ML Project Report

- Team MaRVeL

Members -

- Aditya Vardhan (IMT2019003)
- Archit Sangal (IMT2019012)

We did the work in attempts. **Each attempt** can be considered as a **version control or the new things** we tried. Please read through the points under various attempts that will be a good short guide for all the methods we tried for solving and getting a better score.

Information about the dataset

There were no null values or duplicated rows. The dataset is an unbalanced dataset. It has around 40000 data for positive classification and about 640000 data for negative classification. Due to this, f1-score was difficult to improve. As the data was all in string, we needed to convert it to numeric data so that the machine can process it. And we can apply different machine learning methods on it.

Exploratory data analysis

- **Data preprocessing**
  Basic Preprocessing like-
  - checking for null values or duplicated rows.
  - Removing multiple spaces.

  We tried different techniques on the data like-
  - removing punctuation
  - changing the words to lower case
  - Stemming
  - Lemmatization.
  - Removing stop words

But none of these improved our score.

- **Data Charts and Visual Techniques used**

We used word cloud for getting insights of data. We also used confusion matrix to understand the reason of getting not a good score and to improve the score.

- **Feature Engineering**

We used feature engineering over the traditional **WordToVec** and **TF IDF** score. We found the vectors to words in each sentence. Now, we needed a Sentence to vector kind of thing. Basically, a vector that represents the sentence. For this, we first took the average of the vectors of words in sentence. But each **word has different weightage**, so we **multiplied TF IDF score** of the word with its vector and then took the average.

Model Selection

- Logistic regression
- Bernoulli Naive Bayes
- Perceptron
- Linear SVC
- XGBoost
- SVM model with polynomial and rbf kernels
- RandomForest
- AdaBoost

As our codes are in different attempts, I am writing the things and approaches I tried in the order I tried them under different folders.

Attempts and Approaches (What worked and what didn't)

Attempt 1

1. Preprocessing:
    a. For vectorizing words:
        i. Bag of words
        ii. TF IDF
    b. Split the train.csv train and test data

2. Model(s) used:
    a. Logistic regression (solver= 'liblinear'):
        i. Prediction: predict_proba
            1. Threshold: 0.225

3. Confusion matrix:
    a. True neg = 90.24%
    b. True pos = 4.72%

c. False neg = 1.5%
d. False pos = 3.54%

Attempt 2

1. Preprocessing
    a. Data balance check
    b. Balancing the data (Manual) - **used SMOTE and duplication of data**
        i. under sampling of majority class
        ii. over sampling of minority class **for various ratios**
    c. Cleaning data
        i. Removing !,? Etc.
        ii. Converting sentences to lower case
    d. Stemming and Lemmatization

2. Model(s) used:
    a. Logistic regression (solver= 'liblinear'):
        i. Prediction: predict_proba
            1. Threshold: **0.5**

Attempt 3

1. Manually balanced data was used only to generate new vocabulary by analysing the graph given by "word cloud".
2. In the sentences, only those words were kept which were included in our generated vocabulary.
3. For training, the model original data was used, NOT the balanced data.

4. Confusion matrix for train data:
    a. True neg = 46.38%
    b. True pos = 47.34%
    c. False neg = 2.48%
    d. False pos = 3.81%

5. Confusion matrix for test data:
    a. True neg = 86.07%
    b. True pos = 4.56%
    c. False neg = 1.32%
    d. False pos = 7.74%

Attempt 4

1. Optimized hyperparameters and the threshold for “predict_proba”.
2. We tried various permutations and combinations of the preprocessing methods used in above attempts.

Attempt 5

1. Removed the part where we modified sentences by using the generated vocabulary in attempt 3.

2. Changed the default value of the "ngram_range" parameter in the vectorize functions of the "Bag of Words" and "TF IDF" methods to *(1,3)*, which improved our score. This is the thing that worked for us.

3. Models added:
    a. Bernoulli Naive Bayes
    b. Perceptron

Attempt 6

1. Added the part where we modified sentences by using the generated vocabulary in attempt 3; that we removed in attempt 5.

2. Tried “Word2Vec” but it gave poor score (around 0.5)

3. Added models:
    a. Linear SVC: Poor score (around 0.4)
    b. XGBoost: Poor score (around 0.4)
    c. SVM model with polynomial and rbf kernels: took too long to run

4. Confusion matrix for train data:
    a. True neg = 86.44%
    b. True pos = 7.75%
    c. False neg = 2.98%
    d. False pos = 2.82%

5. Confusion matrix for test data:
    a. True neg = 90.32%
    b. True pos = 4.13%
    c. False neg = 2.03%
    d. False pos = 3.52%

Attempt 7

1. Removed the part where we modified sentences by using the generated vocabulary in attempt 3.

2. Added models:
    a. RandomForest with the best result for tree_depth 25 (score around 0.4)
    b. AdaBoost with the best result for “n_estimators = 500” (score around 0.5)

3. Tried Word2Vec with TF IDF:
    a. With “ngram_range= (1,3)”: We waited for around 5 hours, but it could process only up to 90,000 rows. So, we gave up.
    b. With default value of “ngram_range” i.e. (1,1): gave poor score.

4. Confusion matrix for test data:
    a. True neg = 86.72%
    b. True pos = 3.14%
    c. False neg = 3.02%
    d. False pos = 7.11%

Attempt 8 and 9

1. Tried to optimize hyperparameters but to no avail. Hyperparameter of model and calculating the optimum threshold.

Attempt 10

1. Added the part where we modified sentences by using the generated vocabulary in attempt 3; that we removed in attempt 7.

2. Removed the common words from sentences of minority and majority class. The score was around 0.6, which couldn’t beat our high score.

3. Confusion matrix for train data:
    a. True neg = 86.44%
    b. True pos = 7.75%
    c. False neg = 2.98%
    d. False pos = 2.82%

4. Confusion matrix for test data:
    a. True neg = 90.32%
    b. True pos = 4.13%
    c. False neg = 2.03%
    d. False pos = 3.52%

Attempt 11
Tried to implement a “sequential model” but each epoch was taking about 4.5 hours and for effective results we needed to have at least 5 epochs. So, we couldn’t pursue it further.

Attempt 12
This was our best submission. We did try to use **ngram_range = (1,4)** but it gave me relatively bad results, so I didn’t submit it on Kaggle. But on the last day, we did try to optimize it and submit it. We got our best results. We could have tried for ngram_range = (1,5) but there wasn’t sufficient time and memory. The **threshold** which gave us best results is **0.132**.

In short, nothing except the very basic bag of words and Logistic regression worked for us. All the other approaches didn’t give us good results.

Unbalanced dataset was the biggest problem which we faced. None of the different types of SMOTE worked. None of the different type of Upsampling or undersampling worked.

Result and Rank

We achieved a **f1 score of 0.63049**, and we were able to achieve **rank 1** before and after including the test results among all teams that participated in the contest **‘AskReddit Troll Question Detection Challenge’.**

#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	MaRVeL			0.63879	19	3d

Folder Structure

1. Complete Code - Contains the attempts in the respective folders and models are separated out.
2. Best Solution - This folder contains the code that gave the best solution on the leader board.
3. Final Slides
4. This Report