Identifying Relevant Claims in Tweets

A DAT(550) project

Marius Gade[*]
m.gade@stud.uis.no
University of Stavanger
Stavanger, Rogaland, Norway

Maroof Mushtaq[*]
m.mushtaq@stud.uis.no
University of Stavanger
Stavanger, Rogaland, Norway

Bhakti Ravishankar Prabhu[*]
br.prabhu@stud.uis.no
University of Stavanger
Stavanger, Rogaland, Norway

Matthias Anton Schedel[*]
matthias.a.schedel@stud.uis.no
University of Stavanger
Stavanger, Rogaland, Norway

ABSTRACT

Twitter is a popular platform for public discourse, but it is also prone to misinformation and harmful content. To address this problem, we developed models using natural language processing techniques to automatically detect tweets that require fact-checking, contain verifiable factual claims, or are harmful to society.

We trained our models on a dataset focused on "Fighting the COVID-19 Infodemic and Fake News Detection", and used upsampling and balanced class weights to address class imbalance. We compared several models, including Linear SVC, Bernoulli, and Logistic Regression, using both CountVectorization and TF-IDF. Our best-performing model achieved an accuracy of 76 indicating that our approach has potential for improving the detection of misinformation on social media.

KEYWORDS

Machine Learning, Data Mining, Artificial Intelligence, Twitter Fact Checking

1 INTRODUCTION

Context and Motivation. The spread of misinformation about COVID-19 and vaccines on social media, including Twitter, has posed a significant challenge to public health and safety. While Twitter and other big platforms have introduced fact-checks and moderation to address harmful content, the sheer volume of tweets posted every second makes it impossible to review each one by hand [5].

Research Problem. In this study, we address the problem of detecting tweets that require fact-checking (subtask 1A), contain verifiable factual claims (subtask 1B), or are harmful to society (subtask 1C). To achieve this, we employ Natural Language Processing (NLP) algorithms to flag tweets that meet these criteria, which can then be reviewed by an employee. This approach has the potential to improve the efficiency and effectiveness of fact-checking and moderation on social media platforms, and to mitigate the spread of harmful misinformation.

To develop and evaluate our approach, we trained and evaluated our NLP algorithms on three English language datasets provided by the CheckThat! lab 2022 [3]. These datasets consist of tweets scraped from Twitter using various COVID-19 related keywords. They were annotated for subtasks 1A (detecting tweets that require fact-checking), 1B (detecting tweets that contain verifiable factual claims), and 1C (detecting tweets that are harmful to society). The datasets each contain a different number of tweets: 3040 for subtask 1A, 4023 for subtask 1B, and 4791 for subtask 1C [4]. To address class imbalance, we used upsampling and balanced class weights to train our models. Our models include Linear SVC, Bernoulli, and Logistic Regression, and we used both CountVectorization and TF-IDF to encode the tweets. We evaluated subtask 1A and 1C using the F1-measure with respect to the positive class, while subtask 1B was evaluated by accuracy. Our best-performing model achieved an F1-measure of 0.60 for subtask 1A, an accuracy of 67% for subtask 1B, and an F1-measure of 0.37 for subtask 1C, indicating that further improvements are needed in this area. While our results for detecting harmful tweets were not as strong as for the other subtasks, our approach shows potential for detecting tweets that require fact-checking or contain verifiable factual claims, and could be applied to other domains beyond COVID.

2 BACKGROUND

"The mission of the CheckThat! lab is to foster the development of technology to assist in the process of fact-checking news articles, political debates, and social media posts" [2, Page 417]. They organize annual labs where different groups compete in several NLP and information retrieval tasks that are related to factuality [2].

3 OUR METHOD

We approached the binary classification problems of subtasks 1A, 1B, and 1C by implementing a machine learning-based method that utilizes NLP techniques. Our method builds upon the principles of sentiment analysis, which involves analyzing text to determine the sentiment or emotion conveyed. We chose sentiment analysis as the basis for our approach because it shares many similarities with the problem of detecting tweets that require fact-checking,

[*]All authors contributed equally to this research.

Supervised by Mina Farmanbar and Vinay Jayarama Setty.

Data Mining and Deep Learning (DAT550), IDE, UiS
2023.

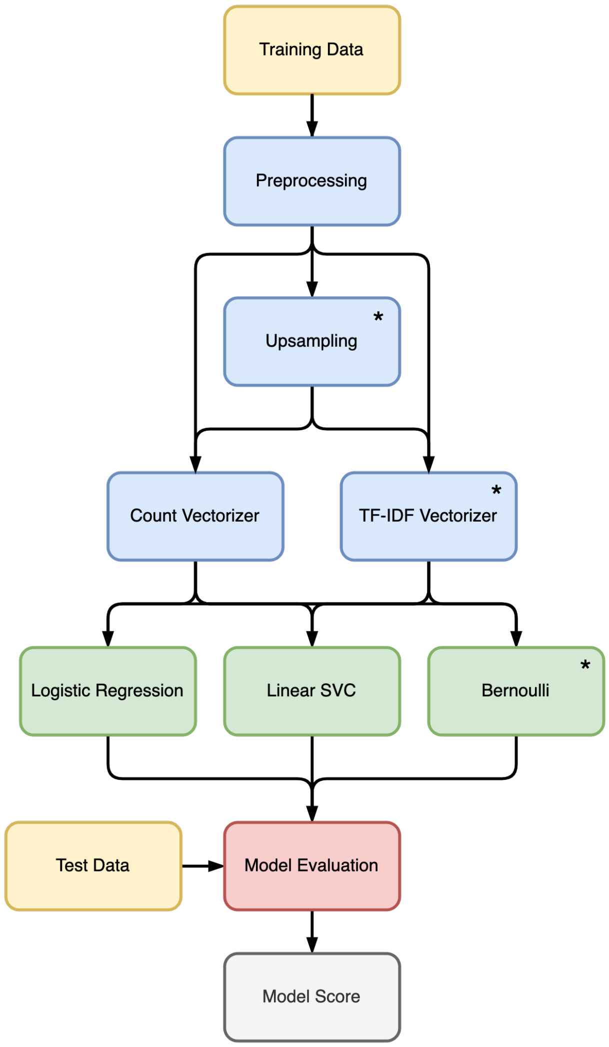


Figure 1: Visual representation of our method to evaluate different models on each dataset (* Upsampling was only done for subtask 1A and 1C while TF-IDF vectorizer and Bernoulli was only used for subtask 1B and 1C).

contain verifiable factual claims, or are harmful to society. In the following sections, we will introduce the general method used for all three subtasks and show the necessary alterations to fit the specific requirements of each subtask."

The specific code and parameters used for preprocessing, vectorization and classification are detailed in the code repository for this project.

3.1 Design

3.1.1 Preprocessing. As shown in figure (1), we first preprocessed each tweet to remove emoticons, URLs, user mentions, and sequence patterns. This was done to ensure that only the text content of the tweet was used in following analysis. Emoticons, for example, do not carry any semantic meaning and can be safely removed. URLs and user mentions were removed as they were not relevant to the classification task, and could have introduced noise to the data. Sequence patterns were also removed to avoid bias towards any specific word orderings in the tweets. An example of the before and after preprocessing of a tweet is shown in figure (2).

3.1.2 Vectorization. After preprocessing, the tweets were vectorized to represent them in a numerical format that could be used for machine learning algorithms. We used two different methods for vectorization - Count Vectorizer and TF-IDF Vectorizer. Count Vectorizer counts the frequency of each word in the tweet and represents the tweet as a sparse matrix. TF-IDF Vectorizer takes into account not only the frequency of each word, but also the importance of each word in the context of the entire corpus. The choice of these two methods was based on their commonly used success in NLP tasks, and their relative simplicity to implement.

3.1.3 Classification. For classifying we used Logistic Regression and Linear SVC models. For subtask 1B and 1C we also used a Bernoulli model. Each of the models was trained on both the Count-vectorized as well as the TF-IDF-vectorized tweets and fine-tuned to give the best results.

3.2 Optimization

3.2.1 Dealing with class imbalance. The datasets for subtask 1A and subtask 1C are highly imbalanced. To adress this we used three different approaches:

- Using the "balanced" keyword for class weights for Logistic Regression and Linear SVC.
- Finding class weights using grid search.
- Upsampling the dataset to include a similar amount of positive samples compared to the negative ones.

When calling the scikit learn implementation of a Logistic Regression or Linear SVC algorithm, using the parameter "balanced", class weights will be given by [6]:

```
n\_samples / (n\_classes * np.bincount(y))
```

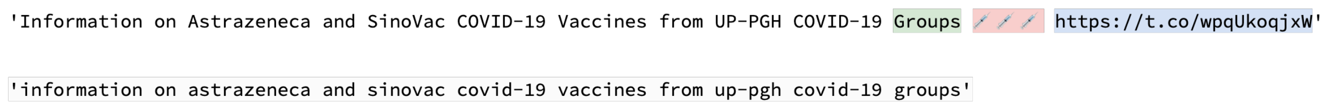
For subtask 1C where we have 3031 tweets for class 0 (not harmful) and 292 tweets for class 1 (harmful to society) this results in the class weights of `[0.54816892, 5.69006849]` for class 0 and class 1 respectively.

Gridsearch is an exhaustive method to run a model with all the hypermeter combinations and find optimal solution to a specified parameter (in our case f1-score).

Upsampling was done in two ways. For subtask 1C the entries for class 1 were simply repeated 10 times to approximately match the number of entries for class 0. In subtask 1A upsampling was implemented using SMOTE. "SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b." [1]

3.3 Implementation

Each subtask was implemented in its own jupyter notebook with Maroof Mushtaq implementing subtask 1A. Based on this notebook



```
'Information on Astrazeneca and SinoVac COVID-19 Vaccines from UP-PGH COVID-19 Groups 💉💉💉 https://t.co/wpqUkoqjxW'

'information on astrazeneca and sinovac covid-19 vaccines from up-pgh covid-19 groups'
```

Figure 2: Example of a tweet before and after preprocessing. Pink highlights emoticons, blue higlights links and green highlights a word that will be turned into its basic form.

```python
def model_Evaluate(model):

    # Predict values for Test dataset
    y_pred = model.predict(x_test_vector)

    # Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))

    # Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)

    categories  = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / np.sum(cf_matrix)]

    labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)

    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
                xticklabels = categories, yticklabels = categories)

    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
    plt.ylabel("Actual values"   , fontdict = {'size':14}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Listing 1: Custom model evaluation function

Bhakti Ravishankar Prabhu implemented subtask 1B. Finally subtask 1C was implemented by Marius Gade and Matthias Anton Schedel.

A custom model evaluation function (listing 1) was written that takes in a model and automatically predicts the values for the testing dataset, prints the evaluation metrics and plots a confusion matrix. This helped to speed up the implementation of all models and made it easy for us to compare our results to one another.

4 EXPERIMENTAL EVALUATION

For the evaluation it makes sense to look at each subtask separately. We will compare the different approaches taken and how they affected the model quality.

4.1 Subtask 1A – Check-worthiness of tweets

4.1.1 Experimental setup and results. We used 10 different configurations to classify the check-worthiness of tweets as shown in table (1). Comparing the results with regards to the F1-Score for the positive class shows that logistic regression benefits the most from implementing a weight handling method. We could get an increase of 0.13 point compared to no weight handling at all.

The best performing configuration using no up-sampling and grid-search to determine the weights. The classifier was called with these parameters:

```
LogisticRegression(C=10, class_weight=class_weight,
↪  max_iter=1000)
```

The parameter "C" is used to control overfitting by adding a penalty term to the loss function. A smaller value of C specifies stronger regularization. "max_iter" specifies the maximum number of iterations for the solver to converge. If the solver fails to converge within this limit, it will terminate the algorithm and return the best solution found so far.

Looking at the confusion matrix shown in figure (3) we can see that false positives are still quite high with 11.85%. In light of the given task it would be better to wrongly classify a tweet as being fact-check-worthy than the other way around.

4.1.2 Conclusion. We can see from the comparison table (1) that both models compare to each other with minute difference.

Model	Up-sampling	Weight handling method	F1 score (0|1)	Accurcy
LogisticRegression	none	none	0.88|0.47	81 %
LinearSVC	none	none	0.87|0.52	80 %
LinearSVC	none	inbuild	0.87|0.53	80 %
LogisticRegression	**none**	**grid-search**	**0.87|0.60**	**80 %**
LinearSVC	none	grid-search	0.77|0.55	70 %
LogisticRegression	SMOTE	none	0.85|0.49	77 %
LinearSVC	SMOTE	none	0.84|0.49	76 %
LogisticRegression	SMOTE	grid-search	0.85|0.48	76 %
LinearSVC	SMOTE	grid-search	0.86|0.47	78 %

Table 1: Configuration and result for subtask 1A

ML Algorithm	Feature Extraction Method	Precision (0|1)	F1-score (0|1)	Accuracy
Linear SVC	CountVectorization	0.61|0.81	0.66|0.77	73 %
Bernoulli	**CountVectorization**	**0.68|0.80**	**0.67|0.81**	**76 %**
Logistic Regression	CountVectorization	0.62|0.84	0.68|0.78	74 %
Linear SVC	TF-IDF	0.71|0.76	0.61|0.81	75 %
Bernoulli	TF-IDF	0.67|0.79	0.64|0.80	75 %
Logistic Regression	TF-IDF	0.72|0.75	0.60|0.82	75 %

Table 2: Configuration and result for subtask 1B

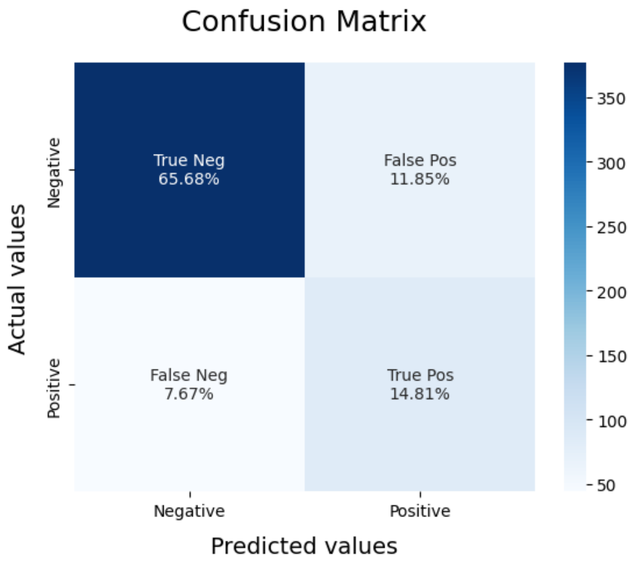


Figure 3: Confusion matrix of the best performing model configuration for subtask 1A

However Logistic regression does well when class balancing is done manually.

In conclusion, both models can be used to fact check tweets. Performance of both models improve when using class balanced training and shows little to no improvement with up-sampling method.

4.2 Subtask 1B – Verifiable factual claims detection

4.2.1 Experimental setup and results. For subtask 1B we compared three different models with two different vectorisation methods. The different configurations and their respective results are shown in table (2). For subtask 1B we had to optimize the accuracy of the model. The best performing configuration is using no up-sampling and Bernoulli as a classifier. The classifier was called with these parameters:

```
classifier = BernoulliNB(alpha = 0.5)
```

The "alpha" parameter specifies the Laplace smoothing parameter, which is used to avoid zero probabilities in the probability estimates. A smaller value of alpha results in less smoothing and more reliance on the data, while a larger value of alpha results in more smoothing and less reliance on the data.

Looking at the confusion matrix of the best performing model for subtaskt 1B shown in figure (4) we can see, that both the false negative and the false positives have similar probalilities around 12%. This shows that the model does not tend to vote for one of the classes but is balanced in its classifications.

4.2.2 Conclusion. Table (2) shows us, that all the models have nearly same Accuracy, Precision and F1-Score, with just minor differences. Based on overall score, we could consider Bernoulli with CountVectorization as the best model due to good precision value, better F1-score and the best accuracy. We conclude that we can use any of these models to predict whether a tweet contains verifiable factual claims.

ML Algorithm	Up-sampling	Feature Extraction Method	Precision (0|1)	F1-score (0|1)	Accuracy
Linear SVC (no weight)	none	CountVectorization	0.92|0.60	0.95|0.24	91 %
Linear SVC (balanced)	**none**	**CountVectorization**	**0.94|0.40**	**0.94|0.37**	**89 %**
Bernoulli (0.1)	none	CountVectorization	0.93|0.45	0.95|0.37	90 %
Logistic Regression	none	CountVectorization	0.93|0.39	0.94|0.36	89 %
Linear SVC	none	TF-IDF	0.91|0.62	0.95|0.11	91 %
Bernoulli	none	TF-IDF	0.92|0.61	0.95|0.22	91 %
Logistic Regression (balanced)	none	TF-IDF	0.92|63	0.96|0.24	92 %
Logistic Regression (grid search)	none	TF-IDF	0.92|63	0.96|0.24	92 %
Linear SVC	repeating entires	CountVectorization	0.93|0.40	0.94|0.34	90 %
Bernoulli	repeating entires	CountVectorization	0.94|0.34	0.93|0.35	88 %
Logistic Regression (balanced)	repeating entires	CountVectorization	0.93|0.36	0.94|0.35	89 %
Linear SVC	repeating entires	TF-IDF	0.92|0.63	0.96|0.24	92 %
Bernoulli	repeating entires	TF-IDF	0.93|0.50	0.95|0.31	91 %
Logistic Regression (balanced)	repeating entires	TF-IDF	0.92|0.63	0.96|0.24	92 %
Logistic Regression (grid search)	repeating entires	TF-IDF	0.92|0.52	0.95|0.26	91 %

Table 3: Configuration and result for subtask 1C

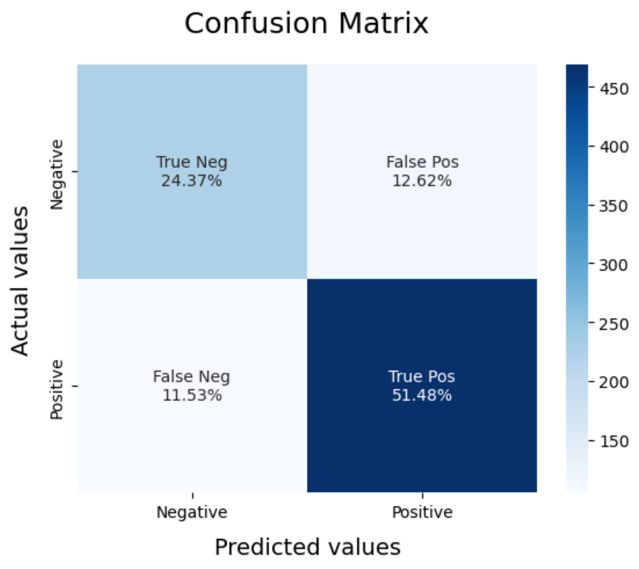


Figure 4: Confusion matrix of the best performing model configuration for subtask 1B

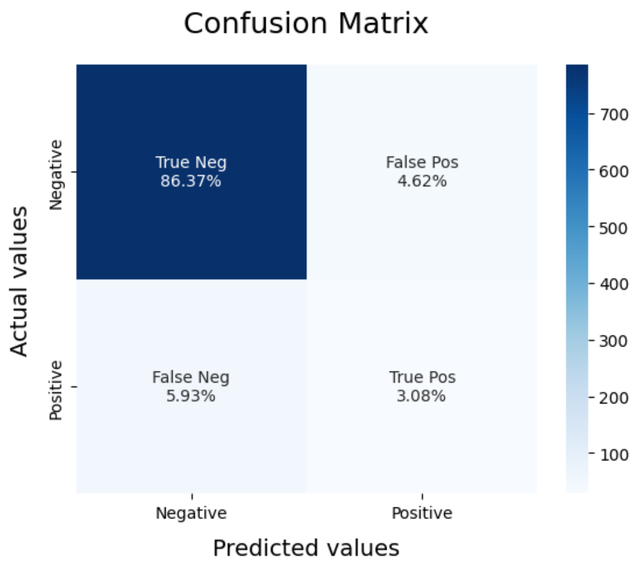


Figure 5: Confusion matrix of the best performing model configuration for subtask 1C

4.3 Subtask 1C – Harmful tweet detection

4.3.1 Experimental setup and results. For subtask 1C, 15 different configurations were tested and compared. This includes two different vectorisation methods, up-sampling and different class weights. The configurations and the achieved results are shown in table (3). Because of the large imbalance in classes we tried to optimize the models in respect to the F1-score of the positive class. The best performing configuration is the linear SVC model using count vectorization. The classifier was called with the following parameters:

```
classifier = LinearSVC(C=0.05,
↪  class_weight={0: 0.54816892, 1:5.69006849},
↪  max_iter=1000)
```

The class weight was obtained using the scikit learn utility function compute_class_weight [6]

Taking a look at the confusion matrix shown in figure (5) we can see, the severity of the class imbalance. We can also assume that for this model to be used in production, a bias to classify towards the positive class would have to be introduced.

4.3.2 Conclusion. The dataset for this subtask was highly imbalanced. We found it difficult to improve the F1-Score for class one. We believe that further research will provide better results. In future projects with similar problems we would like to test other methods like:

- Ensemble methods: Ensemble methods such as bagging and boosting can also help to balance the class distribution. For example, the Random Forest algorithm uses bagging and can handle imbalanced data well.
- Cost-sensitive learning: In cost-sensitive learning, the cost of misclassifying instances from the minority class is given higher weight during training. This approach encourages the model to focus more on correctly predicting the minority class.
- Data augmentation: Generating synthetic data like we have done in subtask 1A. Using techniques such as data augmentation can help to increase the size of the minority class and balance the class distribution.
- Anomaly detection: In some cases, the minority class represents anomalies or outliers. In these cases, anomaly detection techniques such as one-class SVM can be used to identify and classify the minority class.

5 CONCLUSION

In this project we have implemented and compared several method to identify relevant claims in Tweets. The english language dataset contained labaled tweets for the three subtasks:

- Subtask 1A – Check-worthiness of tweets
- Subtask 1B – Detection of verifiable factual claims
- Subtask 1C – Detection of harmful tweets

Our top-performing models for each subtask achieved an F1-score of 0.60 for subtask 1A, an accuracy of 67% for subtask 1B, an F1-score of 0.37 for subtask 1C. However, the results indicate that there is room for further improvement in this particular subtask. Although our approach for detecting harmful tweets did not perform as well, our methodology has the potential to identify tweets that require fact-checking or contain verifiable factual claims. Furthermore, this approach could be adapted to other domains beyond COVID.

6 ACRONYMS

NLP Natural Language Processing

7 CREDITS

This paper is the result of a group project for Data Mining and Deep Learning (DAT550). The tasks were devided as follows:

- Marius Gade: Subtask 1C and preparing the presentation
- Maroof Mushtaq: Subtask 1A
- Bhakti Ravishankar Prabhu: Subtask 1B
- Matthias Anton Schedel: Subtask 1C and writing the report

All the code can be found in our github repository (https://github.com/MattthiasSchedel/G09-DAT550-Project).

REFERENCES

[1] Haibo He and Yunqian Ma. 2013. *Imbalanced learning: Foundations, algorithms, and applications / edited by Haibo He, Yunqian Ma.* John Wiley & Sons, Chichester.

[2] Preslav Nakov, Alberto Barrón-Cedeño, and Giovanni Da San Martino. 2022. *The CLEF-2022 CheckThat! Lab on Fighting the COVID-19 Infodemic and Fake News Detection.* Springer, Switzerland.

[3] Preslav Nakov, Alberto Barrón-Cedeño, and Giovanni Da San Martino. 2022. *Overview of the CLEF–2022 CheckThat! Lab on Fighting the COVID-19 Infodemic and Fake News Detection.* Springer Science and Business Media Deutschland GmbH, Switzerland.

[4] Preslav Nakov, Alberto Barrón-Cedeño, and Giovanni Da San Martino. 2022. *Overview of the CLEF-2022 CheckThat! Lab Task 1 on Identifying Relevant Claims in Tweets.* CEUR Workshop Proceedings (CEUR-WS.org).

[5] Omnicore Agency. 2023. Twitter by the Numbers (2023): Stats, Demographics & Fun Facts. https://www.omnicoreagency.com/twitter-statistics/

[6] scikit learn. 2023. sklearn.utils.class_weight.compute_class_weight. https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html