**AGU COMPUTER ENGINEERING**

**CAPSTONE PROJECT**

**FINAL REPORT**

**by**

**Ismayil Allahverdiyev**

**2011051076**

**11/12/2023**

**SUMMARY**

As we know on the internet, people are not necessarily kind to each other. The goal of this project is to create a model to detect the profanity level of the sentences and make it as accurate and precise. I have used different methods to make sure the rate I have reached is the highest I can reach.

**INTRODUCTION**

The dataset used in the project had 6 columns, but only 2 of them, the tweet itself and how offensive it is out with 0 being not offensive, 1 being mildly and 2 being highly offensive, were the most important. The other 4 columns are describing what the 3 contributors’ decisions were. The project consists of 4 stages:

1. Cleaning the data
   1. Removing stop words
   2. Removing unnecessary words
   3. Spellcheck
2. Lemmatization
3. Vectorization
4. Applying different regression models
   1. Logistic Regression
   2. Support Vector Machine Regression
   3. Random Forest regression
5. Deep learning
   1. Label encoding, tokenization and padding
   2. Model
   3. Training and evaluation

**PROJECT**

Since the data is from the internet, a thorough cleanup was necessary which consisted of 3 parts: Removing stop words, removing unnecessary words, and fixing the spelling. After removing the stop words and numeric values with the help of the Natural Language Toolkit, with Gokhan Bakal’s advice the next step was removing the usernames from the data which usually came at the start of each row of data and repeated 1(a) again. Once the cleanup was finished, using the SpellChecker package from Python resulted in a much cleaner dataset.

By finishing part 1 and obtaining a cleaner and more focused dataset, it was time to lemmatize words to get to the root of them. Again, by using the Natural Language Toolkit lemmatization of the dataset was completed, meaning stages 3 and 4 can be implemented.

After careful discussion with Gokhan Bakal and considering the length of the sentences in the dataset, we decided to separate the sentences into groups of unigrams, bigrams, trigrams, and the combinations resulting in 6 different ways of vectorization. Shuffling the dataset 10 times and applying TfidfVectorizer to get the combinations above resulted in 60 combinations to work with, so the next step was ready to be started. 5-fold cross-validation was implemented in each case scenario.

**TRAINING MODELS AND RESULTS**

Three different models were implemented, namely Logistic regression, Support Vector Machine, and Random Forest Regression with 20% of the dataset being used as a test. For each of these models, cross-validation scores, mean accuracy, and standard deviation of accuracy were calculated to make sure the models were running well and fair in all circumstances. After discussions with Gokhan Bakal, precision, recall, and f1 scores were calculated for each case to describe how precisely the models worked in numbers.

After careful inspections, it was obvious that the combinations that included unigrams did considerably better than the rest with around 90.5% accuracy whereas usage of individual cases of bigrams and trigrams without the unigrams resulted in an average accuracy of 80% and 78% respectively. Our best-case scenarios (the ones including unigrams) had an average of 90% in cross-validation scores and 0.003% standard deviation of accuracy on average proving that our dataset was well-distributed. In cases of (2, 2) and (2, 3) ngram combinations cross-validation scores were lower being around 81% and having 0.004% standard deviation in accuracy, whereas (3, 3) combinations were the lowest with 78.5% in cross-validation scores, and 0.001% standard deviation in accuracy.

In terms of regression models, all of the models did pretty well in the best-case scenario with 90% accuracy scores while logistic regression was slightly lower than the rest. Moreover, in terms of average precision, all of the models nearly performed the same around 89% - 90% meaning it was at an acceptable rate. All the training models were sensitive which is proven by above 90% recall scores and above 89% f1 scores. Keep in mind that this data is only from the best-case scenarios in which the models trained using all the combinations that included unigrams, and of course the best one was the (1,3) combination as expected.

**DEEP LEARNING**

After successful training with the regression models above, deep learning was applied to the dataset. After encoding and vectorizing the data, padding was applied where the maximum length was set to be 15. In the next step, the CNN model was created where we applied embedding, Conv1D to perform convolutions on the dataset, a 50% dropout rate to prevent overfitting, and other essentials. Adam was selected to perform neural network optimization on the CNN model where the learning rate was set to be 0.0001.

With an early stopping where the patience was set to be 5 model was fit with 20 epochs. The accuracy was not as positive as we hoped, however, considering our dataset consisted of sentences taken from social media 77.28% is not as bad since it can be improved with a larger dataset.

**CONCLUSIONS**

After careful calculations and observations, the training models were 90% accurate, which can still be improved if the dataset was enlarged enough. Further research is needed to get better accuracy. The Python notebooks are accessible in the references, where you can see each step taken to train the models.

**REFERENCES**