Machine Learning Implementation for Text Classification using MediaEval Dataset

Jakub Markiewicz

mjmarkiewicz@outlook.com

January 5, 2024

1. Introduction

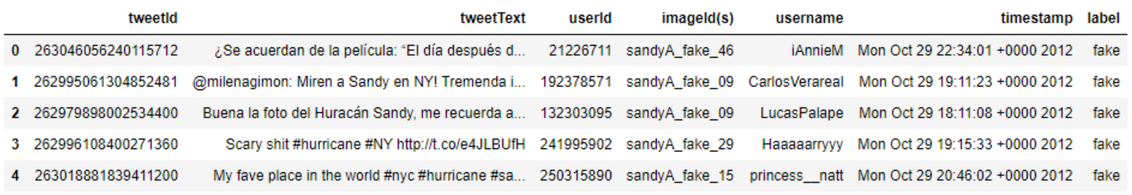
The rate of at which information is shared online grows exponentially as social media websites such as X.com (Twitter) gain more users. Software that creates spam, botting and fake media is becoming more documented and easily accessible to the typical person. With malicious intent, it is possible to utilise such tools to, at the least, clutter the feed of the user and, at the worst, create havoc over artificial news and cause political indoctrination. Increasingly, with the ease of tools such as Adobe's AI photo editing, it is hard to distinguish what is real and what is fake on the internet.

Nevertheless, supervised Machine Learning (ML) algorithms have proven to be effective at analysing patterns in multimedia and being able to classify them as either fake or real. The importance of having robust and reliable detectors will grow as the line between these two labels continue to blur.

This project aimed to demonstrate the complete implementation of such a classifier algorithm on the MediaEval (2015) text dataset. The report includes an analysis of the dataset in Chapter 2, the pipeline design (preprocessing & dimensionality reduction, feature selection, and algorithm training) in Chapter 3, and a performance evaluation of the learner in Chapter 4. Finally, a summary of the findings and will be provided in Chapter 5.

2. Exploring The Data

The MediaEval dataset is a text file composed of tweets related to certain events. Each tweet is labelled as either fake or real. Additional information in the datasets is shown:



	tweetId	tweetText	userId	imageId(s)	username	timestamp	label
0	263046056240115712	¿Se acuerdan de la película: "El día después d...	21226711	sandyA_fake_46	iAnnieM	Mon Oct 29 22:34:01 +0000 2012	fake
1	262995061304852481	@milenagimon: Miren a Sandy en NY! Tremenda i...	192378571	sandyA_fake_09	CarlosVerareal	Mon Oct 29 19:11:23 +0000 2012	fake
2	262979898002534400	Buena la foto del Huracán Sandy, me recuerda a...	132303095	sandyA_fake_09	LucasPalape	Mon Oct 29 18:11:08 +0000 2012	fake
3	262996108400271360	Scary shit #hurricane #NY http://t.co/e4JLBUfH	241995902	sandyA_fake_29	Haaaaarryyy	Mon Oct 29 19:15:33 +0000 2012	fake
4	263018881839411200	My fave place in the world #nyc #hurricane #sa...	250315890	sandyA_fake_15	princess__natt	Mon Oct 29 20:46:02 +0000 2012	fake

Figure 1: Five Entries of Training Data; 7 Columns: tweetId, tweetText, userId, imageId(s), username, timestamp and label.

In total, there are 14277 and 3755 tweets in the training and test dataset respectively. This is a classic 80/20 data split which is optimal for training an ML algorithm [1]. A further investigation into the ground-truth labels of the data can yield the distribution of fake and real tweets, shown in Figure 2.

In the training data, there is a 34/66 real to fake ratio of tweets. Similarly, this is 32/68 in the testing set. This implies a data bias in the larger number of fake tweets within the dataset. Moreover, this may cause the learner to be better at correctly flagging fake tweets rather than real ones (overfitting). Fortunately, due to the similarity of the training and testing data, the effect of this is not severe.

In both datasets, there is a limited number of events that the tweets discuss. The summed occurrences of these events is shown in Figure 3. In the training data, there is a large bias towards the "sandyA" and "sandyB" hurricane event with 12318 out of 14277 occurrences. In the testing data, a similar bias exists towards the "syrianboy" shootout

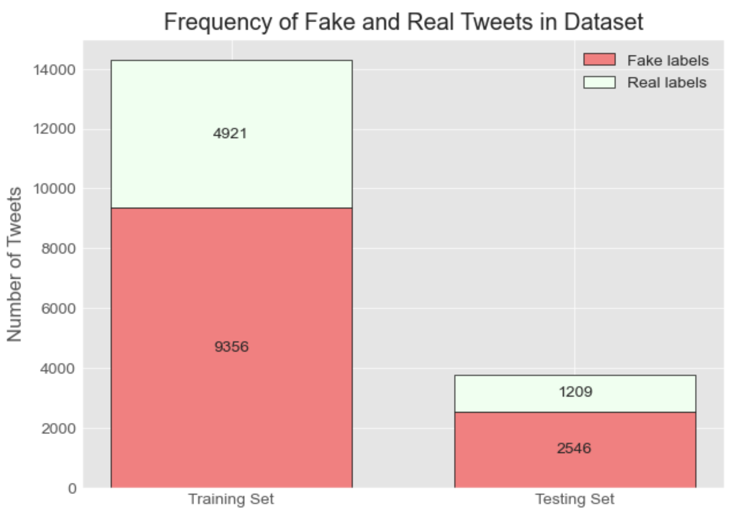
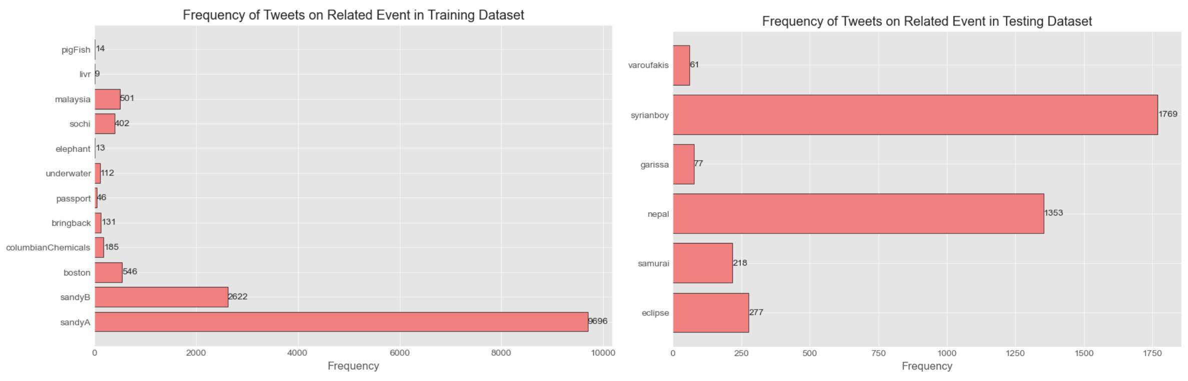


Figure 2: Proportion of Fake and Real Tweets within Dataset

and ”Nepal” earthquake events. It’s also clear that none of the events are shared between the datasets. The repetition of words (e.g. in hurricane related tweets) in some feature processing methods (bag-of-words) can reduce the generalisation ability of the model; consequently a detriment to an algorithm applied on unseen words (e.g. SYRIAN HERO BOY, in syrianboy related tweets).



((a)) ((b))

Figure 3: Frequency of Tweets on Related Events in MediaEval 2015 Dataset

Twitter gives the users the ability to repost (and ”retweet”) original tweets. Defining a tweet as fake partly depends on the time context at which the event occurs. If a hurricane tweet is reposted months later as if it occurred at the current time, it is fake. Therefore it is useful to also consider the number of retweets within both datasets. This is shown in figure 4. Furthermore, an equal split of real and fake retweets can be observed. Reposts interestingly show a bias towards real labels. This may be due to the ease of creating credibility through artificial follower counts, leading to users retweeting popular accounts [2]. Additionally, the split of retweets of sandyA is shown. 84% of fake retweets relate to sandyA, but comparatively, only 22% are real. This indicates another bias in the training dataset which must be addressed.

Conclusively, the dataset includes a good amount of data to work with, though not enough to create computation speed concerns. There are certain aspects that must be addressed

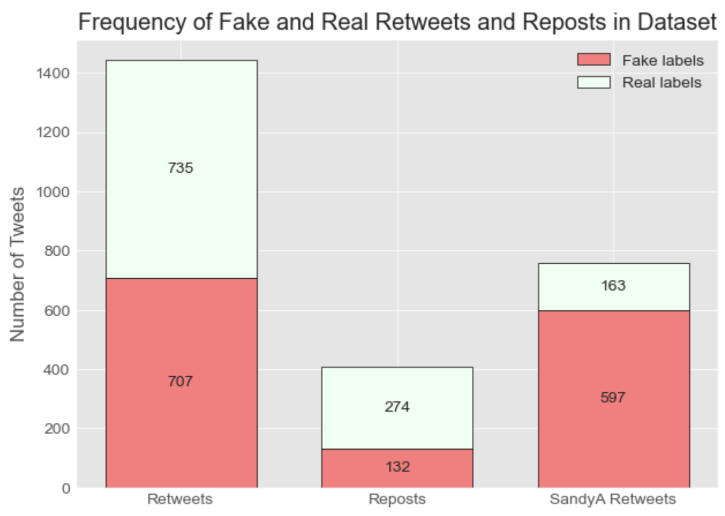


Figure 4: Proportion of Fake and Real Retweets, Reposts and SandyA Retweets. Retweets, usually denoted as ”RT” in the tweetText, usually include a username whereas reposts typically come without.

in the preprocessing stage to improve the quality of the data: dealing with bias and noise in the tweets (newlines, ampersands, emojis, etc). The model will be then trained on the ”tweetText” and the ground-truth ”labels” columns. No preprocessing will be done on the languages, due to the large bias of english tweets (AppendixA).

3. Pipeline Design

3.1. Preprocessing

The first steps were to rename all ”humor” labels to ”fake” and address the bias within the training data. A variety of methods were applicable, such as random sampling a smaller number of sandyA/sandyB related tweets, however, this would affect the amount of training data available and could introduce new biases. Instead, retweets and reposts were completely removed to remove the fake bias within sandyA retweets to keep things simple and replicable. This reduced the dimensions of the dataframe from 14277 to 12397; the data split became 77/23.

Secondly, the noise of the data was addressed in several steps. The hyperlinks within the tweetText were erased as they contain multimedia that would not be of use with later feature extraction methods. Then, due to the way tweets are formatted, many kept their newlines and ampersands symbols within the text file; these signs were removed. Emojis appear in both fake and real tweets, and thus were categorized as noise and removed.

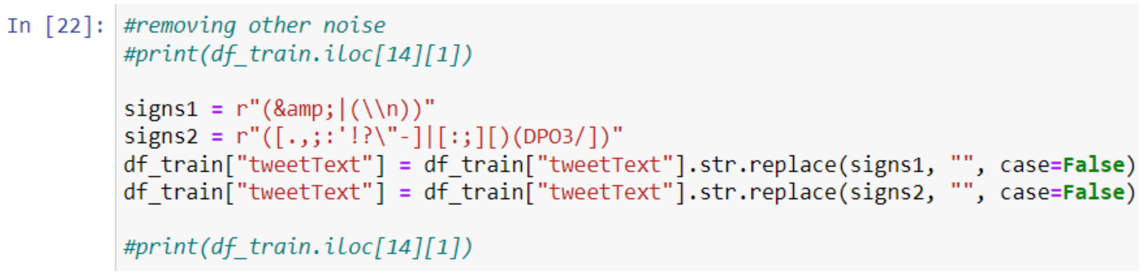
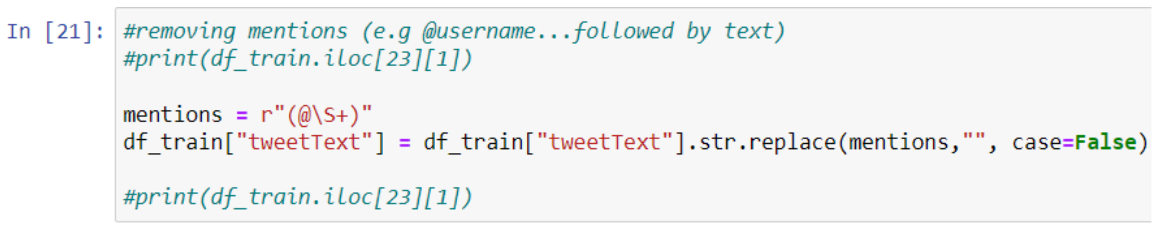

```
In [22]: #removing other noise
         #print(df_train.iloc[14][1])

         signs1 = r"(&amp;|(\\n))"
         signs2 = r"([.,;:'!?\"-]|[:;][)(DPO3/])"
         df_train["tweetText"] = df_train["tweetText"].str.replace(signs1, "", case=False)
         df_train["tweetText"] = df_train["tweetText"].str.replace(signs2, "", case=False)

         #print(df_train.iloc[14][1])
```

Figure 5: Code snippet of Preprocessing; removing punctuation and text emojis

Certain punctuation was originally untouched. Arguably, it is possible that certain excessive use of punctuation could point towards fake tweets. However, after evaluation, removing punctuation showed an improvement in the performance of the model. Similarly, lowercasing on the tweets was attempted but appeared detrimental to the performance; keeping the original casing of the tweets seemed to work better with the feature processing methods. Hashtag symbols were kept, as removing those showed a slight decrease in performance.

In the final steps, stopwords were removed using the nltk library, whitespace (from the erasure in previous steps) was removed and tweetText was also lemmatized.



```
In [21]: #removing mentions (e.g @username...followed by text)
         #print(df_train.iloc[23][1])

         mentions = r"(@\S+)"
         df_train["tweetText"] = df_train["tweetText"].str.replace(mentions,"", case=False)

         #print(df_train.iloc[23][1])
```

Figure 6: Code snippet of Preprocessing; removing hyperlinks

At all steps, the initial input and output of the preprocessing was printed for easy debugging as shown in Figure 6. The lemmatized text was put into a new column of the dataframe (”lemmatizedText”) to assess the effectiveness of this step.

The languages within the dataset were left untouched. Ideally, all posts being translated into english would provide a larger vocabulary for the model to learn from and therefore improve performance. However, to keep things simple, an assumption was made that due to the high bias towards english tweets in both training and test data, the severity of skipping this step was less than the time cost to implement such a method.

3.2. Feature Selection

To extract features, two different methods of vectorisation were attempted. The Bag-of-words (BOW) method counts the frequency of words in the tweetText column of the dataframe and applies a weight to its importance. This method is often used for text classification due to its simplicity and reliable performance [3]. However, this method vectorises by single tokens (unigrams) and thus disregards the importance of the order of words. BOW was selected due to its high performance in relevant literature, showing accuracy of up to 88% on similar text classification tasks [4].

Another state-of-the-art feature extraction method used in text classification is Term Frequency - Inverse Document Frequency (TF-IDF). It measures the importance of a words based on their frequency, where a frequent word gets assigned a lower weight. Like BOW, TF-IDF is used due its reliability and high performance. This is documented in literature with an accuracy up to 91% on text classification tasks [5]. This method was thought to be particularly useful due to the existing bias of hurricane-related data.

3.3. Algorithm Training

Two learners were selected in a similar way. The Naive Bayes (NB) model is frequently used for text classification tasks due to its reliable performance and low computational cost. Research on text classification shows F1 scores of 76% [6]. Both the Multinomial

NB and Bernoulli NB models were trained and evaluated; Multinomial NB model provided the highest score. In such algorithm evaluation studies, Support Vector Machines (SVM) models are equally popular and often scoring higher F1 scores of 89%. Multiple SVM models were tested; linearSVC provided the highest score.

The benefit of using these specific models is their thorough documentation on Scikit-learn.com and forums such as StackOverflow, which provides for easy implementation. Certain RNN models such as Long Short-Term Memory (LSTM) is often reported to have the highest performance due to their ability to store data, which would be useful if using the timestamps of the dataframe in the model. However, the simplicity of NB and SVM models and their relative high performance was preferred.

4. Performance Evaluation

The performance of both trained models were evaluated using standard metrics such as the confusion matrix, accuracy, precision, recall and F1 scores.

The confusion matrix is a table that reports the number of correctly predicted labels and incorrectly predicted labels when the supervised model is applied on the testing dataset. The model's predictions are compared to the ground-truth labels of the testing dataframe. The first row of the matrix shows true positive (TP) and false positive (FP) values. The second row similarly shows false negative (FN) and true negative (TN) values.

In the context of tweet data: TP is a correct ”fake” classification; FP is an incorrect fake classification (ground-truth label is ”real”); FN is an incorrect real classification (ground-truth label is ”fake”); TN is a correct ”real” classification. The larger number of TPs and TNs, the better the performance of the model.

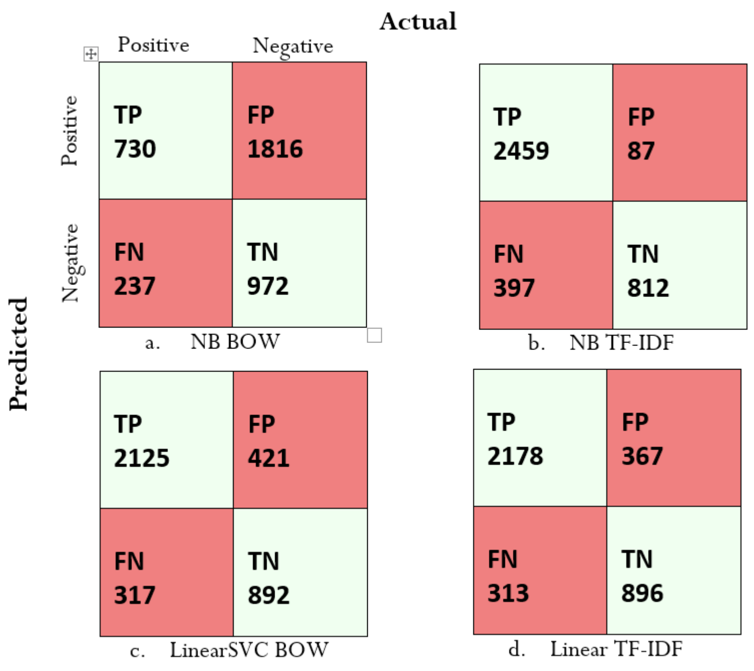


Figure 7: Confusion Matrix Report of Final Models

A confusion matrix report can be followed by simple computations to determine the accuracy, precision, recall and the F1 score. Accuracy is the proportion of correctly predicted ”fake” and ”real” labels and was calculated using:

$$\frac{\sum TP + TN}{\sum TP + TN + FP + FN} \tag{1}$$

Precision is the proportion of correct positive (”fake”) classifications and was calculated using:

$$\frac{\sum TP}{\sum TP + FP} \tag{2}$$

Recall is the proportion of actual positive (”fake”) labels identified correctly and was calculated using:

$$\frac{\sum TN}{\sum TN + FN} \tag{3}$$

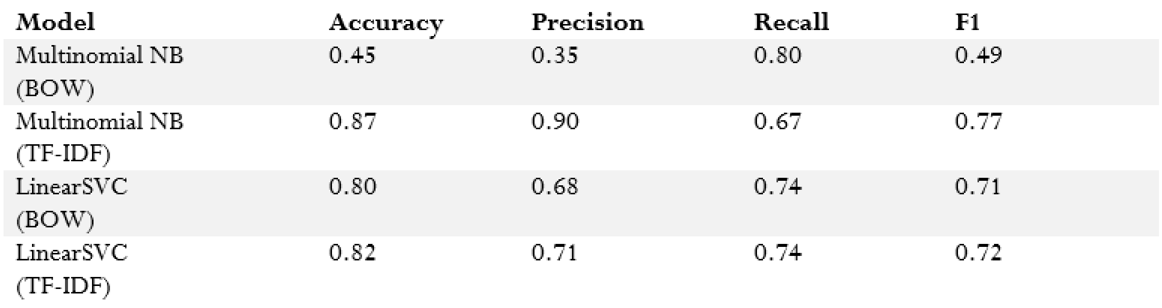
The F1 score is then the harmonic mean of the precision and recall scores and was calculated using:

$$\frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

For all these metrics, the higher the result, the better the performance of the model. In modern ML papers, the F1 score is reported to have a greater importance than accuracy. The F1 score combines the recall and precision through computing the harmonic mean, which puts a great emphasis on having variables that do not deviate far from one another. This means, both the recall and the precision have to be high for a good overall F1 score. However, achieving both high recall and precision is a challenging task discussed in literature [7].

For initial evaluation, the scores for accuracy and F1 were all low, ranging in 0.42-0.49, for both Multinomial NB and the original standard SVC models used. To improve these scores, a lot of tweaking and experimentation was done in the preprocessing stage.

First, each step in the preprocessing stage was isolated and removed to check for improvements. Consequently, it was clear that lowercasing did not work well for the feature extraction methods used; keeping original casing increased the accuracy to an average 0.66 for all iterations of the models except BOW NB. Furthermore, it appeared that removing all sorts of punctuation further improved the score. The accuracy in NB models now became 0.45 (BOW) and 0.87 (TF-IDF) respectively. However, the SVC models remained at 0.66. The standard SVC model was replaced by LinearSVC which gave an improvement in their respective scores of 0.80 (BOW) and 0.82 (TF-IDF). The final classification report for all models is shown below.



Model	Accuracy	Precision	Recall	F1
Multinomial NB (BOW)	0.45	0.35	0.80	0.49
Multinomial NB (TF-IDF)	0.87	0.90	0.67	0.77
LinearSVC (BOW)	0.80	0.68	0.74	0.71
LinearSVC (TF-IDF)	0.82	0.71	0.74	0.72

Figure 8: Classification Report of Final Models

It is clear from Figure 8 that the most stable model is particularly the LinearSVC with TF-IDF feature processing, providing the smallest deviating precision and recall rates. Nevertheless, Multinomial NB with TF-IDF provides the largest F1 score of 0.77. It is important to point out the large deviation between precision and recall, 0.90 and 0.67 respectively. It is clear that the model is incredibly good at accurately predicting the correct labels on this particular training dataset, as shown by the 0.87 accuracy; however the F1 score is limited by the subpar recall.

A higher relative accuracy compared to the F1 score implies that the classes of the dataset are unbalanced. Indeed, looking back at the label split shown in Figure 2, it is clear that the dataset contains more ”fake” labels. Consequently, as shown in Figure 7, the model correctly predicts 2459 labels to be ”fake”, compared to 812 correct ”real” classifications. The model is trained better to identify a fake post. Hypothetically, if the testing dataset contained more real labels than fake labels, the accuracy of the model would most likely suffer.

For typical Multinomial NB models in literature, it is common to see F1 scores of 0.76 [6]. In the context of this research, an accuracy of 0.87 and F1 score of 0.78 is certainly a success.

5. Conclusion

In this report, a text classification machine learning model was implemented and trained on the MediaEval 2015 Dataset. The project aimed to accurately distinguish fake tweets from real tweets through a supervised method. The dataset was initially analysed and visualized to gather an understanding of tweetText, the percentage split of the ground-truth labels and the occurrences of various covered events within the data. It was useful to identify the biases within the data before beginning the preprocessing stage of the pipeline.

In the preprocessing stage, the retweets/reposts in the training dataset were removed, and the data was cleaned for emojis, punctuation and other noise. Features were then extracted using Bag-of-words and Term Frequency - Inverse Document Frequency methods.

Finally, the Multinomial NB and LinearSVC models were then trained on tweetText and the ground-truth labels. The performance of the models was evaluated and certain stages of preprocessing were tweaked to improve accuracy and F1 scores. Multinomial NB performed the best with an accuracy of 0.82 and F1 score of 0.77.

It was insightful to learn how the performance of the model is affected by the codependency of the preprocessing stage and the feature extraction methods. Additionally, it was useful to see how the quality of the data, particularly unbalanced classes, can drastically affect the recall and precision of the model. In future research, it would be useful to implement a language translation feature within the prepossessing stage to increase the number of useful features for learning. Focus should be maintained on improving the performance of the Multinomial NB (TF-IDF) and LinearSVC (BOW) models, or exploring neural network algorithms such as LSTM.

References

[1] A. Gholamy, V. Kreinovich, and O. Kosheleva, “Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation,” 2018.

[2] Z. Gilani, R. Farahbakhsh, G. Tyson, L. Wang, and J. Crowcroft, “Of bots and humans (on twitter),” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ser. ASONAM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 349–354. [Online]. Available: https://doi.org/10.1145/3110025.3110090

[3] W. Qader, M. M. Ameen, and B. Ahmed, “An overview of bag of words;importance, implementation, applications, and challenges,” 06 2019, pp. 200–204.

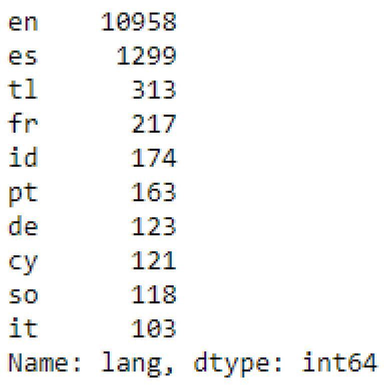
[4] Y. AYDIN, “Sentiment analyzing from tweet data’s using bag of words and word2vec,” *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 12, no. 2, p. 412–417, 2023.

[5] R. Dzisevič and D. Šešok, “Text classification using different feature extraction approaches,” in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2019, pp. 1–4.

[6] P. Jain, S. Sharma, Monica, and P. K. Aggarwal, “Classifying fake news detection using svm, naive bayes and lstm,” in *2022 12th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2022, pp. 460–464.

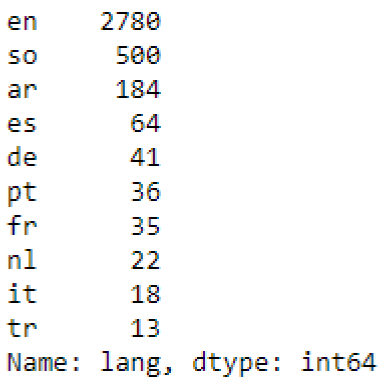
[7] M. Buckland and F. Gey, “The relationship between recall and precision,” *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12–19, 1994. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4571%28199401%2945%3A1%3C12%3A%3AAID-ASI2%3E3.0.CO%3B2-L

A. Languages in MediaEval 2015 Dataset



```
en    10958
es     1299
tl      313
fr      217
id      174
pt      163
de      123
cy      121
so      118
it      103
Name: lang, dtype: int64
```

Figure 9: Top 10 Languages in Training Data



```
en    2780
so     500
ar     184
es      64
de      41
pt      36
fr      35
nl      22
it      18
tr      13
Name: lang, dtype: int64
```

Figure 10: Top 10 Languages in Testing Data