

**T.C.**

**GEBZE TECHNICAL UNIVERSITY**

**Computer Engineering Department**

**TOXIC COMMENT CLASSIFICATION**

**Taha Atakan İpekçi**

**Project Advisor**

**Dr. Yakup GENÇ**

**MAY,2018**

**Gebze, KOCAELİ CSE 495**



**T.C.**

**GEBZE TECHNICAL UNIVERSITY**

**Computer Engineering Department**

**TOXIC COMMENT CLASSIFICATION**

**Taha Atakan İpekçi**

**Project Advisor**

**Dr. Yakup GENÇ**

**MAY,2018**

**Gebze, KOCAELİ CSE 495**

Bu çalışma ..../..../200.. tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü’nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

|  |  |  |
| --- | --- | --- |
| Danışman Adı |  |  |
| Üniversite |  |
| Fakülte |  |
|  |  |  |
| Jüri Adı |  |  |
| Üniversite |  |
| Fakülte |  |
|  |  |  |
| Jüri Adı |  |  |
| Üniversite |  |
| Fakülte |  |

**TABLE OF CONTENTS**

**FIGURE LIST...................................................................................................VI**

**ACRONYM LIST............................................................................................VII**

**ABSTRACT....................................................................................................VIII**

**1. INTRODUCTION.......................................................................................... 1**

**2. FEATURE EXPLORATION......................................................................... 3**

**3. WORD VECTORIZATION.......................................................................... 6**

**3.1 TF-IDF............................................................................................................................ 6**

**3.2 WORD2VEC.................................................................................................................. 7**

**3.3 GLOVE........................................................................................................................... 8**

**4. IMPLEMENTED MODELS.......................................................................... 9**

**4.1 NAIVE BAYES............................................................................................................. 9**

**4.2 LOGISTIC REGRESSION....................................................................................... 11**

**4.3 LSTM........................................................................................................................... 13**

**5. TURKISH DATASET................................................................................... 15**

**6. CONCLUSION AND FUTURE WORK..................................................... 17**

**7. REFERENCES.............................................................................................. 18**

**FIGURE LIST**

[Figure 1 Examples from the training data 3](#_Toc515416995)

[Figure 2 Histogram Graph for the Wikipedia Dataset that shows the total number of each label 4](#_Toc515416996)

[Figure 3 The length of comments(chararacter length) 5](#_Toc515416997)

[Figure 4 A word cloud created by some of the words that were taken from comments that are labeled as toxic 5](#_Toc515416998)

[Figure 5 The 10 most similar words to "ugly" and "great" in the word2vec model trained by the words in the training dataset 7](#_Toc515416999)

[Figure 6 Confusion Matrix for the "Toxic" label 9](#_Toc515417000)

[Figure 7 Confusion Matrix for "Threat" label 10](#_Toc515417001)

[Figure 8 Confusion Matrix for the "Toxic" label. Balanced LR 11](#_Toc515417002)

[Figure 9 Confusion Matrix for the "Threat" label. Balanced LR 12](#_Toc515417003)

[Figure 10 Confusion Matrix for the "Threat" label. Unbalanced LR 12](#_Toc515417004)

[Figure 11 The results of the Neural Network with 2 Epochs 14](#_Toc515417005)

[Figure 12 Word examples from Turkish Dataset Toxic Comments 15](#_Toc515417006)

[Figure 13 The Confusion Matrix for the Toxic Turkish Data 16](#_Toc515417007)

**ACRONYM LIST**

**CNN :** Convolutional Neural Networks

**IDF :** Inverse Document Frequency

**LR :** Logistic Regression

**LSTM :** Long Short Term Memory

**TF :** Term Frequency

**NB :** Naive Bayes

**NN :** Neural Networks

**ABSTRACT**

Online discussion about the subjects you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. Simple word censoring is not enough to prevent toxicity since the real danger is not only in words but also in meanings as well. Toxic comments can include different toxic subjects. They can include some simple insults or vulgar words but they can also be racist or even life threatening.

Some models were built to find toxicity bu so far the current models still make errors and they don’t allow users to select which types of toxicity they are interested in finding.

In this project we built multi-headed models by using classifiers like Naive Bayes, Logistic Regression and Long Short Term Memory Networks to detect different types of toxicity like threats, obscenity, insults and identity based hate. We used a dataset of comments from Wikipedia. We also collected Turkish comments from Steam reviews to test our models on Turkish comments as well. We were able to get an accuracy score of 98% and Kaggle score of 0.9812 by using the LSTM for the English data. For the Turkish data we were able to get an accuracy score of 95.43%.

Keywords: Toxic Comment Classification, NLP, LSTM, Sentiment Analysis, Text Classification

This project was inspired by a Kaggle competition. [1]

**1.INTRODUCTION**

The toxic is defined as “an awkward, insolent, impolite or a comment that is likely to make one leave a discussion” [2]. A report made by McAfee in 2014 [3] shows that almost half of the total young population using social media in India experienced toxic comments. The same report [3] suggests that the major reason behind a large number of suicides committed by teens are the toxic comments they receive on social media. Even if the receiver of these comments don’t suicide it usually results in emotional trauma.

The media hosting these online discussions usually fails in filtering these adverse comments because they usually rely on their reporting systems which enables the users to report these comments to be removed from the social platform. [4]. Since the users of such platforms are given the power and freedom to post offensive content, it results in failure to only rely on them to report these comments. These platforms don’t provide any sort of security mechanisms on server side to restrict the toxicity and the damage caused by it. [4]

To solve this problem several machine learning techniques that use supervised or unsupervised methods were implemented before [5] [6] [7] [8] [9]. Apart from these specific examples many researhces were done under the topic of text classification [10] [11] [12] [13] which this project also is part of. Recent models for sentence and document represantation fall into two categories according to the form of input sentence/document: sequence based models and tree structured models [14] [15] [16]. From all these methods the most promising ones were created by using Neural Networks. Recurrent Neural Networks have emerged as a widely used architecture combined with sequence or tree structured models. The most used NNs for text classificiation are Convolutional Neural Nerworks and Recurrent Neural Networks, specifically the Long Short Term Memory NNs.

Based on tests and implementations like these we build different multi-headed models that’s capable of detecting different types of toxicity such as threaths, obscenity, insults etc. The best results were generated by models that use Logistic Regression and LSTM. We used a dataset of comments from Wikipedia’s talk page edits. These models and similar other ones were tested on a dataset of Turkish comments that were hand picked from Steam User Reviews to see the success rates for the Turkish language. This paper will show the steps we followed starting from feature and data exploration to developing the models, followed by comparing the different models we created.

**2. FEATURE EXPLORATION**

We started the project by examining the Wikipedia dataset. It features 159,571 comments which have been labeled by human raters for toxic behaviour [1]. The types of toxicity are:

* Toxic
* Severe Toxic
* Obscene
* Threat
* Insult
* Identity Hate

Each comment in training set had a binary value for each label, a unique comment id and of course the comment text. Apart from this training set we had an unlabeled test set that we used for scores.

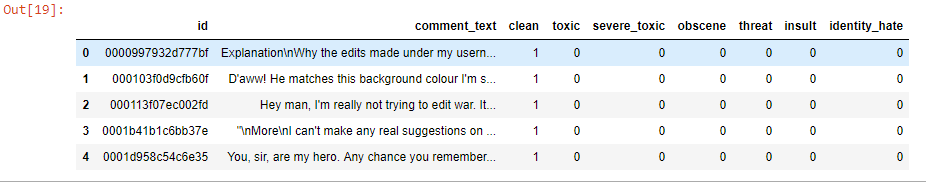


Figure Examples from the training data

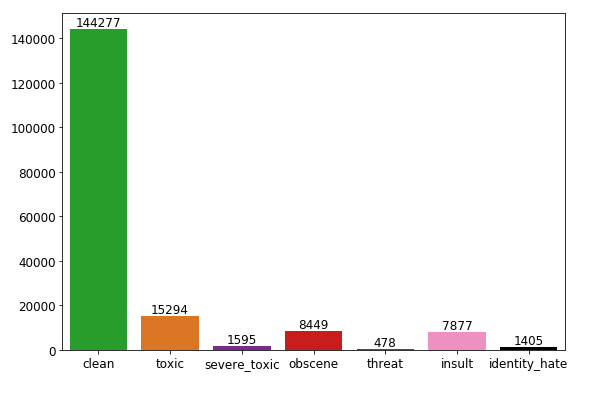


Figure Histogram Graph for the Wikipedia Dataset that shows the total number of each label

As you can see in Figure 1, out of 159,571 comments 144,277 of them are clear. Which means they don’t have any kind of toxicity that this project covers. This shows that our dataset features a heavily-skewed class distrubition. Only 9.5% of all comments are toxic. This percentage is even lower for the other classes. Because of this, raw classification accuracy is not a terribly useful indication of performance. For example consider a dumb classifier that marks all labels as clear. This classifier would have a 90% success rate for the toxic comments even though it would be absolutely useless. Therefore when comparing and discussing the performance of our classifiers we will be referring to either the confusion matrices or the scores that are calculated by Kaggle. According to Kaggle, the Kaggle scores are evaluated on the mean column-wise ROC AUC. In other words, the score is the average of the individual AUCs of each predicted column [1].

After examining the class distribution of the comments we examined the length of the comments.

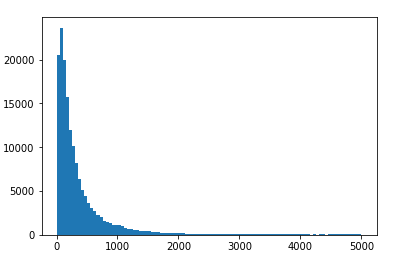


Figure The length of comments(chararacter length)

As you can see in Figure 2 most comments have the character length between 0-1000. The longest one is around 5,000 characters long. Most of the comments are not formal comments but are rather very informal and are unstructured from of grammar.



Figure A word cloud created by some of the words that were taken from comments that are labeled as toxic

**3. WORD VECTORIZATION**

Different methods were used to turn the text data into vectors that can be used to train the models with.

**3.1 TF-IDF**

The first method to vectorize the text data that we used were the TF-IDF.TF-IDF is an information retrieval technique that weighs a term’s frequency(TF) and its inverse document frequency(IDF). Each word or term has its respective TF and IDF score [17]. The TF-IDF algorithm is used to weigh a keyword in any content and assign the importance to that keyword based on the number of times it appears in the document. More importantly, it checks how relevant the keyword is throughout the web, which is referred to as corpus.

For a term t in a document d the weight Wt,d is given by:

The already implemented version of TF-IDF were used in the project by using the Sklearn library of Python [18].

The TF-IDF is used on a word level only. Meaning that the analyzer for the TF-IDF were words not characters. Also the stopwords of the English language were removed from the texts since they don’t bring much meaning for our classifiers to be useful. The one thing that were interesting is that the highest scores that were generated while using TF-IDF were only with 1 grams. This means that most of the toxicity in comments are caused by single words rather than word grams which we thought while starting the project.

|  |  |  |  |
| --- | --- | --- | --- |
| N-Grams | 1 | 2 | 3 |
| Score | 0.9742 | 0.9727 | 0.9720 |

Table Examples of scores that were generated by using TF-IDF with Logistic Regression

**3.2 WORD2VEC**

The second approach to word vectorization in this project were the word2vec models.Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words [19]. It is a breakthrough algorithm as it demonstrated very effectively how to surface seemingly semantic relationships by unsupervised processing of large amounts of text.

Consider word analogy “man is to woman as king is to X” which was famously demonstrated in word2vec. The algorithm is able to come up with an answer queen by simple vector differences. The main idea, called distributional hypothesis, is that similar words appear in similar contexts of words around them.

By splitting the comments in the Wikipedia dataset into a list of words and cleaning them from punctuation and stopwords we build our own word2vec model using the Gensim library of Python [20].

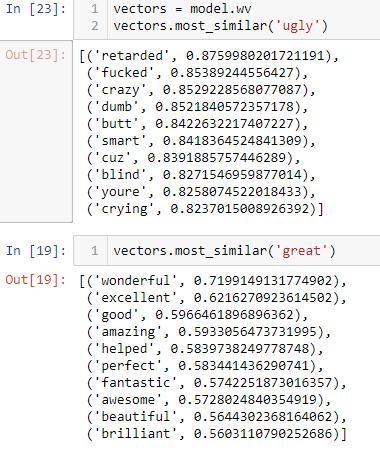


Figure The 10 most similar words to "ugly" and "great" in the word2vec model trained by the words in the training dataset

**3.3 GLOVE**

The last method for word vectorization and word embedding were GloVe. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space [21].

GloVe works similarly as Word2Vec. While Word2Vec is a "predictive" model that predicts context given word, GloVe learns by constructing a co-occurrence matrix (words X context) that basically count how frequently a word appears in a context.

We used the pre-trained word vectors from GloVe. The Wikipedia 2014+ Gigaword 5 file(6 billion tokens,400k vocabulary).

**4. IMPLEMENTED MODELS**

Even though the tests were made with a lot of different models, most of them either did not produce great scores or they simply didn’t work on this project and dataset. This report will cover the 3 main models that we tested.

**4.1 NAIVE BAYES**

An obvious first approach to any text classification problem is multinomial Naive Bayes. Being relatively robust, easy to implement, fast, and accurate; naive Bayes classifiers are used in many different fields. We used the multinomial Naive Bayes classifier from the sklearn library [22]. Normally the multinomial distrubition requires integer feature counts but in practice fractional counts such as TF-IDF also works. So, after vectorizing the data with TF-IDF we trained our model.

The results were not promising. Naive Bayes performed very poorly with a CV score of 0.967. Kaggle score which was explained in section 3 for NB were 0.8538. Like we said before raw accuracy is not really meaningful in this case so we constructed the confusion matrices for each label to see the labelings. The matrices were created by using 5 fold CV.

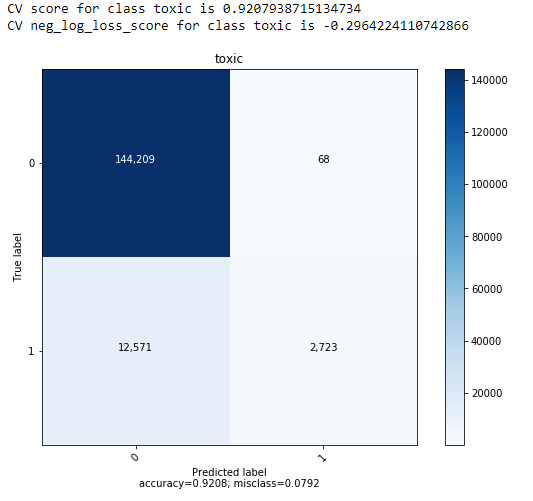


Figure Confusion Matrix for the "Toxic" label

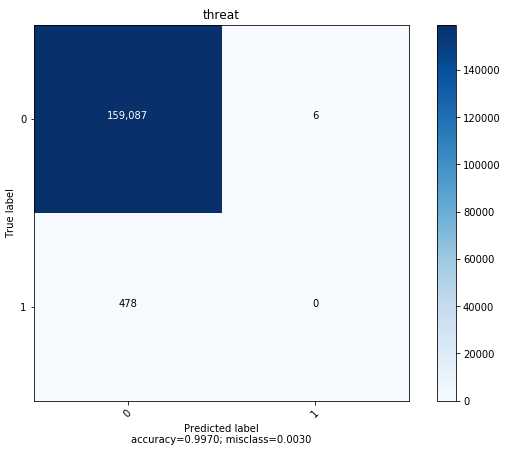


Figure Confusion Matrix for "Threat" label

As you can see NB did not do much better than a dumb classifier that would label all data as clean. The lack of positive labels for toxicity and the document length is playing a huge role here. Normally NB works extremely well on small datasets or short documents [23] [24] But the dataset is apparently too big for NB to handle correctly. Also as we disscussed before the class distribution is heavily-skewed. For example the threat label that is shown in Figure 7 has only 478 positive examples. This means that it is only the 0.003% of the whole data. Later on by changing the parameters we were able to improve the Kaggle score to 0.9010 from 0.8538 but it was still not enough to convince us to spend more time on NB. So we moved on.

**4.2 LOGISTIC REGRESSION**

Our second approach to the problem were Logistic Regression. Logistic Regression has a number of advantages over naive Bayes. The overly strong conditional independence assumptions of Naive Bayes mean that if two features are in fact correlated naive Bayes will multiply them both in as if they were independent which results in overestimating the evidence. Logistic Regression is much more robust to correlated features. If two features *f1* and *f2* are perfectly correlated, LR will simply assign half the weight to *w1* and half to *w2*. Thus when there are many correlated features, LR will assign a more accurate probability than NB [25]. The vectorization for LR were also done by using TF-IDF. The results were much better than the NB as expected. It produced a CV score of 0.98 and a Kaggle score of 0.9763. The one thing that was interesting about LR was the class weights parameter. When the class weights were balanced the overall CV score went down from 0.98 to 0.9720 but Kaggle score stayed the same. Also the confusion matrices gave better results with balanced class weights.

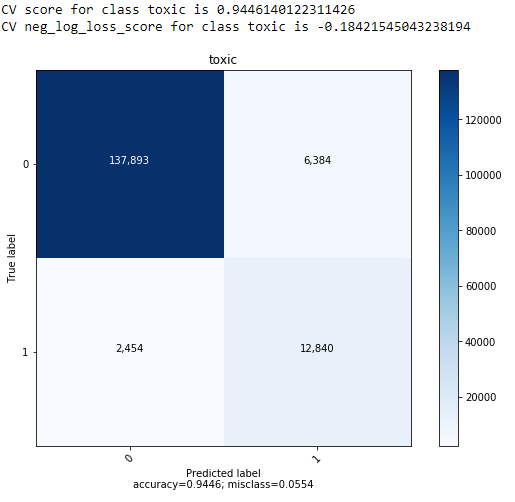


Figure Confusion Matrix for the "Toxic" label. Balanced LR

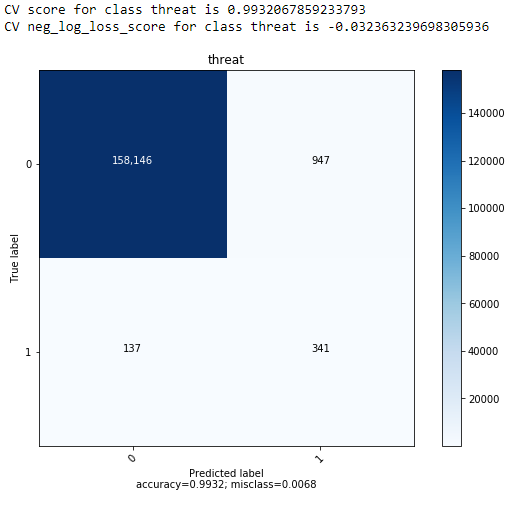


Figure Confusion Matrix for the "Threat" label. Balanced LR

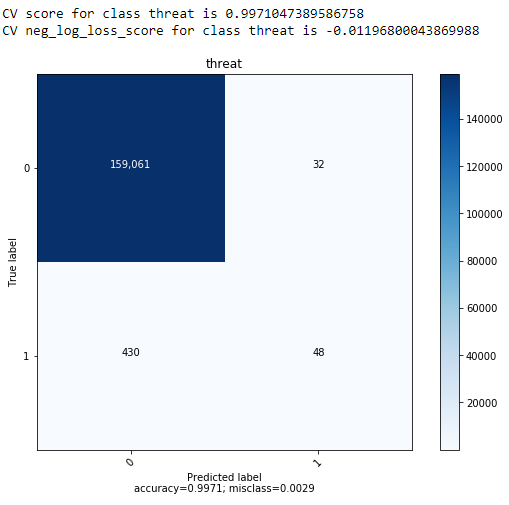


Figure Confusion Matrix for the "Threat" label. Unbalanced LR

We couldn’t get higher scores than 0.9764 with LR with parameter optimization. It definetly provided an actually useful model for predicting the toxicity unlike NB. After LR we moved on to the Neural Networks to get better results.

**4.3 LSTM**

Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used [26].

We used Keras for all the Neural Network layers including the LSTMs [27]. For the word vectorization we used the keras pad\_sequences method followed by glove word embedding. The word embeddings then were given to an embedding layer as weights.

The first model consisted of these layers:

1. Input Layer
2. Embedding
3. LSTM Layer
4. Global Max Pooling 1D
5. Dropout Layer
6. Dense Layer
7. Dropout Layer
8. Dense Layer

This model didn’t use glove embeddings and actually produced worse results than LR. We then started using glove and changed the LSTM layer to Bidirectional LSTM. This model produced much better results. It improved the Kaggle score from 0.9746 to 0.9812. Which was the highest we got.

The last model consisted of these layers:

1. Input Layer
2. Embedding Layer
3. Bidirectional LSTM
4. Global Max Pooling1D and Global Average Pooling 1D (Concatenated)
5. Dense Layer
6. Dropout Layer
7. Dense Layer

This model was tested with max 2 epochs since each epoch took around 4500-5000s to complete. Because of this, using CV with NNs is not really efficient.

The dropout rates that we used were at max 0.2. And since we were expecting 6 outputs for 6 different labels, the last dense layer which is also the output layer had 6 nodes.

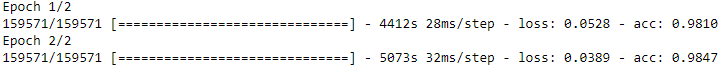


Figure The results of the Neural Network with 2 Epochs

**5. TURKISH DATASET**

We also aimed to predict toxicity for the Turkish Language. However there weren’t any datasets that were created for this before. So we had to create one ourselves. We collected the comments from a platform called Steam [28]. We collected the video game reviews made by customers in Turkish languge. Normally Steam provides an API to automatically collect the reviews. However we couldn’t find any option to collect the reviews that are only written in a specific language. Because of this we collected the comments by hand. Around 500 comments were collected and labeled as toxic or clear unlike the original data which had 6 different labels for toxicity.

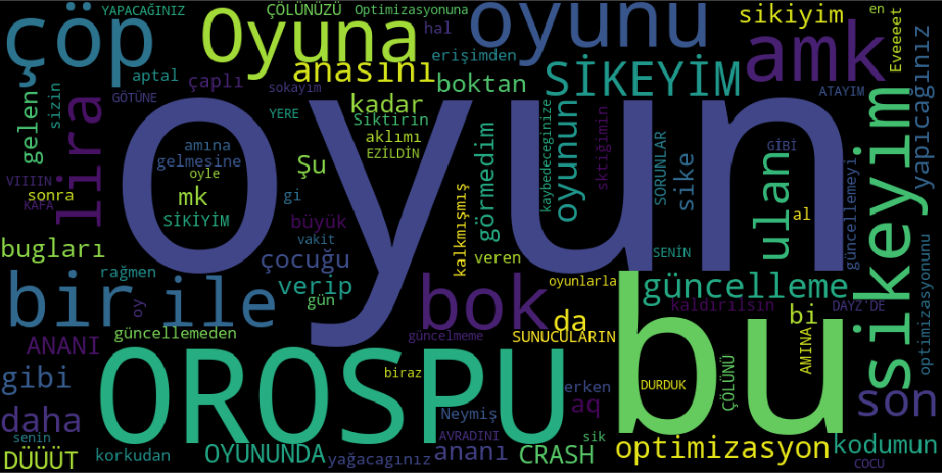


Figure Word examples from Turkish Dataset Toxic Comments

The most difficult problem about the Turkish data is that the Turkish language is an Agglutinating language. Because of this there are a lot of sytax differences between the same words in different comments. For example as you can see in Figure 13 the word “oyun” which means game is written in 3 different ways.

After the data were collected we tested with our LSTM model but it didn’t produce nice results at all. It acted as a dumb classifier and tried to label every comment as clean. This is probably because there weren’t enough data to get the correct weight values. Also we couldn’t use glove with this since it only consists of English words.

The Logistic Regression however were able to come up with a 95.43% accuracy by using 10 fold CV.

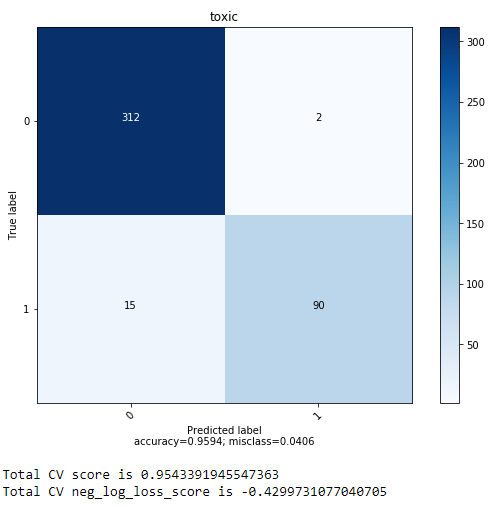


Figure The Confusion Matrix for the Toxic Turkish Data

**6. CONCLUSIONS AND FUTURE WORK**

Some models were built to find toxicity but so far the current models still make errors and they don’t allow users to select which types of toxicity they are interested in finding. In this project we build models to find different kinds of toxicity.

By using Logistic Regression and LSTM we were able to find the toxicity in a Wikipedia dataset that consists of English comments with an accuracy score of 98%. The same models were tested on a Turkish dataset and we were able to get an accuracy score of 95%.

LSTM is definetly promising for text classification as well as CNNs. We weren’t able to test with CNNs in this project but they are another really popular method for text classification. By looking at the projects they are used in, CNN should have good results on this dataset as well. Especially the LSTM and CNN networks together can produce good scores. If the only aim is to improve the Kaggle score then we would recommend using a blending method. This results in overfitting for the problem but it improves the overall Kaggle score.

For word vectorizing different word embeddings and methods can be used to improve the score as well.

# **7.REFERENCES**

|  |  |
| --- | --- |
| [1] | "Kaggle," [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge. [Accessed MAY 2018]. |
| [2] | H. K. S. Z. B. &. R. Hosseini, "Deceiving Google's Perspective API Built for Detecting Toxic Comments. arXiv preprint arXiv:1702.08138.," 2017. |
| [3] | M. Duggan, "Online Harassment," Pew Research Center., 2014. |
| [4] | S. H. &. P. M. Yadav, "An Approach for offensive text detection and prevention in social networks. In Innovations in Information, Embedded and Communication Systems (ICIIECS)," 2015 International Conference on (pp. 1-4). IEEE., (2015, March. |
| [5] | H. P. Patel, "Toxic Comment Tools: A Case Study," ResearchGate, May 2017. |
| [6] | "Jigsaw," Google, [Online]. Available: https://jigsaw.google.com/. [Accessed May 2018]. |
| [7] | "PerspectiveAPI," [Online]. Available: https://perspectiveapi.com/#/. [Accessed MAY 2018]. |
| [8] | M. a. I. G. a. M. M¨uhlh¨auser, "Automatically Assessing the Post Quality in Online Discussions on Software," Ubiquitous Knowledge Processing Group, Division of Telecooperation Darmstadt University of Technology,, Germany. |
| [9] | S. K. T. V. P. P. A. G. V. Spiros V. Georgakopoulos, "Convolutional Neural Networks for Toxic Comment," 2018. |
| [10] | T. Joachims, "Transductive Inference for Text Classification using Support Vector Machines," University of Dortmund, Dortmund, Germany. |
| [11] | A. K. McCallum, "Multi-Label Text Classification with a Mixture Model Trained by EM," Pittsburgh. |
| [12] | L. X. K. L. J. Z. Siwei Lai, "Recurrent Convolutional Neural Networks for Text Classification," Institute of Automation, Chinese Academy of Sciences, China. |
| [13] | R. J. Tong Zhang, "Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings," NY and Beijing, 2016. |
| [14] | C. S. Z. L. F. C. L. Chunting Zhou, "A C-LSTM Neural Network for Text Classification," Department of Computer Science, The University of Hong Kong,School of Innovation Experiment, Dalian University of Technology2, 2015. |
| [15] | R. J. a. T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," Human Language Techonologies: The 2015 Annual Conference of the North American Chapter of the ACL, pages 103-112, 2015. |
| [16] | A. P. J. Y. W. J. C. C. D. M. A. Y. N. C. P. Richard Socher, "Proceedings of Empirical Methods on Natural Language Processing," vol. 1631, p. 1642, 2013. |
| [17] | B. Góralewicz, "Elephate," [Online]. Available: https://www.elephate.com/blog/what-is-tf-idf/. [Accessed May 2018]. |
| [18] | "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/. [Accessed May 2018]. |
| [19] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Word2vec. |
| [20] | "Gensim," [Online]. Available: https://radimrehurek.com/gensim/. [Accessed May 2018]. |
| [21] | R. S. C. D. M. Jeffrey Pennington, "Stanford NLP," [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Accessed May 2018]. |
| [22] | "Scikit Learn Multinomial Naive Bayes," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.MultinomialNB.html. [Accessed May 2018]. |
| [23] | A. Y. a. J. M. I. Ng, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *NIPS 14,* pp. 841-848, 2002. |
| [24] | S. a. M. C. D. Wang, "Baselines and bigrams: Simple, good sentiment and topic classification," *ACL 2012,* pp. 90-94, 2012. |
| [25] | D. J. &. J. H. Martin., Speech and Language Processing, Chapter 7, 2017. |
| [26] | [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1. [Accessed May 2018]. |
| [27] | "Keras," [Online]. Available: https://keras.io/. [Accessed May 2018]. |
| [28] | "Steam," [Online]. Available: https://store.steampowered.com/. [Accessed 2018]. |
| [30] | M. Duggan., "Online harassment.," Pew Research Center., 2014. |