Implementing and Evaluating NLP Application Models using Python

Dominic Boy P. Almazan

IT Elective IV, BSIT

Jose Rizal University

Mandaluyong, Philippines

dominicboy.almazan@my.jru.edu

***Abstract*—This study investigates the implementation of four natural language processing (NLP) tasks text classification, sentiment analysis, part-of-speech (POS) tagging, and text summarization using both manually labeled and sourced datasets. Each model was trained and evaluated using appropriate preprocessing techniques and tailored hyperparameters to optimize performance. The text classification and sentiment analysis models demonstrated high precision and F1 scores, highlighting the importance of accurate manual labeling. POS tagging on COVID-19-related tweets achieved reliable token-level predictions, while extractive text summarization produced concise and informative summaries, evaluated using ROUGE metrics. The study underscores the critical role of data quality, careful preprocessing, and hyperparameter selection in achieving robust NLP models. Iterative experimentation and trial-and-error processes were essential in optimizing model performance, and future work may explore alternative pipelines and hyperparameter configurations to further enhance model generalization.**

Keywords—Natural Language Processing, Text Classification, Sentiment Analysis, Part-of-Speech Tagging, Text Summarization, SpaCy, ROUGE Metrics, Hyperparameter Tuning, Data Preprocessing

I. INTRODUCTION

For years, one significant goal of many researchers in artificial intelligence in general, is to enable computers to understand and process human languages. With models like Natural Language Processing (NLP), researchers and developers moved from a limited knowledge of the mechanics of how machines might help us process language to comprehensive tools that perform many of the task. For example, by using spaCy simply creates a convenient and user-friendly framework for developers to analyze plain text. From the effective detection of undesired spam emails to creating a succinct summary of several hundred articles, we see NLP systems become part of our communication technologies, digital assistants, and data analysis systems[1].

Keep in mind, however, that simply to build NLP systems, the evaluation piece is as critical in the overall process. In terms of classification tasks, we have accuracy, precision, recall, and F1-score that allow for us to evaluate effectiveness. Summarization task with ROUGE (Recall Oriented Understudy for Gisting Evaluation) is viewed widely as one of the standard metrics that can allow for evaluation[4] of quality of machine-generated summaries against human-generated summaries.

A significant aspect of building trustworthy NLP applications is the use of pipelines, which create a consistent workflow for processing language data. Typically, a pipeline includes a number of steps that are always: text cleanup, tokenisation, part-of-speech tagging, feature extraction, model training, and evaluation. Not only does the pipeline methodology moderate the tasks within an NLP system and allow for reproducibility; it also allows researchers to optimise single processing stages independently; sometimes resulting in better performance across multiple NLP tasks [2]. The modularity of pre-processing is especially relevant when transitioning from small experiments to professional implementations. In addition to accuracy, pipeline design also considers documentation, evaluation metrics, and reproducibility.

In this study,the researcher outline a methodical, five-step process for creating a gold-standard annotation set and evaluating spaCy's NER model on curated domain-specific text samples. We uncover modeling capabilities and limitations with spaCy's NER model while creating our gold-standard annotation set and basing performance evaluation on accuracy, precision, recall, and F1-score metrics. We believe we are also undertaking a comprehensive evaluation which is prudent for the safe and effective deployment of NER tools in specialized domains in the real world.

II. METHODOLOGY

A. Datasets of Sourced Data and Own Dataset

The dataset source taken from Kaggle, contains tweets related to COVID-19 that were labeled for a range of NLP tasks. Each tweet in this dataset contains both raw and preprocessed text, both tokenized versions of the text and associated POS tags. This dataset contains authentic textual data that can be used for POS tagging and extractive text summarization, providing the models with authentic social media data while assessing and learning on noisey and linguistically diverse text.

- OriginalTweet:
  The raw, unprocessed tweet text as collected from the Kaggle dataset.
- Clean_Tweet:

The tweet after text cleaning, including removal of special characters, extra whitespace, and normalization.

- NoStop_Tweet:

The cleaned tweet split into individual tokens (words) for SpaCy processing.

- Tokenized_Tweet:
  The cleaned tweet split into individual tokens (words) for SpaCy processing.
- POS Tags:
  Part-of-speech tags corresponding to each token in Tokenized_Tweet, used for evaluation of POS tagging.

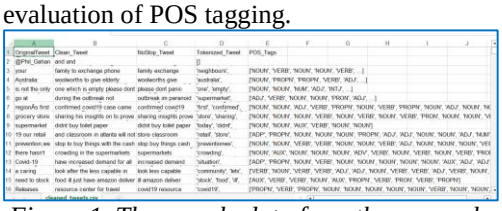


Figure 1. The sample data from the sourced data

This dataset was manually created and labeled for the study, and it complements the sourced COVID-19 tweet datasets by providing controlled examples for text classification and sentiment analysis tasks.

- UserName:

Name of the email sender.

- ScreenName:

Email sender's screen name or alias.

- Location:

Location of the sender,if available.

- EmailDate

Date when the email was sent.

- EmailContent

Original raw content of the email

- CleanedEmail

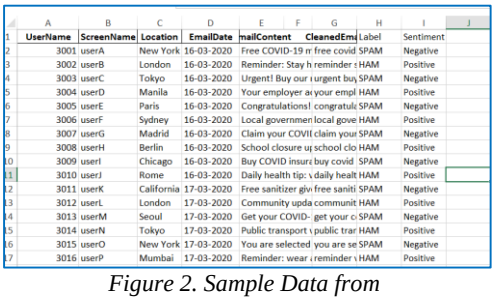
Preprocessed email with noised removed and normalized NLP task

- Label

Classification label indicating whether the email is SPAM or HAM.

- Sentiment

Sentiment label of the email text,sandarized as POSITIVE or NEGATIVE.



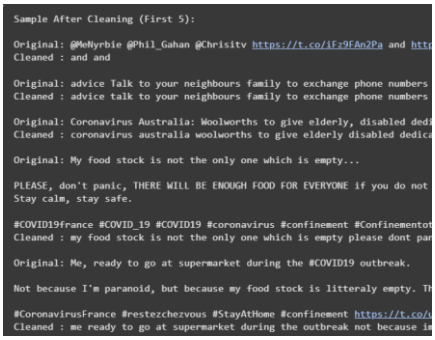
	A	B	C	D	E	F	G	H	I
1	UserName	ScreenName	Location	EmailDate	mailContent	CleanedEma	Label	Sentiment	
2	3001	userA	New York	16-03-2020	Free COVID-19 m	free covid	SPAM	Negative	
3	3002	userB	London	16-03-2020	Reminder: Stay h	reminder :	HAM	Positive	
4	3003	userC	Tokyo	16-03-2020	Urgent! Buy our	urgent buy	SPAM	Negative	
5	3004	userD	Manila	16-03-2020	Your employer a	your empl	HAM	Positive	
6	3005	userE	Paris	16-03-2020	Congratulations!	congratula	SPAM	Negative	
7	3006	userF	Sydney	16-03-2020	Local governmen	local gove	HAM	Positive	
8	3007	userG	Madrid	16-03-2020	Claim your COVI	claim your	SPAM	Negative	
9	3008	userH	Berlin	16-03-2020	School closure u	school clo	HAM	Positive	
10	3009	userI	Chicago	16-03-2020	Buy COVID insura	buy covid	SPAM	Negative	
11	3010	userJ	Rome	16-03-2020	Daily health tip:	daily healt	HAM	Positive	
12	3011	userK	California	17-03-2020	Free sanitizer giv	free saniti	SPAM	Negative	
13	3012	userL	London	17-03-2020	Community upda	communit	HAM	Positive	
14	3013	userM	Seoul	17-03-2020	Get your COVID-	get your c	SPAM	Negative	
15	3014	userN	Tokyo	17-03-2020	Public transport u	public trar	HAM	Positive	
16	3015	userO	New York	17-03-2020	You are selected	you are se	SPAM	Negative	
17	3016	userP	Mumbai	17-03-2020	Reminder: wear a	reminder :	HAM	Positive	

Figure 2. Sample Data from AlmazanDominic_ManualData_Cleaned.xlsx

B. *Data Preprocessing*

Data preprocessing is a critical stage in the NLP pipeline, as it directly influences the performance and reliability of the models. Both the sourced dataset of COVID-19 tweets and the manually created email dataset underwent systematic preprocessing to remove noise, normalize content, and prepare the text for analysis.For the sourced dataset (COVID-19 tweets):

- Stopwords Removal: Eliminated common non-informative words (e.g., the, and, is) to focus on more meaningful tokens.
- Tokenization: Segmented text into individual tokens for analysis.
- Lowercasing: Converted all text to lowercase to avoid treating the same words with different cases as separate features.
- Removing URLs: Hyperlinks were removed since they do not contribute semantic meaning.
- Removing Hashtags (#): Hashtag symbols were stripped, while retaining the core words.
- Punctuation Removal: Deleted punctuation marks that introduce noise without semantic contribution.
- Removing "RT": The retweet marker "RT" was removed to reduce redundancy.
- Whitespace Cleaning: Extra spaces were normalized for cleaner token structures.

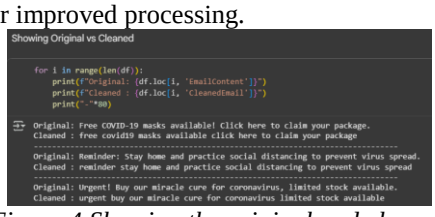


```
Sample After Cleaning (First 5):

Original: @MeNyrbie @Phil_Gahan @Chrisitv https://t.co/iFz9FAn2Pa and http
Cleaned : and and

Original: advice Talk to your neighbours family to exchange phone numbers
Cleaned : advice talk to your neighbours family to exchange phone numbers

Original: Coronavirus Australia: Woolworths to give elderly, disabled ded
Cleaned : coronavirus australia woolworths to give elderly disabled dedica

Original: My food stock is not the only one which is empty...

PLEASE, don't panic, THERE WILL BE ENOUGH FOOD FOR EVERYONE if you do not
Stay calm, stay safe.

#COVID19france #COVID_19 #COVID19 #coronavirus #confinement #Confinementot
Cleaned : my food stock is not the only one which is empty please dont par

Original: Me, ready to go at supermarket during the #COVID19 outbreak.

Not because I'm paranoid, but because my food stock is litteraly empty. Th

#CoronavirusFrance #restezchezvous #StayAtHome #confinement https://t.co/
Cleaned : me ready to go at supermarket during the outbreak not because i
```

Figure 3. Showing the original and cleaned data

For own Dataset:

- Lowercasing: Standardized text representation for uniformity.
- Removing URLs: Stripped hyperlinks from email content.
- Removing Hashtags: Any hashtag symbols were removed.
- Punctuation Removal: Cleansed punctuation that could interfere with tokenization.
- Removing "RT": Eliminated instances of the retweet marker where present.
- Whitespace Cleaning: Normalized spacing for improved processing.



```
Showing Original vs Cleaned

for i in range(len(df)):
    print(f"Original: {df.loc[i, 'EmailContent']}")
    print(f"Cleaned : {df.loc[i, 'CleanedEmail']}")
    print("-"*80)

Original: Free COVID-19 masks available! Click here to claim your package.
Cleaned : free covid19 masks available click here to claim your package
--------------------------------------------------------------------------------
Original: Reminder: Stay home and practice social distancing to prevent virus spread.
Cleaned : reminder stay home and practice social distancing to prevent virus spread
--------------------------------------------------------------------------------
Original: Urgent! Buy our miracle cure for coronavirus, limited stock available.
Cleaned : urgent buy our miracle cure for coronavirus limited stock available
```

Figure 4 Showing the original and cleaned data

Through the application of these preprocessing steps, the two datasets were converted into cleaner and more structured textual inputs. This preparation not only minimized irrelevant noise but also allowed the models to

interpret the data with greater consistency. By standardizing elements such as casing, punctuation, and spacing, and by removing non-informative content like URLs and retweet markers, the data became more representative of its underlying linguistic features. As observed in the experimental results, these refinements contributed directly to improvements in model performance, particularly in terms of accuracy, precision, and F1 scores across the classification, sentiment analysis, part-of-speech tagging, and summarization tasks.

C. *Model Implementation*

Four distinct NLP models were implemented in this study, each targeting a specific task: spam detection, sentiment analysis, part-of-speech tagging, and text summarization. Standard evaluation metrics (accuracy, precision, recall, and F1-score) were applied to assess performance. Hyperparameters were explicitly tuned only in the POS Tagging and Text Summarization models, due to the complexity of their tasks and the noisy nature of social media data.

- Text Classification:

The spam detection model was developed via the SpaCy 'textcat' pipeline and trained with a manually tagged dataset of email messages that were designated as: SPAM or HAM. In alignment with SpaCy's supervised training plan, the spam detection model applied a supervised learning approach to update weights as it 'saw' each annotated instance of either SPAM or HAM until convergence.

```
# ====================
# Prepare training data
# ====================

# Initialize an empty list to hold training examples
train_data = []

# Loop through each email text and its corresponding label
for text, label in zip(X_train, y_train):
    # If the label is SPAM, encode it with {"SPAM":1, "HAM":0}
    if label.upper() == "SPAM":
        train_data.append((text, {"cats": {"SPAM": 1, "HAM": 0}}))
    # Otherwise, encode it with {"SPAM":0, "HAM":1}
    else:
        train_data.append((text, {"cats": {"SPAM": 0, "HAM": 1}}))
```

Figure 5. Code snipet of text classification

- Sentiment Analysis:

The sentiment model was created using the same pipeline that was used for the spam classifier, however, it learned on a manually labeled dataset using POSITIVE and NEGATIVE labels instead. One critical change to the training setup was the number of epochs. The default 10 epochs were changed to 30 epochs, which increased the model's accuracy and recall from the baseline to a much greater degree.

```
# ====================
# Prepare training data
# ====================

# Initialize an empty list to hold training examples
train_data = []

# Loop through each training text and its corresponding label
for text, label in zip(X_train, y_train):
    if label == "POSITIVE":
        # If label is POSITIVE → assign 1 to POSITIVE and 0 to NEGATIVE
        train_data.append((text, {"cats": {"POSITIVE": 1, "NEGATIVE": 0}}))
    else:
        # If label is NEGATIVE → assign 1 to NEGATIVE and 0 to POSITIVE
        train_data.append((text, {"cats": {"POSITIVE": 0, "NEGATIVE": 1}}))
```

Figure 6. Code snipet of sentiment analysis

- POS Tagger:

To develop the part-of-speech tagging model, SpaCy's pre-trained English model, en_core_web_sm was used, and the model was adapted and optimized for processing the COVID-19 tweet source dataset by searching the best hyperparameter value for the model. The batch size was set at 32, accountingly for processing efficiencies, while Named Entity Recognition (NER) was turned off since during something like tagging, entity recognition would not be needed. The maximum text length when used in is greater significantly greater than 1M, try 2, 000 000 for the maximum text length so that the text input would be read completely without anything wrong with it (i.e., truncation error). The above changes enabled the model to produce only model-level token grammatical tagging while marginally decreasing processing time on real-world, noisy data.

```
# ----------------------------
# POS Analysis Function
# ----------------------------
def analyze_text(text):
    # Process the input text using the SpaCy NLP pipeline
    doc = nlp(text)

    # Extract each token and its corresponding POS tag into a list of tuples
    pos_list = [(token.text, token.pos_) for token in doc]

    # Return the list of (word, POS) pairs
    return pos_list
```

Figure 7. Code snipet of Part-of-speech tagging

- Text Summarizer:

The summarizing system was still extractive in nature, and key sentences were selected through statistical scoring. In this instance, tweets were relatively short, therefore the system was configured to extract one sentence per summary (N_SENTENCES = 1). Stopwords were also retained (REMOVE_STOPWORDS = False) to help preserve context within the sentence. Minimum token frequency (MIN_TOKEN_FREQ = 5), and minimum sentence length (MIN_SENT_LEN = 1) thresholds were also applied making sure meaningful sentences were included. Therefore, the summarizer supported concise outputs, while being coherent, with the added challenge of noisy text within social media.

```
# ============================
# Summarization of text
# ============================
def summarize(text, n_sentences=2):
    doc = nlp(str(text))
    sentence_scores = Counter()

    # Score sentences based on token frequency
    for sent in doc.sents:
        for token in sent:
            if not token.is_stop and not token.is_punct:
                sentence_scores[sent] += 1
    # Select top N sentences
    top_sentences = [sent.text for sent, score in sentence_scores.most_common(n_sentences)]
    return " ".join(top_sentences)
```

Figure 8. Code sniper ot Text Summarization

In summary, while the Spam and Sentiment Classifiers required minimal customization beyond iteration tuning, the POS Tagger and Summarizer depended heavily on hyperparameter adjustments to achieve reliable performance in domain-specific, real-world data.

III. RESULTS

The schema for the experimental study consisted of four sets of natural language processing tasks. Text classification (spam classification), sentiment classification classification, part-of-speech (POS) tagging, and extractive text summarization. The models were trained or validated with the datasets described in Section 2.1 which included both manually labelled datasets and externally sourced datasets. The models' performances were evaluated quantitatively against the classification performance metrics of accuracy, precision, recall, and F1 score. For the text summarization task, addition ROUGE metrics were used to evaluate the informativeness / quality of the summaries. The evaluation measures described above provide an overall evaluation of each model, and their generalization effort across the four respective NLP tasks.

The first task that we implemented was the text classification task of spam detection using a manually labeled email dataset. Our text classifier built using SpaCy performed well, such that it accurately categorized most of the SPAM and HAM emails in the validation set. From our evaluation metrics we obtain an accuracy of 0.83, precision of 1.00, recall of 0.71 and F1 score of 0.82. While it is obvious that our classifier is highly precise (the precision of 1.00 indicates that the classifier perfectly recognized each email flagged in the SPAM category as SPAM), the recall of 0.71 demonstrates that some SPAM emails were labelled as HAM emails (indicating that the classifier is conservative in its classification). The F1 score provides balance to the performance measures and indicates that the model provides overall good performance in separating SPAM from HAM messages. The evaluation of the text classification model is shown in Table 3.1.

Text Classification Model Performance

Accuracy	0.83
Precision	1
Recall	0.71
F1- Score	0.82

Table 3.1 Result of Text Classification Model

The second task dealt with part-of-speech (POS) tagging, which we applied to a publicly available data set of tweets related to COVID-19. The model was evaluated at the token-level, and achieved a token accuracy of 0.816, a precision of 0.724, a recall of 0.753, and a F1 score of 0.729. These results indicate that the model was able to correctly identify the grammatical roles of most tokens, but as expected there was some variability in performance as tweets are very noisy.

I implemented specific hyperparameters during training for performance optimization: a batch size of 32 so that the data could be processed in batches, i turned off the Named Entity Recogniton (NER) to limit the calculation resources, and i set the maximum text length of 2,000,000 characters in order to effectively deal with long repetitive inputs, rather than truncation. All of these conditions were very helpful for the model's strong performance of token-level POS tagging predictions. An overview tabulation of the evaluation headings for the POS tagging model is included in Table 3.2.

Part-Of-Speech (POS) Model Performance

Accuracy	0.81
Precision	0.72
Recall	0.75
F1- Score	0.72

Table 3.2 Result of Part-Of-Speech Model

The third occurrence that we carried out was a sentiment analysis using a manually labeled email texts data set that classified each email text as POSITIVE and NEGATIVE. The SpaCy text classifier achieved 0.917 accuracy, 0.800 recall, and 0.889 F1 score on the validation set. Therefore, we can conclude that the model is reliable in distinguishing the positive and negative sentiment and the F1 score being high means that the model was using a good balance of precision to recall .

The increase from 10 to 30 training iterations performed well for the model, which contributed to the convergence of the classifier and allowed the classification model to extract more nuances from the data resulting in statistical improvements in recall and F1 score. This highlights the relevance of training cycles when conducting sentiment classification. Table 3.3 contains the summary evaluation metrics for the the sentiment analysis model.

Sentiment Analysis Model Performance

Accuracy	0.91
Precision	1
Recall	0.88
F1- Score	0.88

Table 3.3 Result of Sentiment Analysis Model

The fourth task was similar to extractive text summarization and applied to a sourced dataset. The summarization model was evaluated by ROUGE metrics (Recall-Oriented Understudy for Gisting Evaluation), which evaluate how well the summaries generated by the model resemble the reference summarization. The model achieved a mean ROUGE-1 F1 score of 0.864, a ROUGE-2 F1 score of 0.829, and a ROUGE-L F1 score of 0.864. This means that the summarizations provided by the model captured the important words as well as the structure of the tweets from which they were extracted.

In my testing, it was apparent that the Summarization process was optimized with different hyperparameters, specifically with the number of sentences (N_SENTENCES) set to 1 to match the brevity of a tweet, keeping stopwords in order to better match the content, and determining minimum token frequency (MIN_TOKEN_FREQ = 5) and sentence length (MIN_SENT_LEN = 1) to capture a meaningful summarization. These settings greatly enhanced the model's performance, assuring capabilities for supported summarizations even while generating from some short and noisy social media texts. The evaluation metrics for the text summarization model are in table 3.4.

Text Summarization Model Performance

Rouge-1 F1	0.86
Rouge-2 F1	0.82
Rouge-3 F1	0.86

Table 3.4 Result of Text Summarization Model

IV. DISCUSSION

The results of the experiments strongly suggest that data preprocessing is a crucial part in natural language processing. It was important to clean the datasets during all four models, such as removing noise, and text normalising/ tokenisation made a visible impact on model performance. For example, there were some emails or tweets that included different formatting not normal characters to describe information and using agents could not interpret the style properly. This made it easier for the classifiers and taggers to recognize genuine features in a cleaned text and thus most of our evaluation metrics increased by significant amounts.

Additionally, the strategic use of hyperparameters played a key role in optimizing model performance. For the POS tagging task, hyperparameters such as a batch size of 32, disabling the Named Entity Recognition (NER) component, and increasing the maximum text length enabled efficient processing while maintaining accuracy across longer inputs. Similarly, for text summarization, hyperparameters including N_SENTENCES = 1, MIN_TOKEN_FREQ = 5, and MIN_SENT_LEN = 1 helped the model extract concise and informative summaries from noisy social media data. These adjustments demonstrate the sensitivity of NLP models to hyperparameter selection and the importance of tailoring them to the specific characteristics of the dataset.

A significant portion of the workflow also involved trial-and-error experimentation and meticulous manual labeling of the training data. For the text classification and sentiment analysis tasks, manually labeling emails ensured that the model had accurate and reliable examples to learn from, which directly contributed to higher precision and F1 scores. The iterative process of refining preprocessing steps, adjusting hyperparameters, and validating model outputs underscores the importance of careful experimentation in achieving robust and generalizable NLP models.

These observations emphasize that both data quality and model configuration are pivotal for developing effective NLP pipelines. The combination of clean, well-labeled datasets with carefully tuned hyperparameters leads to models capable of delivering reliable performance across multiple natural language processing tasks.

V. CONCLUSION

This study explored the implementation of four natural language processing tasks: text classification, sentiment analysis, part-of-speech tagging, and text summarization. The experimental results demonstrated that careful data preprocessing, including cleaning, tokenization, and normalization, is crucial for achieving reliable model performance. The use of hyperparameters tailored to each task such as batch size and component disabling for POS tagging, and sentence selection parameters for summarization further enhanced the effectiveness of the models.

A notable aspect of the research was the iterative trial-and-error process, particularly in manually labeling data and fine-tuning preprocessing steps, which significantly contributed to the observed improvements in evaluation metrics. This highlights the hands-on effort required to develop robust NLP pipelines.

Given more time, the researcher would explore alternative model architectures, different pipeline configurations, and a wider range of hyperparameters to further optimize performance. Such experimentation could potentially improve model generalization and adaptability to diverse textual datasets. Overall, this study underscores the importance of data quality, careful experimentation, and informed hyperparameter selection in developing effective NLP models.

VI. REFERENCES

[1] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural Language Processing: State of the Art, Current Trends and Challenges,” Multimedia Tools and Applications, vol. 82, no. 3, pp. 3713-3744, 2022. Available:https://www.researchgate.net/publication/319164243_Natural_language_processing_state_of_the_art_current_trends_and_challenges

[2] B. B. Elov, Sh. M. Khamroev, and Z. Y. Xusainova, “The pipeline processing of NLP,” E3S Web of Conferences, INTERAGROMASH 2023, Tashkent, Uzbekistan. Available via ResearchGate: https://www.researchgate.net/publication/373072593_The_pipeline_processing_of_NLP

[3] “COVID-19 Tweets NLP Text Classification,” Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?select=Corona_NLP_train.csv