

**IRE Major Project Report**

**Identification and Classification of Offensive Tweets.**

**(SemEval19: OffensEval)**

**Submitted by : Supervisor :**

Group-18 Vivek Anand

* Archit Kumar ( 2018201051 )
* Arnav Kapoor
* Neeraj Barthwal ( 2018201069 )
* Shruti Chandra ( 2018202010 )

# 

# 

# 

# 

# **Abstract**

Offensive language is pervasive in social media. Individuals frequently take advantage of the perceived anonymity of computer-mediated communication, using this to engage in behaviour that many of them would not consider in real life. Online communities, social media platforms, and technology companies have been investing heavily in ways to cope with offensive language to prevent abusive behaviour in social media.

One of the most effective strategies for tackling this problem is to use computational methods to identify offense, aggression, and hate speech in user-generated content (e.g. posts, comments, microblogs, etc.). Thus we aim to create a system for identification and classification of offensive tweets . It involves the following three subtasks which are hierarchical in nature :

## Subtasks

Sub-task A: Offensive language identification; [Offensive: OFF, Not Offensive: NOT]  
 Sub-task B: Automatic categorization of offense type[Targeted: TIN, Untargeted: UNT]  
 Sub-task C: Offense target identification. [Individual: IND, Group: GRP, Other: OTH]

*Dataset used : OLID v1.0 dataset*

**Related Work**

* Seganti, A., Sobol, H., Orlova, I., Kim, H., Staniszewski, J., Krumholc, T., & Koziel, K. (2019). NLPR@SRPOL at SemEval-2019 Task 6 and Task 5: Linguistically enhanced deep learning offensive sentence classifier.
* Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval).
* Frisiani, N., Laignelet, A., & Güler, B. (2019). Combination of multiple Deep Learning architectures for Offensive Language Detection in Tweets.
* Watanabe, H., Bouazizi, M., & Ohtsuki, T. (2018). Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection.
* Rangkuti, R., ., Z., & Lubis, A. (2019). Hate Speech: The Phenomenon of Offensive Language.

# 

# **Architecture**

## Preprocessing Steps

The first step was to clean and process the tweets so as to remove the noise and other unnecessary words, which is needed for any model and would help in providing better results. The steps used were as follows :-

* Decontraction - (can’t -> can not , won’t -> will not )
* Stop Words Removal
* Case folding
* Tokenization and Lemmatization

## Feature Engineering

We added new columns i.e extracting features from the original data to improve the prediction of our models. The features added were as follows :-

* Number of Hashtags
* Number of User Mentions
* Length of Tweet
* Number of URLs and Emojis.

**Justification** - We analysed the results with and without the additional features (# of hashtags , # of mentions , # of URls). For the best model of Task A

F1-Score with additional features - 0.712   
 F1-Score without additional features - 0.699

Across all subtasks/features/models a general positive improvement in F1-Score was seen with additional features.

## Representations Formed

We need to form a representation of the tweets (that is the mapping from textual data to real valued vectors) before sending it to our model. The various representations used were as follows :-

* Bag of Words
* TF-IDF
* GloVe Vectors (300 dimensions)

## Models Used

The models used for the classification tasks included the basic machine learning models (LR, Naive Bayes and SVM), and deep learning models (CNN and RNN).

* Logistic Regression
* Naive Bayes
* SVM
* CNN
* RNN (LSTM and BiLSTM)

# Detailed Architecture for Deep Learning Model

# Embedding Layers : RNN (LSTM and BiLSTM)

### **Making data ready for LSTMs and BiLSTMs.**

GloVe vectors are used to generate a 300-dimentional embedding for our words in the vocabulary. The tweets are first tokenized and then an embedding matrix is obtained using the pretrained glove model.

This matrix is then fed to the first layers of LSTM using an embedding layer and we do not learn word embeddings while training the network.

### **Input to LSTMs and BiLSTMs**

The word embedding layer expects input sequences to be comprised of integers, usually called as word encodings, which represent a sentence, tweet in our case. We use Tokennizer's text\_to\_sequences() to achieve this. Thus, each tweet is fed as a sequence of numbers when training, the appropriate embeddings corresponding to the word is picked from the embedding matrix in the model layer.

Tweets however, are unequal in length and the text\_to\_sequences() function generates array of number corresponding to each tweet which are unequal in size. Since, keras works in batches, we pad the tweets to make them equal in length. This is the final input - padded\_tweets\_train, in our case, on which the model is trained.

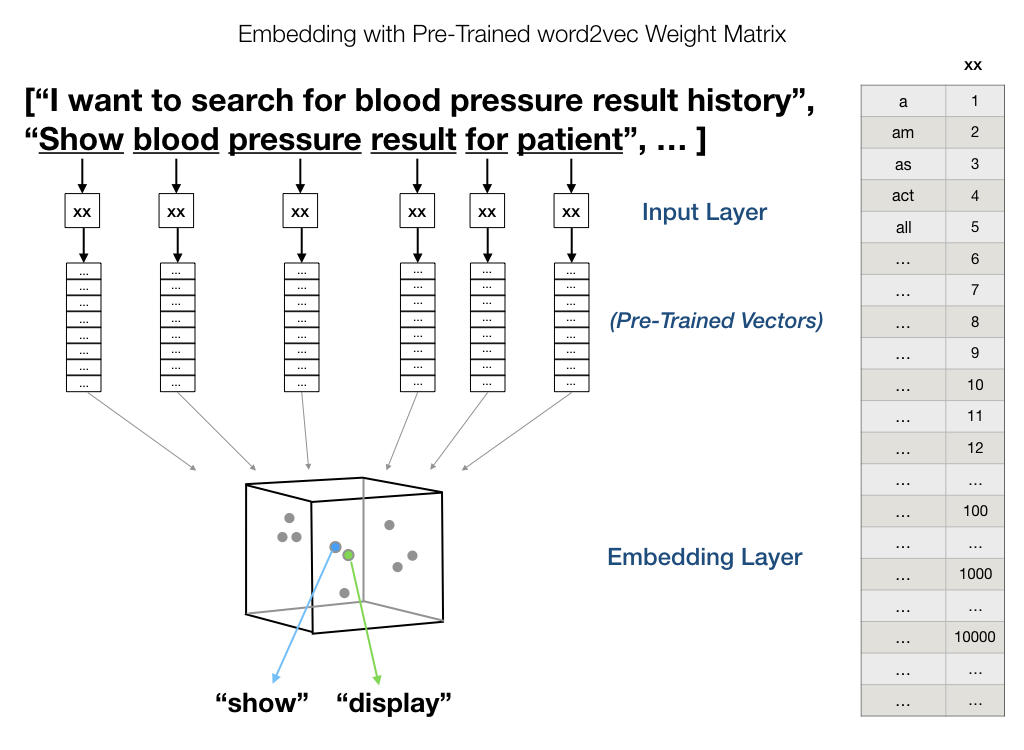


Image References : <https://medium.com/@JMangia/coreml-with-glove-word-embedding-and-recursive-neural-network-part-2-ab238ca90970>

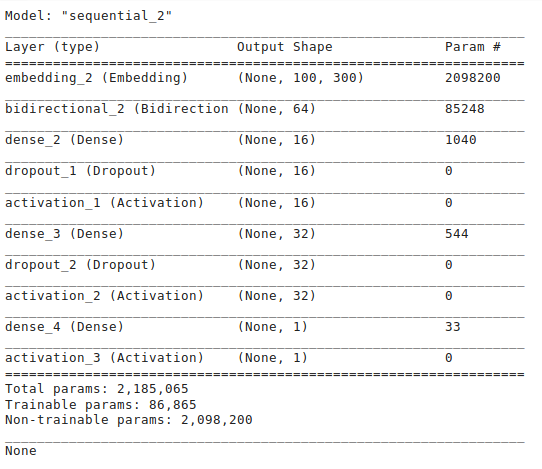
### **Final Layer**

The network differs in architecture at the final layers for sub tasks A, B and C.

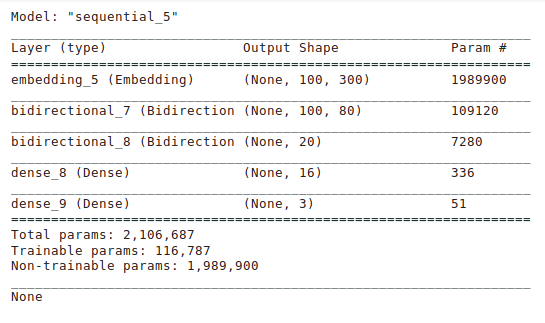
For task A and B we use a **Dense Layer with one output unit** and **sigmoid activation** and **binary crossentropy** as **loss function**. This is because these tasks involve a binary classification - OFF/NOT\_OFF and Targeted or Untargetted.

The final layer for sub task C is a Dense Layer with 3 units, this layer has **softmax as activation** and the **loss function used is categorical cross entropy**

**Architecture for Task B BiLSTM**

****

**Architecture for Task C BiLSTM**

****

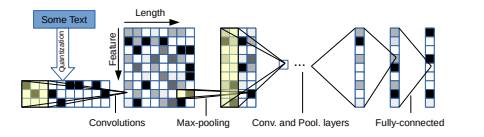
The first layer is for input embedding. Then follow two layers of bidirectional LSTM. The first one has 40 LSTM units, and the second one has 10 LSTM units. The output from last LSTM layer goes to two dense layers. First one of 16 units and the last being the final output layer. We have used softmax activation at the output layer.

## Character-level Convolutional Network

### **Char-level CNN Idea**

The next model we tried was character based CNN since texts in tweet corpus is a bit different than any other regular text corpus. In tweets, word shorthands e.g. ‘FYKI’ instead of ‘for your kind information’, repeating characters e.g. ‘yaaayyyyyy’ to express feelings, emoticons using symbols etc are used. These are not general English words and these usages may differ from person to person. As for example, ‘yaaaayyyyy’ and ‘yaayyy’ both are the same word. So character level model would work better than word level model.

1. **Input to char-CNN**Char-CNN accept a sequence of encoded characters as input. The encoding is done by prescribing an alphabet of size m for the input language, and then quantize each character using 1-of-m encoding (or “one-hot” encoding). Then, the sequence of characters is transformed into a sequence of such m sized vectors with fixed length l0. Any character exceeding length l0 is ignored, and any characters that are not in the alphabet including blank characters are quantized as all-zero vectors. The character quantization order is backward so that the latest reading on characters is always placed near the beginning of the output, making it easy for fully connected layers to associate weights with the latest reading .
2. **Char-CNN Architecture**

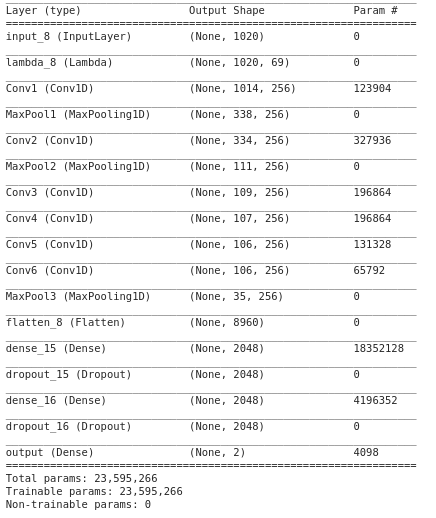


The alphabet used in all of our models consists of 70 characters, including 26 english letters, 10 digits, 33 other characters and the new line character. The non-space characters are:

abcdefghijklmnopqrstuvwxyz0123456789 -,;.!?:’’’/\|\_@#$%ˆ&\*˜‘+-=<>()[]{}

The embeddings are then passed to the CNN network whose architecture is described below.

We also insert 2 dropout modules in between the 3 fully-connected layers to regularize. They have dropout probability of 0.5 .



### 

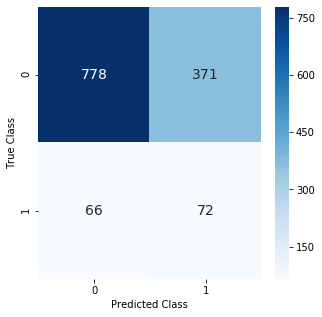
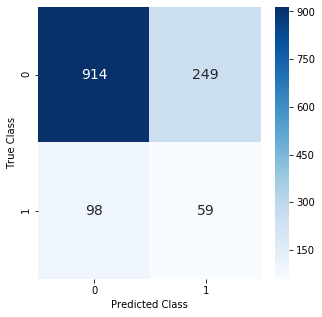
### **About Data Augmentation :**

Many researchers have found that appropriate data augmentation techniques are useful for controlling generalization error for deep learning models. These techniques usually work well when we could find appropriate invariance properties that the model should possess. In terms of texts, it is not reasonable to augment the data using signal transformations as done in image or speech recognition, because the exact order of characters may form rigorous syntactic and semantic meaning. Therefore, the best way to do data augmentation would have been using human rephrases of sentences, but this is unrealistic and expensive due to the large number of samples in our datasets.

On augmenting data by basic methods like Synonym Replacement:

Random Insertion, Random Swap , Random Deletion we artificially created data for improving our DL models.

We observed that the F1 scores , Confusion Matrix (shown for Subtask-B) the results **actually worse on augmentation ,** as contextual information was probably lost.

i) For Augmented Data ii) Original Unaugmented Data

## Experiments:

### **Using stratified K-Fold :**

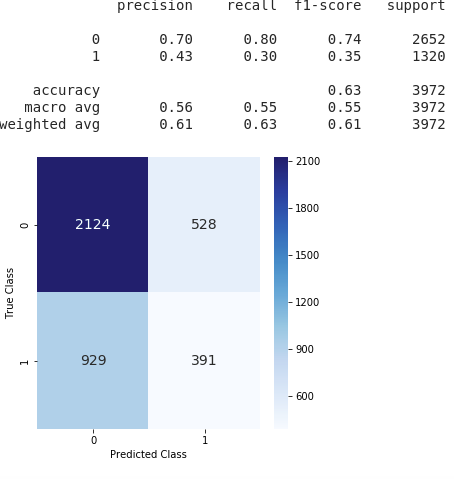
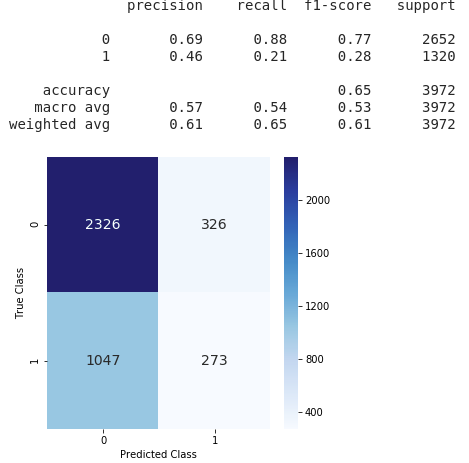
Stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole. For example in a binary classification problem where each class comprises 50% of the data, it is best to arrange the data such that in every fold, each class comprises around half the instances.

It is used to find the best result from model.

1. **Handling Data-imbalances**

* **Using class weight** :

Weight balancing balances our data by altering the weight that each training example carries when computing the loss. Normally, each example and class in our loss function will carry equal weight i.e 1.0. But sometimes we might want certain classes or certain training examples to hold more weight if they are more important.



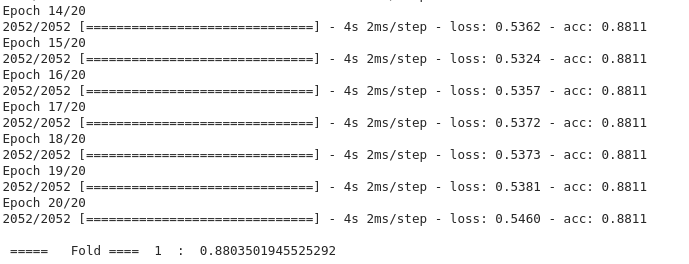
**Without class weights With class weights**

In above results class weights shows a little improvement in the precision , recall and f1 score of minority class .

* **Using Focal-loss :**

The focal loss is designed to address class imbalance by down-weighting inliers (easy examples) such that their contribution to the total loss is small even if their number is large. It focuses on training a sparse set of hard examples.

In our architecture focal-loss lead to vanishing gradient problem . Model stops learning after a certain number of steps . We can mitigate this issue by modifying our architecture , but since we are taking our architecture from an experimental set up as defined in research paper ( Crepe -> char-level CNN ) we preferred to stay with the original architecture . This loss function might be helpful in improving the model performance .



**Result :** Due to very small data size , none of the above approaches give any significant improvement in the final metrics ( precision , recall , f1\_score ) .

# Evaluation Mechanism & Results

Since the problem statement is from the SemEval contest , the evaluation metric is very well defined , we used the same parameters (F1-Score) as used in the contest.

* **Precision:** the ability of a classification model to identify only the relevant data points.
* **Recall:** the ability of a model to find all the relevant cases within a dataset.
* **F1-score:** F1-score is the harmonic mean of precision and recall taking both metrics into account in the following equation:
* **F1 (macro average)** : A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally).
* **F1 (micro average) :** A micro-average will aggregate the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable if there is class imbalance.
* **F1 (weighted average) :** F1-weighted average alters ‘macro’ to account for label imbalance

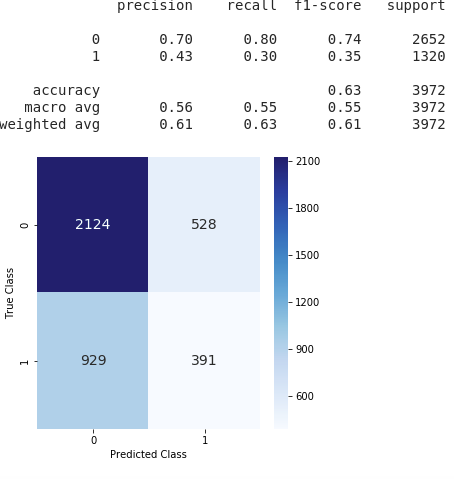
# Results

## **Task A**

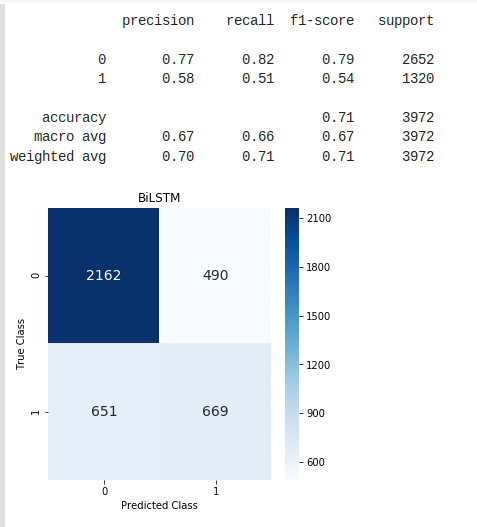
F1- Macro Scores

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bag of Words | Tf-Idf | Avg Word2Vec |
| Logistic Regression | 0.709 | 0.703 | 0.649 |
| Naive Bayes | 0.705 | 0.658 | 0.591 |
| SVM | 0.700 | 0.702 | 0.643 |
| CNN | 0.55 | | |
| RNN | 0.67 | | |

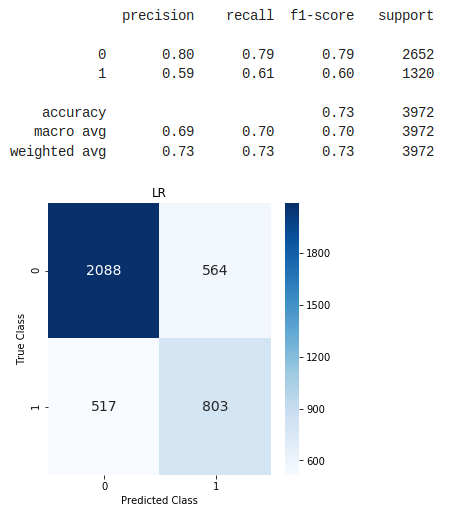
Results for CNN



Results for RNN



Best Baseline ML Model -> Logistic Regression + BOW

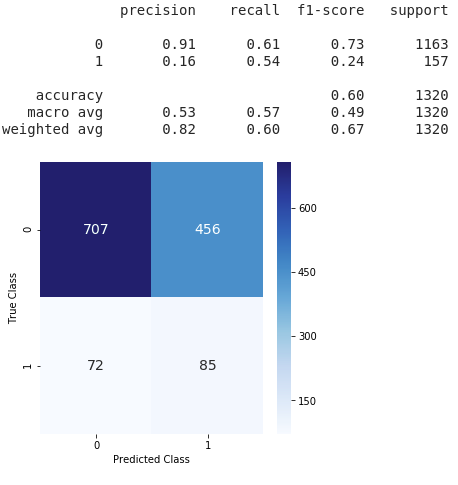


**Task B**

F1- Macro Scores

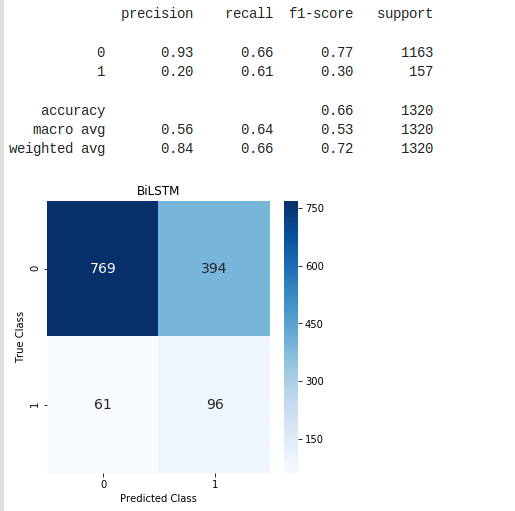
|  |  |  |  |
| --- | --- | --- | --- |
|  | Bag of Words | Tf-Idf | Avg Word2Vec |
| Logistic Regression | 0.53 | 0.5613 | 0.565 |
| Naive Bayes | 0.542 | 0.597 |  |
| SVM | 0.56091 | 0.55884 | 0.538198 |
| CNN | 0.49 | | |
| RNN | 0.534 | | |

Results For CNN

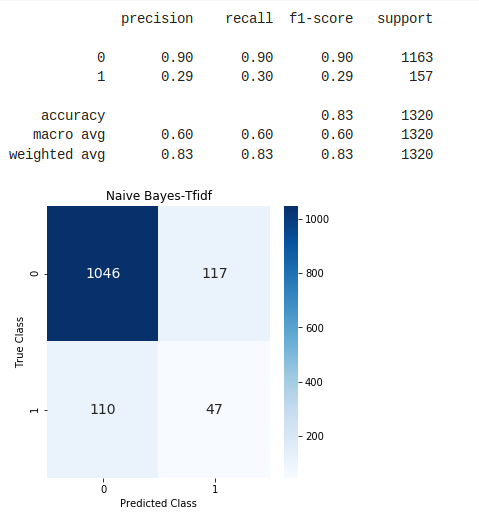


## 

Results for RNN (BiLSTM)



Best Baseline ML Model -> Naive Bayes + Tf-IDF

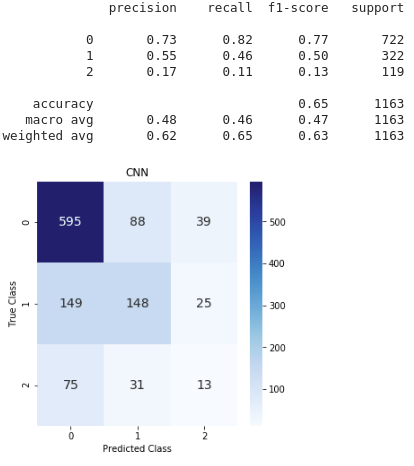


**TASK C**

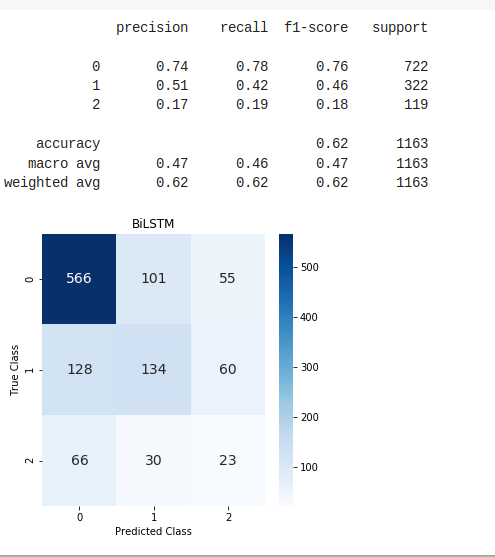
F1- Macro Scores

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bag of Words | Tf-Idf | Avg Word2Vec |
| Logistic Regression | 0.70991522 | 0.72871 | 0.6000737 |
| Naive Bayes | 0.6603611 | 0.6595012 | 0.62080825 |
| SVM | Linear: 0.60705  Rbf: 0.67411 | Linear: 0.63628  Rbf: 0.59587 | Linear : 0.5932  Rbf: 0.5666 |
| CNN | 0.599 | | |
| RNN (BiLSTM) | 0.621 | | |

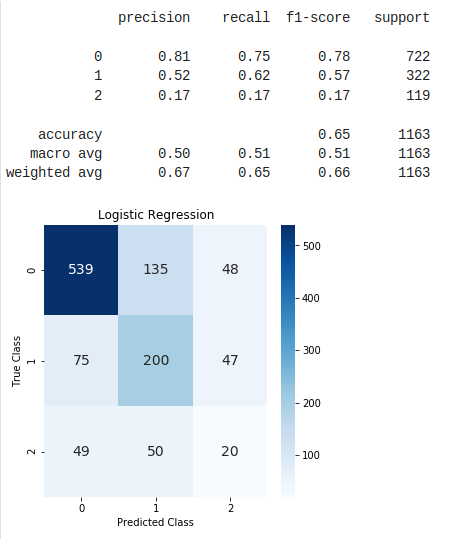
Results for CNN



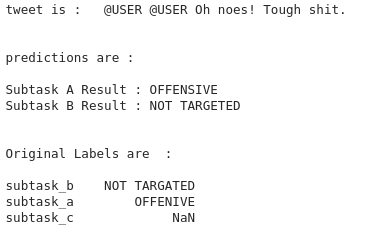
## Results for RNN (BiLSTM)



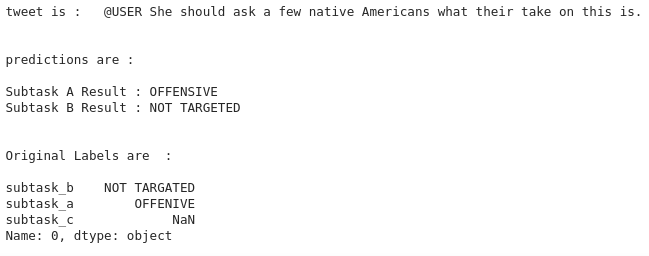
Best Baseline ML Model -> Logistic Regression + Tf-IDF



# FINAl Predictions :

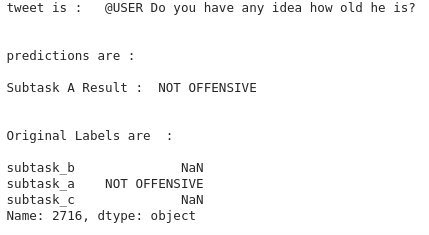


2.



3.

# 4.



# 

# 

# Analysis

## Misclassification examples :

## Task A :

Red: original not offensive (0), classified as offensive (1)   
Green: original offensive (1), classified as not offensive (0)

1. @USER @USER But the liberals do
2. @USER @USER Sshh 🙊 she is lying😏😅😭😂
3. @USER Goodell is the worst commissioner of any sport from any time in history
4. @USER I'm frothing over all of it so far.. 🤤🤤 ..the goodest shit. 😏

## Task B :

Red: original Targeted insult (0), classified as untargeted (1)   
Green: original untargeted (1), classified as Targeted insult (0)

1. @USER @USER If this yutz was any dumber she'd have to be watered twice a week!
2. @USER His mouth goes to one side all smirky and shit and I don’t like it.
3. @USER @USER Son please don’t make me cry today because I never saw this shit
4. @USER He tihnks he is telling the big lie.

## Task C :

Red: original individual insult (0), classified as group (1)

Cyan: original group insult (1), classified as other (2)

Orange: original other insult (2), classified as group (1)

Purple: original group insult (1), classified as individual (0)

Light Blue: original other insult (2), classified as individual (0)

Green: original individual insult (0), classified as other (2)

1. @USER Yeah. That's kinda the fashy thing right now. The Proud Boys scumbags wear tshirts saying Pinochet did nothing wrong" And Antifa are the ones who are violent? Please. It's been an actual fascist every time this has been brought up in my experience. Like dude here, who stopped.."

2. @USER @USER I think the pope and some others should be prosecuted for covering up and protecting child rapists!

3. @USER @USER Beware of rightist false flags like the narrative of the OK sign being a white nationalist dog whistle. It's not. #TheResistance #Resist #BashTheFash #Antifa #SmashFascism URL

4. @USER Its clear you wouldn't listen anyways since you are more interested in arguing over the non existent racism of that tweet rather than calling out the ignorant white girl on display

1. @USER @USER Mine go drafts too. @USER stop this crap I'm gathering proof your censoring conservative .
2. @USER Zimmerman belongs in prison for killing Travon Martin. Florida needs to change its gun control laws.

Misclassified examples shows that some samples are misclassified because they were labeled wrong .

**Code link to the baseline implementations**

**Link to the webpage of the project.**

<https://arnavkapoor.github.io/>