Don’t Eat the Onion:

Detecting “The Onion” vs non-Onion Headlines

Team: Liam Bush, Jacob Feigenberg, Kevin Zhang, Eduardo Gonzalez. **TA:** Kunli Zhang

1) Introduction

Problem: “The Onion” (hereby simply, Onion) is a newspaper organization that publishes satirical news articles. As a famous satire publication, Onion headlines/articles have often mistakenly been taken seriously. One such instance was in May 2015, when former FIFA vice president Jack Warner mistakenly spread the Onion headline “FIFA Frantically Announces 2015 Summer World Cup In United States” as real news on Facebook. We also see more serious implications of this (discussed in “Motivation”). There is even a dedicated subreddit, r/AteTheOnion, with nearly 600K members focused on those who failed to see through The Onion’s satire. As avid readers of The Onion, we decided on the following task:
We aim to detect “The Onion” compared to non-Onion headlines. Our model will be able to output/classify, for a given headline, whether it is an Onion or non-Onion headline.

The inputs to the system are headlines in text form, and the outputs will be a binary classification of whether the headline is Onion or non-Onion. We are using data from a dataset of 24,000 total Onion and non-Onion article headlines, with the non-Onion articles being “Onion-like” sourced from r/NotTheOnion (subreddit dedicated to real headlines that may appear satirical). The breakdown of the dataset is 15,000 non-Onion headlines, and 9,000 Onion headlines. The dataset is linked here: https://github.com/lukefeilberg/onion. We will be splitting this into a 60/20/20 train/validation/test split, with the validation set used as required for any hyperparameter selection. Our aim is to build the best classification system possible, and foresee creating/tuning several iterations of model types/parameters to achieve this objective and contribute relative to previous work (see later sections).

To evaluate our proposed system, we will measure log loss between our predicted Onion/non-Onion label and actual labels for training, and for further evaluation/test, consider metrics such as classification accuracy, precision, recall, and/or F1 scores where appropriate.

Motivation: While the classification of articles as Onion vs non-Onion may seem frivolous at first glance, we believe the implications are actually deep. The very fact that entire subreddits have been dedicated to both the failure to identify Onion articles (r/AteTheOnion) or to real headlines that seem Onion-esque (r/NotTheOnion) highlights an important point: Information, specifically what is real/fake, exaggerated/accurate, and satire/serious, actually walks a fine line between each contrast. In an age where information spreads quicker than ever, systems that can discern between these contrasts will also become more important. As such, a high performing system in this project offers a step forward in the fight against the spread of misinformation, even if such misinformation is more light-hearted in nature. We also believe that the flip side to this is to ensure that the idea of curbing misinformation does not inadvertently become a euphemism for censorship. As students, we commit to ethical research practices, prioritizing that our system will have limited risk for any chance of user harm, of which we see limited chances of within our specific aim of a classification problem.

2) How We Have Addressed Feedback From the Proposal Evaluations

Kunli's feedback on our initial proposal was very helpful, especially his note, "I am not sure about the datasets you are using. It seems that the journal entries are already in string data. Where will you be getting your handwritten journal data from?" Having reviewed the feedback, our group realized that our initial first proposal, "Analyzing mental health through sentiment analysis on handwritten journaling entries", lacked the necessary dataset to be able to fully evaluate our pipeline performance from handwritten journal entries to sentiment analysis scoring. As such, we subsequently met Kunli during his Office Hours, where we discussed several alternatives, eventually mutually agreeing that the classification between Onion and non-Onion article headlines offers a challenging and interesting project idea which we are now implementing. The most important piece of feedback that we received is that we need to work with a proper dataset; we selected data that would be the best fit for this classification task and will introduce it in later sections.

Lastly, another piece of feedback Kunli offered which, *time permitting*, can be explored as an additional feature is to develop a user interface that allows users to input headlines and receive classification results. This interactive feature will be powered by a model trained on a dataset comprising both satirical and genuine news headlines. The model architecture will be enhanced to handle queries efficiently and provide accurate classifications. This allows us to further work towards our stated aim within "Motivation" to consider the problem of classifying Onion vs non-Onion articles as a sub-problem of the general fight against misinformation, and would permit us to better spread awareness of the problem through a more light-hearted application.

3) Prior Work We are Closely Building From

A few works inspired our contribution to the issue of headline satire detection and, more broadly, to the issue of misinformation.

Automatic Satire Detection: Are You Having a Laugh? https://aclanthology.org/P09-2041.pdf. This work by Clint Burfoot and Timothy Baldwin of the University of Melbourne used SVMs and specially constructed features to delineate satirical news articles from real news, giving us a reference point to which we can compare complicated learned features (deep learning) and a simpler bag-of-words approach. The authors also emphasized the importance of headlines as a feature in classifying satirical news, supporting our approach of focusing on Onion headlines.

Fake News Detection: A Deep Learning Approach. https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview. This work by Aswini Thota, Priyanka Tilak, Simrat Ahluwalia, and Nibrat Lohia of Southern Methodist University gave us inspiration on how to use pre-trained word embeddings (like those from Google's Word2Vec) to feed into our own neural networks.

4) What We are Contributing

Our main contribution will be the development of a machine learning model specifically tuned to detect satirical headlines from the Onion, which often resemble real headlines. That is, by tuning and using an LSTM, we will apply an existing machine learning algorithm to a new domain (application). This differs from prior work by attempting to detect satire or misinformation just from a headline; since readers often get most of their information from headlines (and skim the rest of the article), it is critical that, with the help of our model(s), they can discern between fake and real headlines. Additionally, we hope to expose flaws in using other models, like CNNs (designed for images), in detecting satirical headlines (analysis).

5) Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

5.1 Methods

Our approach to investigating how to classify satirical headlines will begin with exploratory data analysis and work from simpler to more complex, higher capacity models. Much of this is reiterated below, but we've looked at word distributions and produced word clouds and confusion matrices to understand the properties of this headline dataset. The next step was preprocessing: creating our own bag of words as well as tokenizing with existing techniques like spaCy and Byte Pair Encoding. These handcrafted features were then fed into a logistic regression classifier to get our first results (discussed below). Next, we plan to use these features to train decision tree-based ensembles (random forest) and boosted variations of these models (Adaboost, XGBoost). Similarly to Thota et al. (Section 3), we plan on using pre-trained word embeddings to feed into neural networks; for this deep learning stage, we also plan on tweaking existing CNN (1D architecture designed for text classification) and LSTM architectures to provide a higher capacity model (in hope of much higher predictive power). This will get us to our main contributions, namely, producing a LSTM model that can predict satirical headlines and demonstrating the LSTMs over CNNs in text-related classification problems. The latter contribution is supported by plenty of research [Kalaivani, 2021].

5.2 Experiments and Results

The key question our experiments will try to answer is whether it is possible to build a high performing classifier for Onion or non-Onion headlines. Our alternate hypothesis for this research question is that it is possible, and that an accuracy of at least 90% is achievable on the dataset. In addition to considering accuracy scores (which will define our alternate/null hypotheses), we will also consider other possible performance metrics such as, but not necessarily limited to, precision, recall, and/or F1 scores. These will also be evaluated in the context of certain task-specific objectives, such as what the weighting of precision vs recall in the context of a misinformation classifier more generally should be.

Our results thus far are as follows:

1. Firstly, our experiment begins with better understanding our dataset. To do this, we utilized various techniques such as plotting word distributions, word clouds, and

confusion matrices to better ascertain the dataset's distributions. *Please see Appendix 1 for some results.*

2. We also worked on preprocessing our data. Some examples of this are removing stop words, and also exploring bag of words and tokenization techniques like spaCy or Byte Pair Encoding. *Please see Appendix 2 for some results.*
3. Finally, we begin to set up actual prediction models for our classification task. We decided that a simple *baseline* for our task will be to utilize logistic regression, specifically with Byte Pair Encoding (BPE) as an initial attempt before further iteration. We also varied the level of preprocessing. A summary of results so far are as follows:
    a. Baseline: Logistic regression, BPE: We used this as our baseline as BPE allowed us to quickly get a tokenization output without too much pre-planning. We didn't do any text preprocessing. This achieved a test accuracy of 0.74. *Please see Appendix 3 for some results.*
    b. Iteration 1: We then considered a rule-based tokenizer, specifically spaCy. Preprocessing here comprised of lowercasing, removing URLs/special characters, and removing stop words. This achieved a test accuracy of 0.82. *Please see Appendix 4 for some results.*
    c. Iteration 2: We then continued with spaCy, but with preprocessing identical as Iteration 1 but keeping stop words. This achieved a test accuracy of 0.84. We believe the higher accuracy may indicate going forward that stop words provide additional context to the classifier for Onion articles. *Please see Appendix 5 for some results.*
    d. Iteration 3: We then experimented with bag of words, with preprocessing identical as Iteration 2. This achieved a test accuracy of 0.85. *Please see Appendix 6 for some results.*
    e. Iteration 4: Finally, we tried bag of words, without text preprocessing. This achieved a test accuracy of 0.854. We believe the higher accuracy indicates that removed strings such as links may provide additional context to the classifier. *Please see Appendix 7 for some results.*

6) Risk Mitigation Plan

To build a viable project within the remaining time, we plan to iterate through various model types (e.g. LSTM, tree-based models, etc.) in order to move towards our objective of the highest performing classifier possible. At each iteration, we will consider the pros/cons of prior approaches before considering next steps to maximize efficiency. For example, if we find prior model iterations tend to overfit, we will take this learning into account before deciding on next model types or hyperparameters going forward. With our group having made considerable progress already in EDA, preprocessing, and baseline model building, we have created a "simplified setting" with early results to continue building off of. While we do not believe compute will become a bottleneck, in the scenario this occurs, we will reduce the size of our training set by subsampling to ensure on-time project completion. Finally, if any other unforeseen headwind occurs, we plan to firstly communicate with our TA mentor, before then considering approaches such as further dataset modification, model complexity downgrading, or reframing objectives.

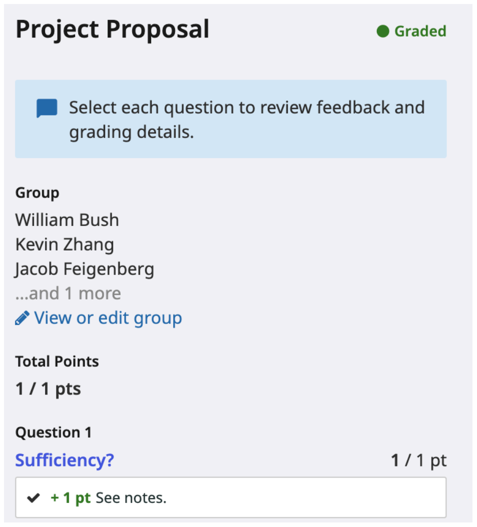
Other Prior Work / References (apart from Sec 3)

M. S. Kalaivani, S. Jayalakshmi, “Comparative Analysis of Convolutional Neural Network and LSTM in Text-Based Sentiment Classification,” IEEE 2021.

Full Work Plan (completed in green)

Person(s)	Task	Apr 9 - Apr 15	Apr 16 - Apr 22	Apr 23 - Apr 29	Apr 30 - May 6	May 7 - May 13 (2 day buffer before final deadline)
Kevin, Jacob	Review Related Works	🟩				
Kevin, Jacob	Project Check-in Writeup		🟩			
Liam	Data Preprocessing		🟩			
Liam	EDA, Constructing bag of words featurization		🟩			
Eduardo	Logistic Regression		🟩			
Eduardo	Modeling with Spacy, BytePair		🟩			
Eduardo	Tree-based Models			🟦		
Jacob, Kevin	CNN			🟦		
Jacob, Kevin	LSTM				🟦	
Liam	Adaboost/XGBoost			🟦		
Liam	Baseline test on ChatGPT API				🟦	
Everyone	Final Writeup and Presentation					🟦

Gradescope



Project Proposal ● Graded

Select each question to review feedback and grading details.

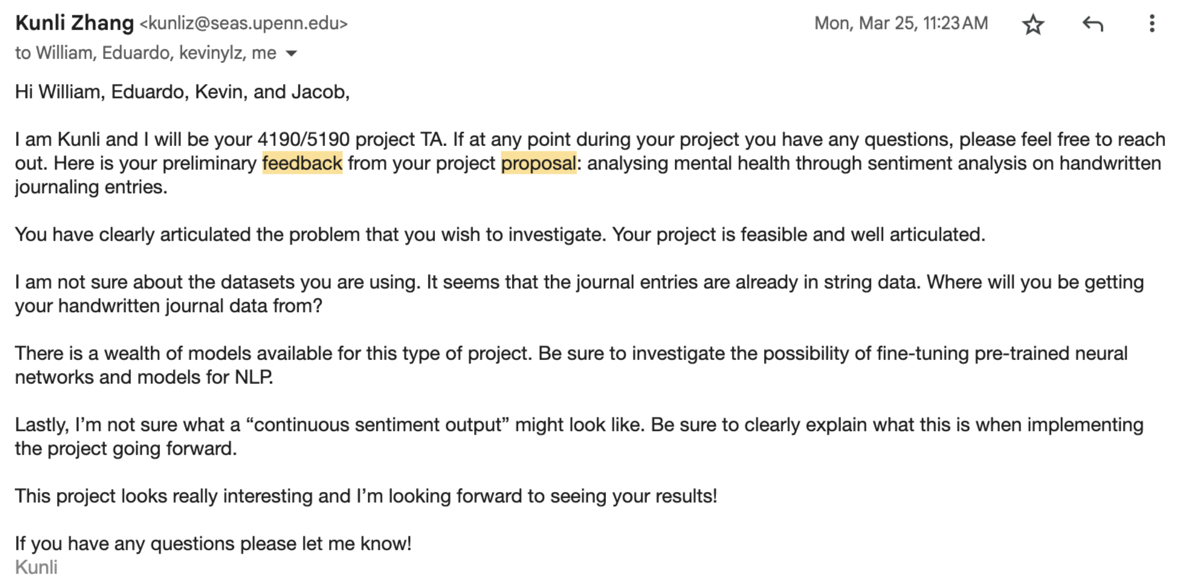
Group
William Bush
Kevin Zhang
Jacob Feigenberg
...and 1 more
View or edit group

Total Points
1 / 1 pts

Question 1
Sufficiency? **1** / 1 pt

✓ **+ 1 pt** See notes.

TA Feedback (via email)



Kunli Zhang <kunliz@seas.upenn.edu>

Mon, Mar 25, 11:23 AM

to William, Eduardo, kevinylz, me

Hi William, Eduardo, Kevin, and Jacob,

I am Kunli and I will be your 4190/5190 project TA. If at any point during your project you have any questions, please feel free to reach out. Here is your preliminary feedback from your project proposal: analysing mental health through sentiment analysis on handwritten journaling entries.

You have clearly articulated the problem that you wish to investigate. Your project is feasible and well articulated.

I am not sure about the datasets you are using. It seems that the journal entries are already in string data. Where will you be getting your handwritten journal data from?

There is a wealth of models available for this type of project. Be sure to investigate the possibility of fine-tuning pre-trained neural networks and models for NLP.

Lastly, I'm not sure what a "continuous sentiment output" might look like. Be sure to clearly explain what this is when implementing the project going forward.

This project looks really interesting and I'm looking forward to seeing your results!

If you have any questions please let me know!
Kunli

Project Proposal

Analyzing mental health through sentiment analysis on handwritten journaling entries

Team: Kevin Zhang, Eduardo Gonzalez, Jacob Feigenberg, Liam Bush

Task T: We aim to analyze mental health states based on sentiment analysis of journaling entries. Our model will be able to convert images of users' handwritten journaling entries to text, after which sentiment analysis is performed to analyze their mental health states.

Experience E: We are using data from a dataset with 1500 journal entries from 500 respondents to train a model on sentiment analysis of text (https://www.kaggle.com/datasets/madhavmalhotra/journal-entries-with-labelled-emotions/data). This will be split into a training set, validation set, and test set for the model. We will also use another dataset containing handwritten text so that we can build a full pipeline from written journal entries to estimated sentiment (https://www.kaggle.com/code/dromosys/handwriting-recognition-cnn/input). This will also undergo a train-val-test split for the other model.

Performance Metrics P: We will measure cross entropy loss between our predicted emotion labels and actual labels through a test set. Furthermore, we may also implement a form of regression to classify intensity of positive and negative emotions, in which case we would also add a mean square error performance metric to measure performance on regression.

Prior Work:

1. Kalluri Shashank, "Deep Learning Based Sentiment Analysis", Blekinge Institute of Technology 2023. (This paper studies text data coming from various social media platforms and performs sentiment analysis on the entries).

Nature of Main Proposed Contribution(s):
Our main contribution will be the development of a machine learning model specifically tuned to detect mental health labels in personal journal entries. This differs from prior work by focusing on unsolicited, naturalistic text data from individuals over time, providing insights into daily life that clinical transcripts might not capture.

Why We Care: We're diving into this project because we believe in the power of merging tech with mental health. By analyzing journal entries for mood trends, we can create something that could make a difference. It allows us to spot early signs of stress or depression and do something about it. This project hits close to home for many of us, and we're excited to see if we can turn journaling into a tool for better mental well-being.

Which Parts of the Curriculum from This Class Do You Expect to Apply?:
We plan to utilize standard NLP analysis, particularly those focused on sentiment analysis. As a baseline, we will consider using a 'bag-of-words' logistic classifier for classification (and similarly, can use a linear regression with 'bag-of-words' encoding to apply this to continuous sentiment output). We expect to employ and compare various models, including decision trees for their interpretability and CNNs for image data.

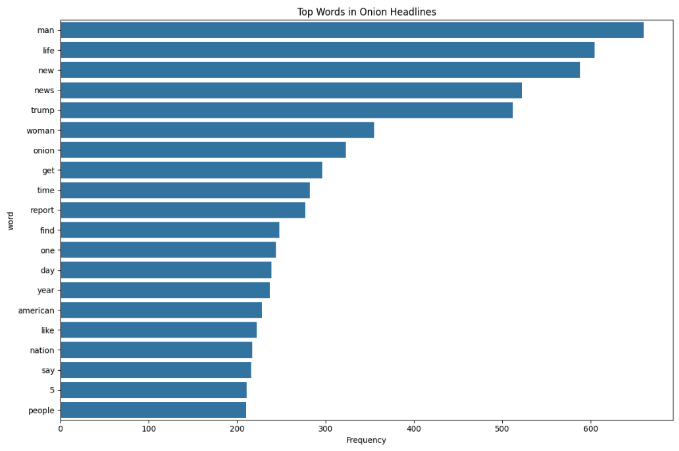
Compute Requirements:
We estimate needing moderate computational resources, including GPUs for training our models due to the extensive dataset of text entries. Cloud-based services will likely suffice, with an emphasis on memory optimization for handling large text.

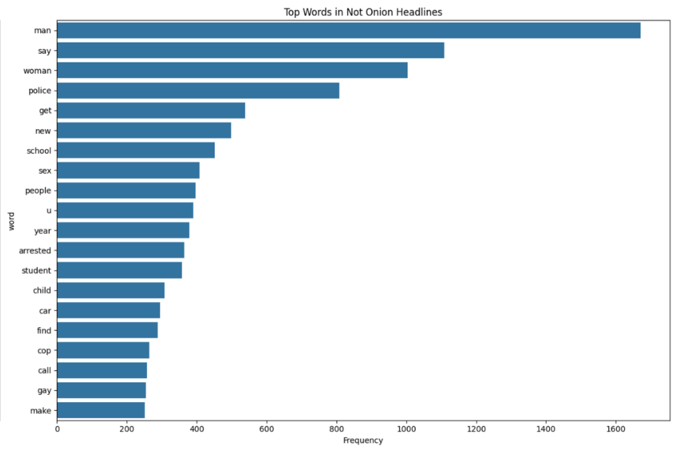
Expected Challenges and Risk Mitigation:
A significant challenge will be understanding outlier handwritten data, such as particularly unreadable handwriting. Another challenge is the subjective nature of mental health and sentiment, which we aim to address by evaluating the problem through multiple lenses (e.g. potentially both classification and regression lens).

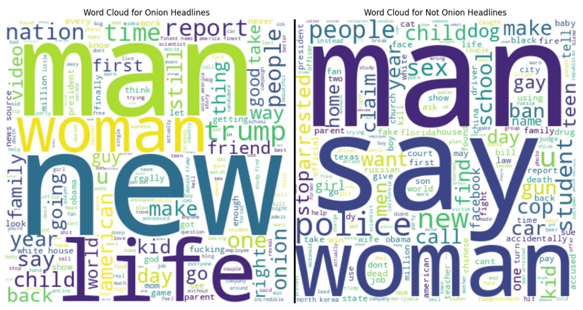
Ethical Considerations and Broader Social Impact:
The successful prediction of mental health states from journal entries could offer a novel, non-invasive tool for early detection of mental health issues. However, it also raises privacy concerns and the risk of misinterpretation or misuse of data. We commit to ethical research practices, prioritizing participant consent and data security.

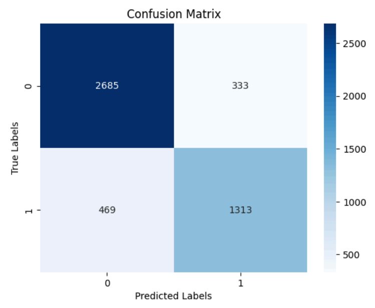
Appendix

Appendix 1

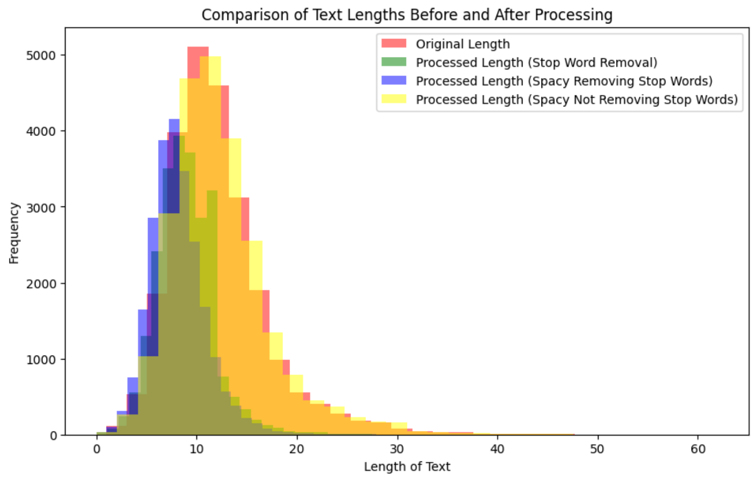




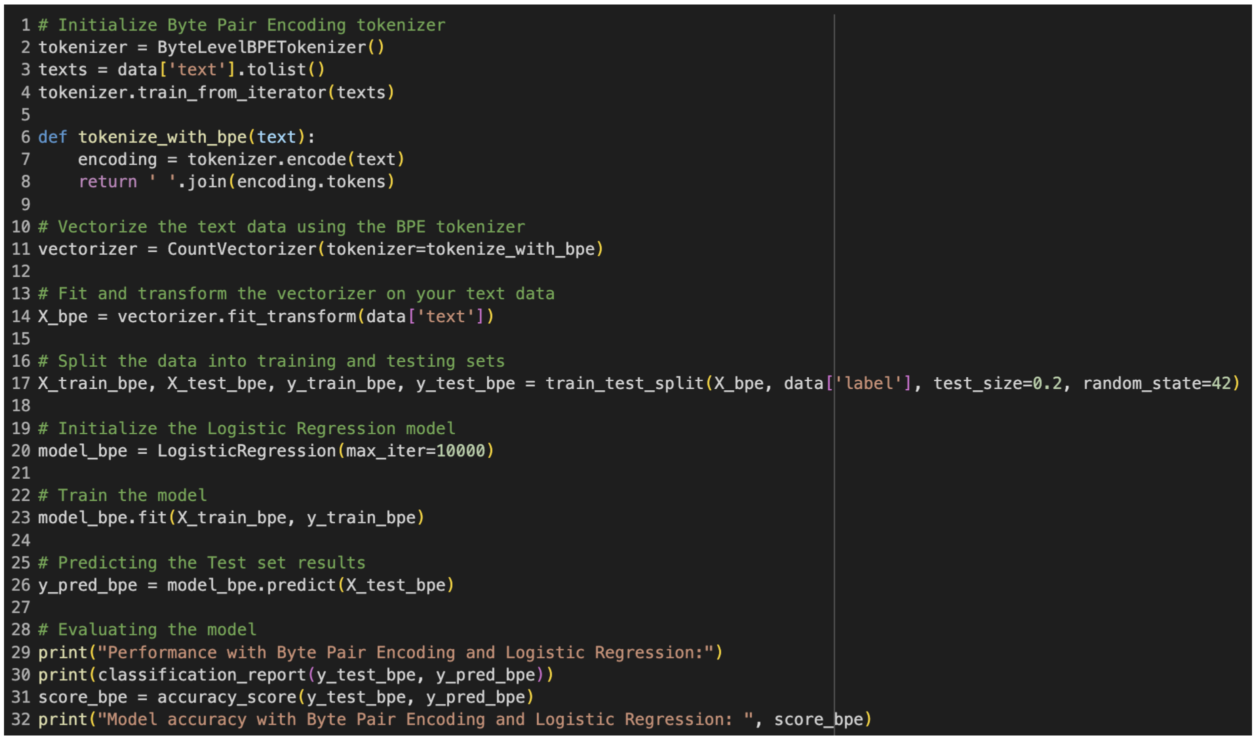




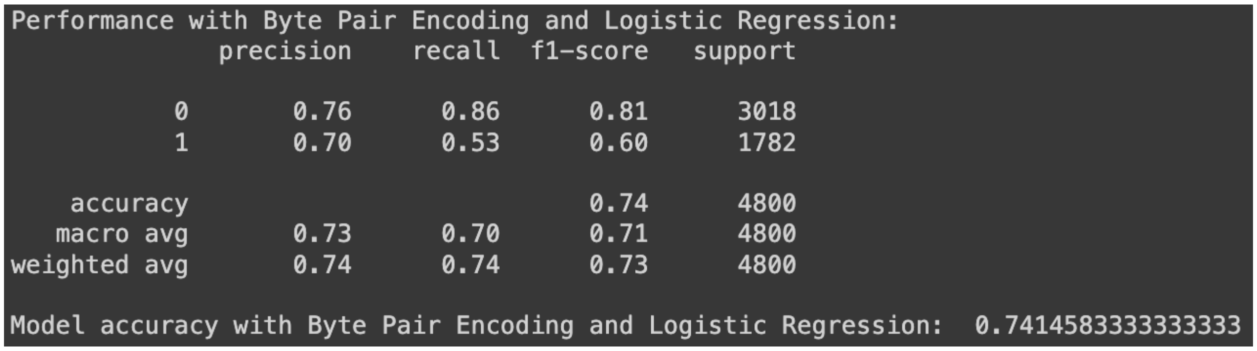
Appendix 2



Appendix 3

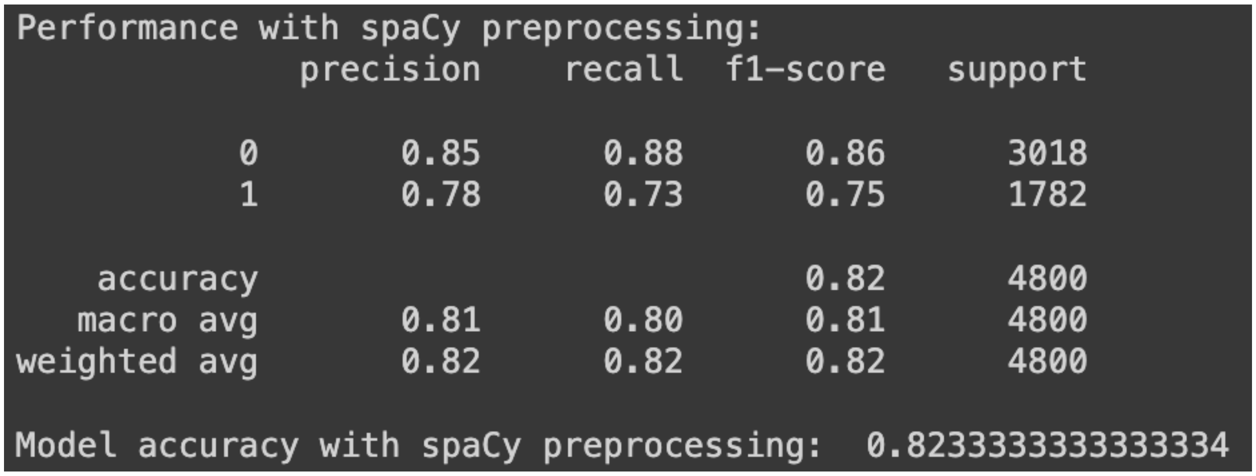


```
# Initialize Byte Pair Encoding tokenizer
tokenizer = ByteLevelBPETokenizer()
texts = data['text'].tolist()
tokenizer.train_from_iterator(texts)

def tokenize_with_bpe(text):
    encoding = tokenizer.encode(text)
    return ' '.join(encoding.tokens)

# Vectorize the text data using the BPE tokenizer
vectorizer = CountVectorizer(tokenizer=tokenize_with_bpe)

# Fit and transform the vectorizer on your text data
X_bpe = vectorizer.fit_transform(data['text'])

# Split the data into training and testing sets
X_train_bpe, X_test_bpe, y_train_bpe, y_test_bpe = train_test_split(X_bpe, data['label'], test_size=0.2, random_state=42)

# Initialize the Logistic Regression model
model_bpe = LogisticRegression(max_iter=10000)

# Train the model
model_bpe.fit(X_train_bpe, y_train_bpe)

# Predicting the Test set results
y_pred_bpe = model_bpe.predict(X_test_bpe)

# Evaluating the model
print("Performance with Byte Pair Encoding and Logistic Regression:")
print(classification_report(y_test_bpe, y_pred_bpe))
score_bpe = accuracy_score(y_test_bpe, y_pred_bpe)
print("Model accuracy with Byte Pair Encoding and Logistic Regression: ", score_bpe)
```



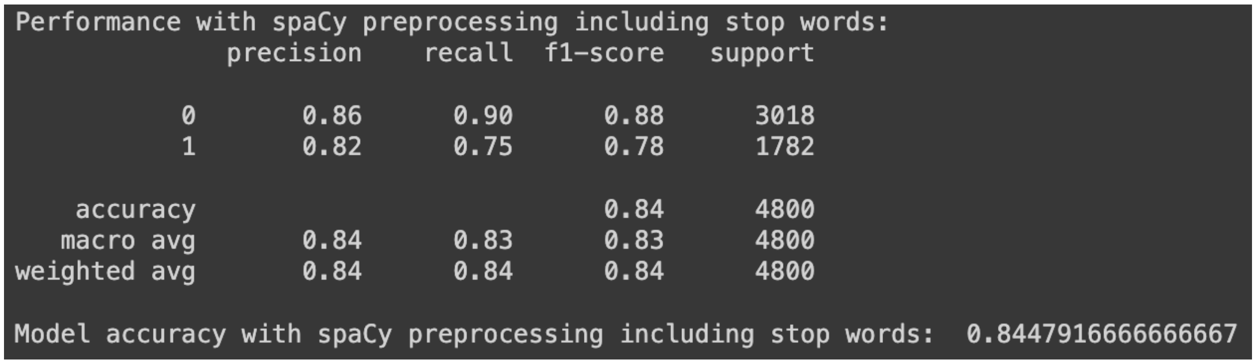
```
Performance with Byte Pair Encoding and Logistic Regression:
              precision    recall  f1-score   support

           0       0.76      0.86      0.81      3018
           1       0.70      0.53      0.60      1782

    accuracy                           0.74      4800
   macro avg       0.73      0.70      0.71      4800
weighted avg       0.74      0.74      0.73      4800

Model accuracy with Byte Pair Encoding and Logistic Regression:  0.7414583333333333
```

Appendix 4



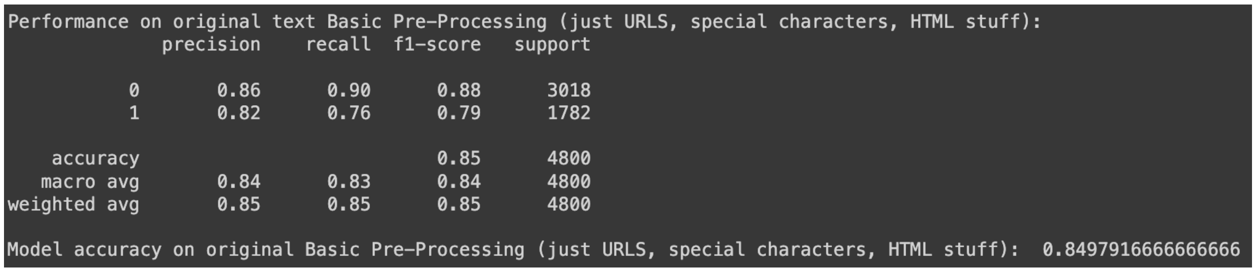
```
Performance with spaCy preprocessing:
              precision    recall  f1-score   support

           0       0.85      0.88      0.86      3018
           1       0.78      0.73      0.75      1782

    accuracy                           0.82      4800
   macro avg       0.81      0.80      0.81      4800
weighted avg       0.82      0.82      0.82      4800

Model accuracy with spaCy preprocessing:  0.8233333333333334
```

Appendix 5



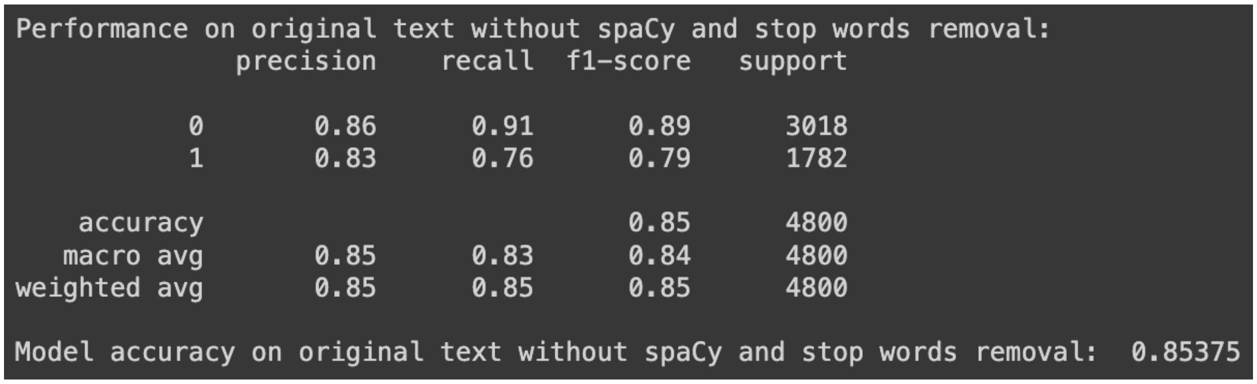
```
Performance with spaCy preprocessing including stop words:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      3018
           1       0.82      0.75      0.78      1782

    accuracy                           0.84      4800
   macro avg       0.84      0.83      0.83      4800
weighted avg       0.84      0.84      0.84      4800

Model accuracy with spaCy preprocessing including stop words:  0.8447916666666667
```

Appendix 6



```
Performance on original text Basic Pre-Processing (just URLS, special characters, HTML stuff):
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      3018
           1       0.82      0.76      0.79      1782

    accuracy                           0.85      4800
   macro avg       0.84      0.83      0.84      4800
weighted avg       0.85      0.85      0.85      4800

Model accuracy on original Basic Pre-Processing (just URLS, special characters, HTML stuff):  0.8497916666666666
```

Appendix 7



```
Performance on original text without spaCy and stop words removal:
              precision    recall  f1-score   support

           0       0.86      0.91      0.89      3018
           1       0.83      0.76      0.79      1782

    accuracy                           0.85      4800
   macro avg       0.85      0.83      0.84      4800
weighted avg       0.85      0.85      0.85      4800

Model accuracy on original text without spaCy and stop words removal:  0.85375
```