number of documents in class c. For predicting the class of a new data point, MAP decision rule is used as follow.

$$c_k = argmax_j P(c_j) P(x_k|c_j)$$

In the Multinomial Naïve Bayes, the $P(x_i|c)$ is calculated as follow [8]:

$$P(x_i|c) = \frac{(\sum_n f_{ni})!\ \Pi_n P(w_n|c)^{f_{ni}}}{f_{ni}!}$$

Where f_{ni} is the number of n^{th} word in x_i and $P(w_n|c)$ is probability of appearance of n^{th}word given class c. This probability can be estimated using Laplace smoothing as follow:

$$P(w_n|c) = \frac{1 + F_{nc}}{N + \sum_x F_{xc}}$$

Where F_{xc} is the total number of word x in the class n.

For classifying an unseen data, we choose a class that maximize the MAP rule:

$$c_k = \boldsymbol{argmax_j P(c_j) P(x_k|c_j)}$$

b. Support Vector Machine (SVM)

SVM classifier is a kind of supervised Method that try to find sets of hyperplanes having maximum margins from data points [10]. In a binary classification problem, the following optimization technique is used to find the hyperplane:

$$\min_{w,b} \frac{1}{2} \left|\left|w\right|\right|^2 + b^2$$

$$s.t \quad y_i(w^T x_i - b) \geq 1$$

Where w is the normal vector of the hyperplane and b is bias term for the hyperplane. x_i is a data point and $y_i \in \{-1,1\}$ is the label corresponding to that.

The aforementioned method is for binary class SVM. For multi class SVM, a key idea is to make use of some binary SVM classifiers and perform $m-1$ one versus all algorithm to find $m-1$ hyperplanes separating our data points. In this algorithm, first we choose one class of data and classify its data points against all other classes. This algorithm performs well in some applications [1]; however, it is time consuming.

Another approach for classifying multi class data sets by SVM is to reformulate the original problem to solve the classification problem. To this end, a new cost function is defined as below:

$$\min_{w_{j,b_{j,\zeta_l}}} \sum_j \left|\left|w_j\right|\right|^2 + b_j^2 + \lambda \sum_l \zeta_l$$

$$s.t \quad w_{y_i}^T h(x_i) - b_{y_i} + \zeta_i > w_k^T h(x_i) - b_k + 1 \quad \forall i,k \in \{1,\dots,N\} \,\&\, k \neq y_i$$

Where N is the number of classes, $h(.)$ is a suitable kernel, and ζ_i are some positive numbers.

When this optimization is solved, we reach some hyperplanes that separate our data point. We classify a new data point based on the following rule:

$$y_i^* = argmax_k \; w_k^T h(x_i) - b_i$$

c. Multi Class Logistic Regression

Logistic Regression model is a probabilistic model which tries to maximize the probability of a certain data point belonging to its corresponding class [11]. This model can be implemented either by one versus all method or by changing the loss function (cross entropy loss) to multi variable cross entropy loss. The multi class cross entropy function is defined as:

$$L = \sigma(w_1^T x_i)^{1\{y_i=1\}} \sigma(w_2^T x_i)^{1\{y_i=2\}} \dots . \phi_n(x_i)^{1\{y_i=n\}}$$

$$\phi_n(x_i) = 1 - \sum_k \sigma\left(w_k^T x_i\right)^{1\{y_i=k\}}$$

Where the weights w_i are hyperplanes that separate our data and $\sigma(.)$ and $1\{.\}$ are sigmoid function and indicator function respectively.

For finding the best w_i for separating our data points, we must solve the following optimization problem:

$$\max_{w_1,w_2,\dots,w_{n-1}} L$$

In order to solve the optimization problem Gradient Ascent algorithm is used as:

$$w_i = w_i + \alpha \frac{\partial L}{\partial \mathrm{w_i}}$$

where α is learning rate.

4.2. The proposed approach

In our method, we first used CounterVectorizer to create the bag of words and build the feature vector by counting the number of words appearing in each sample. We used most frequent words in the document in addition to the existing stop word dictionary to eliminate useless words and reduce the length of our vector and the best for n-grams was found to be 1 since as shown in appendix one by elimination one gram frequent words two grams frequent words are also eliminated. Afterward, we used Tf-Idf algorithm to rescale the weights of each word appearing in the document. We tried to use stemming before our preprocessing but we could not get good results by that, so we eliminated that step. After all these steps, we have a sparse training data that can be fed to the classifier. We analyzed different classifiers (with different parameters) including Multinomial naïve Bayes, SVM, k-nearest neighbors, Multilayer Perceptron, Decision tree and logistic regression as well as Random forest algorithm which is considered as an ensemble method. Next, we tried to ensemble the classifiers with the highest accuracies to reach the best combination. We realized that the best combination in our case is using Multinomial naïve Bayes, SVM and logistic regression with equal weights in voting.

5. Experimental results

To evaluate our models, we selected 70% of our randomly permuted data samples as train set and the rest as validation set. Also, we evaluated our models using accuracy measure. The parameters we used for our classifiers are shown in table 1. Also, the accuracy results of applying all methods that we analyzed on the validation set are shown in table 2. The results show that Multinomial naïve Bayes, logistic regression and SVM algorithms have obtained the best results respectively. As a result, we combined these three methods as an ensemble algorithms and the result shows 1 percent improvement. As was mentioned, our data is difficult to learn, so even one percent here is a significant improvement. Afterward, we applied our best model on the

given test-reddit data set and obtained the accuracy of **57.45%**.

Table1. Parameters of our model

Model	Parameter
Multilayer Perceptron	Hidden layer sizes: (24, 16)
k Nearest Neighbors	K=7, Distance:Euclidean distance
SVM	Kernel: Linear

Table2. The obtained accuracy of our methods on the validation set

Model	Accuracy%	Rank
Multilayer Perceptron	44.28	5
K Nearest Neighbors	6.23	7
SVM	54.12	3
Logistic regression	54.64	3
Decision tree	45.24	4
Multinomial Naïve Bayes	55.91	2
Random forest algorithm	43.21	6
Ensemble of Multinomial Naïve Bayes, SVM and Logistic regression	56.85	1

To analyze the results in more detail, as Table2 shows, k-Nearest Neighbors algorithm could poorly classify our dataset. This is because when number of words in the bag of words grows bigger, the dimension of word vector of each data point increase too and in high dimension, Euclidian norm is meaningless. In this high dimensional data points, some algorithms, such as Multinomial Naïve Bayes and logistic regression, that try to predict the distribution of data points given labels, could perform better than other algorithms. Also, Multi label Linear SVM could separate data points in the high dimensional space since it tries to find hyperplanes with maximum margin. At the end, we ensemble the results of the three algorithms (Multinomial Naïve Bayes, SVM and Logistic regression). We used majority voting technique for prediction of label of each validation data point, however, when these models predict three different labels, the model with the highest accuracy (Multinomial Naïve Bayes) indicates the label.

6. Conclusion

In this project, we tried to classify some comments from a well-known social media, Reddit website, into twenty classes. To this end, first we tried to extract some useful features from data points and deleted stopping words from the dataset. Then we vectorized our dataset using word counting and TF-IDF techniques. Second, we evaluated different classifiers such as Logistic Regression, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Multi Layer Perceptron, Random Forest and SVM to classify the feature vectors. Finally, we ensemble classifiers with the highest accuracies and apply that on validation dataset in order to get a better performance. At the end, we used our ensemble model to classify the reddit_terst data set and reported the obtained accuracy.

Statement of Contributions

All the members of the team contributed in both programming and write up. Bahareh Nikpour and Mohammadreza Sadeghi focused on the kaggle competition and data preprocessing. Negar Hassantabar wrote the Bernoulli naïve Bayes code.

Refrences

[1] B. Liu, Sentiment analysis: a multifaceted problem, IEEE Intelligent System 25 (2010) 76–80.

[2] R. Xia Ch. Z. Shoush, Ensemble of feature sets and classification algorithms for sentiment classification, Information Sciences, Volume 181, Issue 6 (2011), 1138-1152.

[3] N. M. Preradović, Damir Boras, Marta Vlainic, Importance of Surface Methods in Human and

Automatic Text Summarization, Journal of Computers 8, (2014), 9-16.

[4] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, C. A.Iglesias, Enhancing deep learning sentiment analysis with ensemble techniques in social applications, Expert Systems with Applications, Volume 77, (2017), 236-246.

[5] Singh, P. (2019). Natural language processing. In Machine Learning with PySpark (pp. 191-218). Apress, Berkeley, CA.

[6] H. Wu and R. Luk and K. Wong and K. Kwok. "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, 26 (3). 2008.

[7] Lovins, Julie Beth (1968). "Development of a Stemming Algorithm" . Mechanical Translation and Computational Linguistics. 11: 22–31

[8] Kibriya, Ashraf M., et al. "Multinomial naive bayes for text categorization revisited." Australasian Joint Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2004.

[9] Raschka, Sebastian. "Naive bayes and text classification i-introduction and theory." arXiv preprint arXiv:1410.5329 (2014).

[10] Liu, Yi, and Yuan F. Zheng. "One-against-all multi-class SVM classification using reliability measures." Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. Vol. 2. IEEE, 2005.

[11] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., & Klein, M. (2002). Logistic regression. New York: Springer-Verlag.

Appendix 1

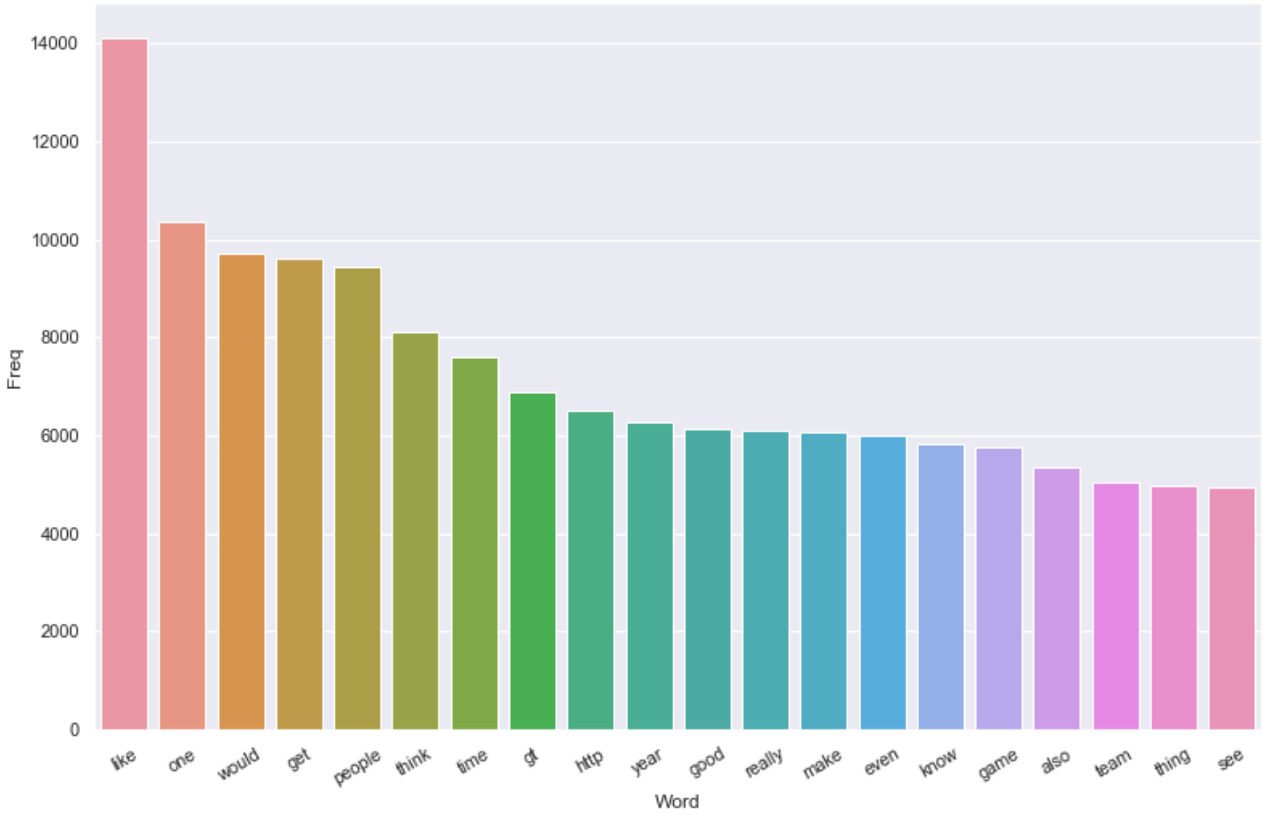

Figure 1: The most frequent 1-gram

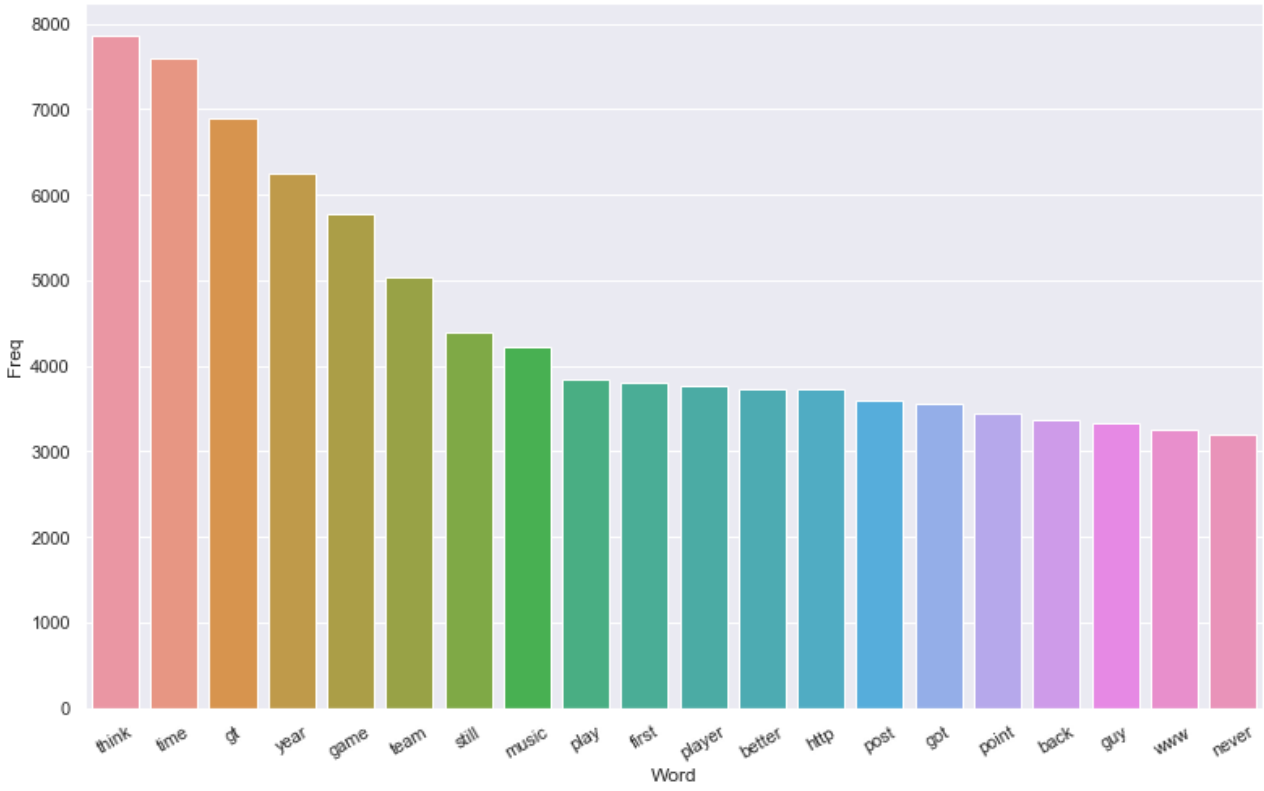

Figure 2: The most frequent 1-gram after eliminating not relevant frequent words

Appendix 2

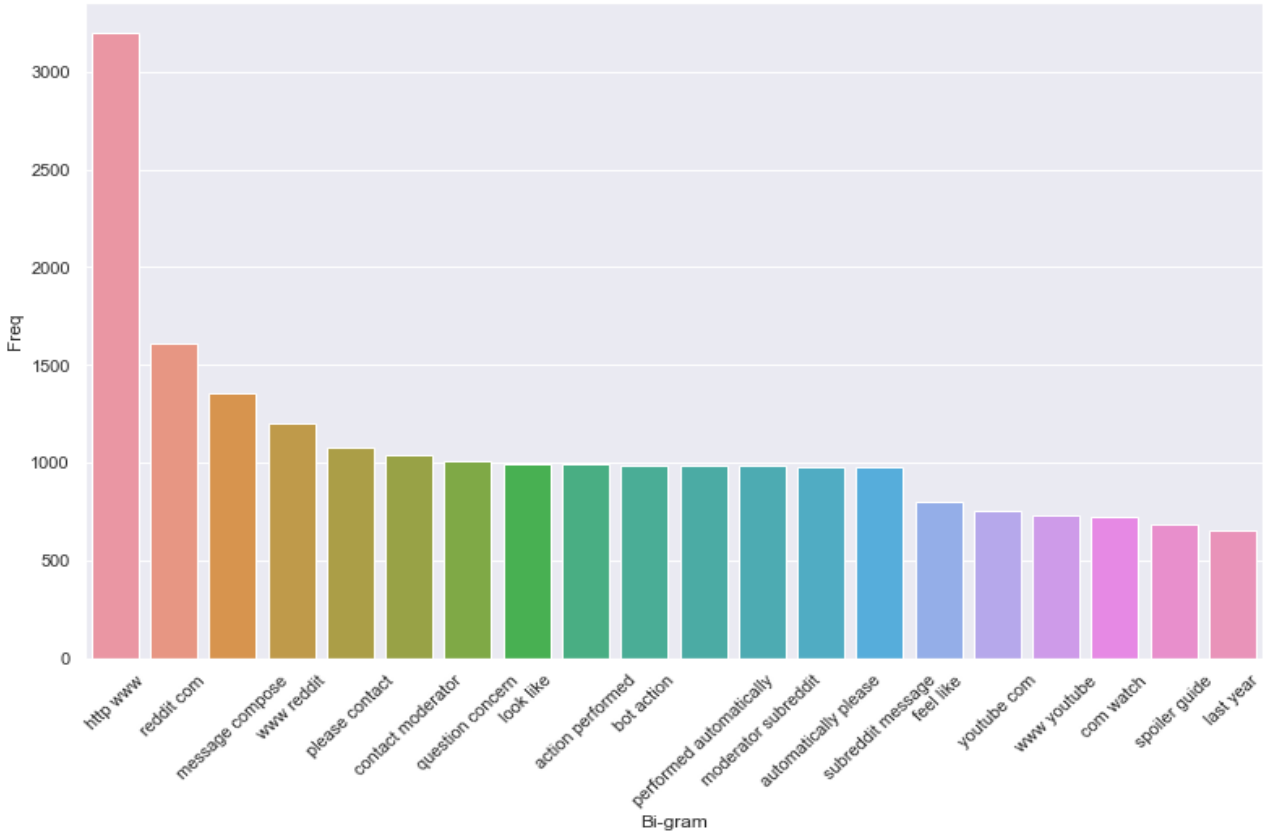

Figure 3: The most frequent 2-gram

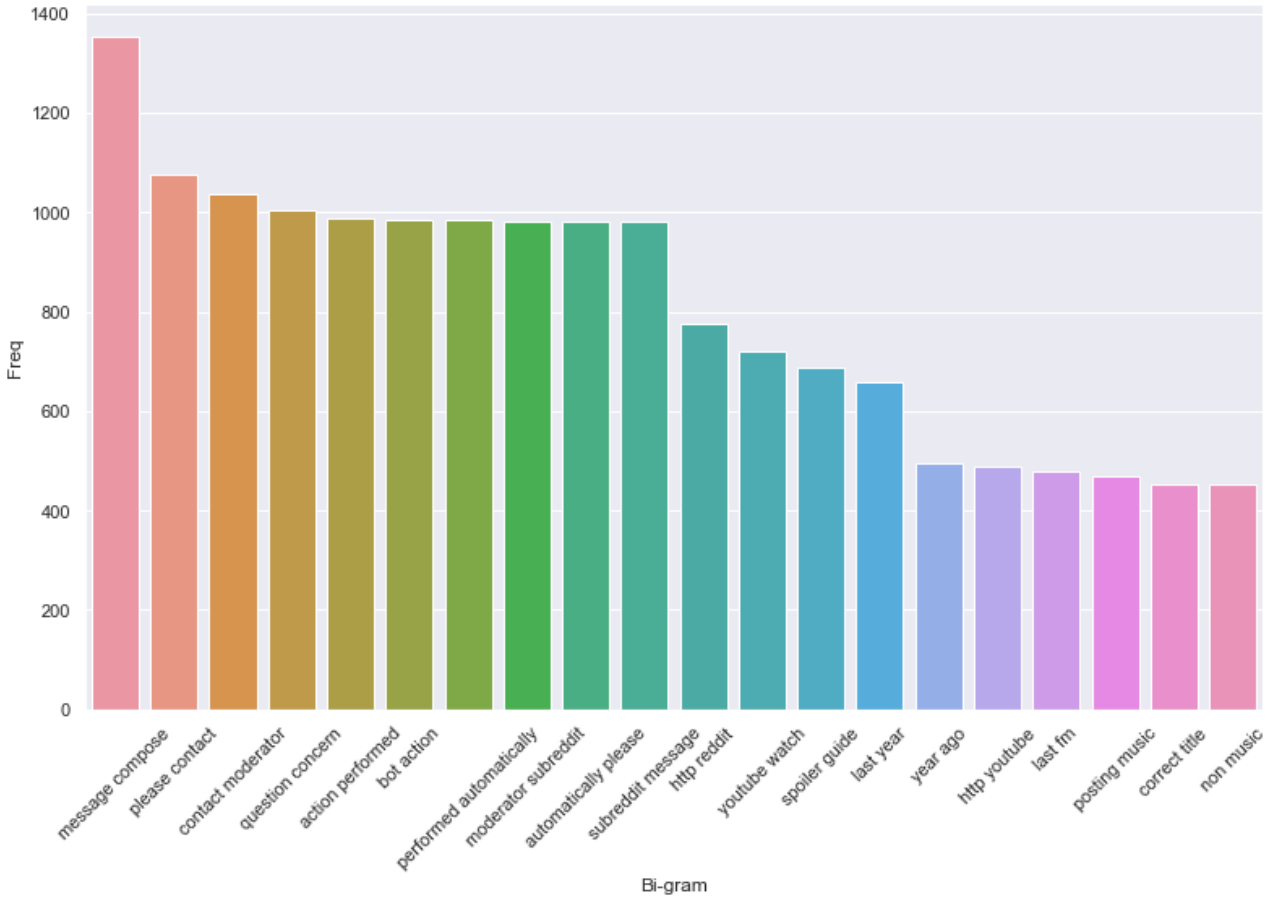

Figure 4: The most frequent 2-gram after eliminating not relevant frequent 1-gram words