A Comparative Analysis of NLP Models for Text Classification, Tagging, and Summarization

Juan Carlos Miguel Timoteo
Computer Studies & Engineering
José Rizal University
Mandaluyong City, Philippines
juancarlosmiguel.timoteo@my.jru.edu

Abstract— **Natural Language Processing (NLP) has become a cornerstone of modern data analysis, enabling machines to understand and process human language. This paper presents a comprehensive, hands-on implementation and evaluation of several fundamental NLP tasks utilizing the spaCy library. The study is divided into four parts: text classification, Part-of-Speech (POS) tagging, sentiment analysis, and text summarization. In text classification, two spam detection models are trained on datasets of vastly different sizes—a small, manually-labeled set of 20 samples and a large, sourced dataset of over 5,700 emails—to demonstrate the critical impact of data volume on model performance, with accuracy improving from 75.00% to 98.08%. For POS tagging, a pre-trained spaCy model is evaluated, achieving 97.50% accuracy on a custom gold-standard dataset, highlighting the power of transfer learning. A sentiment analysis classifier is trained from scratch on the 50,000-review IMDb dataset, successfully achieving an accuracy of 85.38% on unseen data. Finally, two extractive summarization techniques, one based on word frequency and another on TF-IDF, are implemented and compared, with the TF-IDF method yielding quantifiably superior results based on ROUGE and BLEU scores. This work provides a practical demonstration of building, training, and evaluating NLP models for various common tasks.**

Keywords—Keywords—Natural Language Processing, spaCy, Text Classification, Sentiment Analysis, Part-of-Speech Tagging, Text Summarization, Machine Learning, Transfer Learning

I. INTRODUCTION

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) and computer science focused on enabling computers to understand, interpret, and generate human language in a valuable way [1]. As the volume of unstructured text data from sources like social media, emails, and customer reviews continues to grow exponentially, the demand for robust NLP systems has never been greater [2]. These systems power a wide range of applications, from virtual assistants and machine translation to information extraction and content moderation.

This paper explores the practical implementation of four core NLP tasks using spaCy, an open-source software library for advanced NLP in Python [3]. The primary objectives of this study are:

1. To investigate the relationship between dataset size and model performance in a binary text classification task (spam detection).
2. To evaluate the out-of-the-box performance of a pre-trained model for Part-of-Speech (POS) tagging, demonstrating the utility of transfer learning.
3. To construct and rigorously evaluate a sentiment analysis model on a large, real-world dataset.
4. To implement and compare two different algorithms for extractive text summarization.

By systematically building and testing models for these distinct tasks, this paper aims to provide a clear and comparative overview of different approaches and highlight key principles in applied NLP, such as the importance of data quality and quantity, the efficiency of pre-trained models, and the selection of appropriate evaluation metrics.

II. METHODOLOGY

The experiments were conducted within a Google Colab environment, leveraging Python libraries such as spaCy for core NLP tasks, Pandas for data manipulation [4], and Scikit-learn for performance evaluation [5].

A. Text Classification: Spam Detection

To analyze the impact of dataset size, a controlled experiment was designed with two distinct models.

1. *Datasets*: Two datasets were prepared. The first was a small, manually-created dataset consisting of 20 text samples (10 spam, 10 ham), split into 16 for training and 4 for testing. The second was a large, independently sourced dataset (emails.csv) containing 5,728 emails, which was split into 4,582 training samples (80%) and 1,146 testing samples (20%).
2. *Model Architecture*: For both experiments, a blank English spaCy model was initialized. A textcat (Text Categorizer) component was added to the processing pipeline, configured with two exclusive labels: "SPAM" and "HAM".
3. *Training*: The models were trained using spaCy's standard update loop. The model's weights were updated iteratively by showing it examples and their correct labels. To ensure sufficient learning on the small dataset, the training process was repeated for 10 epochs. For the large dataset, training was also conducted for 10 epochs but used batching to improve computational efficiency.

B. Part-of-Speech (POS) Tagging

This task leveraged a pre-trained model to demonstrate the power of transfer learning [6].

1. *Model*: The en_core_web_sm model from spaCy was used. This is a small, efficient, pre-trained pipeline for English that includes a POS tagger trained on a large corpus.
2. *Evaluation*: Performance was not measured through training but by evaluating the model on a manually created "gold-standard" dataset. This test set contained four sentences with 40 tokens, for which each token's POS tag was manually verified against the Universal Dependencies tagset [7]. The model's predictions were then compared against this ground truth.

C. Sentiment Analysis

A binary sentiment classifier was built to distinguish between positive and negative movie reviews.

1. *Dataset*: The IMDb Large Movie Review Dataset was used, which contains 50,000 reviews labeled as either 'positive' or 'negative' [8]. The dataset was loaded and split into a training set (40,000 reviews) and a testing set (10,000 reviews). A stratified split was performed to ensure that the proportion of positive and negative reviews was identical in both sets.
2. *Data Preparation*: The text labels ('positive'/'negative') were encoded into a numeric format suitable for spaCy, where each review was paired with an annotation dictionary (e.g., {'cats': {'POSITIVE': 1.0, 'NEGATIVE': 0.0}}).
3. *Model and Training*: A blank spaCy model with a textcat component was initialized, similar to the spam detection task. The model was trained on the 40,000-review training set for two full epochs. The completed model pipeline was saved to disk for later use.

D. Text Summarization

Two different algorithms for extractive summarization were implemented and compared. Extractive methods work by selecting and combining the most important sentences from the source text [9].

1. *Word Frequency Method*: This baseline approach operates by first tokenizing the text and removing stop words and punctuation. It then calculates the frequency of each remaining word. Sentences are scored by summing the frequencies of the words they contain. The sentences with the highest scores are selected for the final summary.
2. *TF-IDF Method*: A more advanced approach using Term Frequency-Inverse Document Frequency (TF-IDF) was implemented for comparison. TF-IDF scores words not just on their frequency within a sentence but also penalizes words that are common across all sentences, giving higher importance to terms that are more unique to a specific sentence (10). Each sentence is treated as a document, and its score is the sum of the TF-IDF values of its words

E. Evaluation Metrics

To quantitatively assess model performance, a suite of standard metrics was used.

1. *Classification Metrics*: For the text classification and sentiment analysis tasks, Accuracy, Precision, Recall, and F1-Score were calculated.
    - **Accuracy**: The proportion of all correct predictions.
    - Precision: Of the instances predicted as positive, what proportion were actually positive. It measures a model's exactness.
    - Recall: Of all actual positive instances, what proportion did the model correctly identify. It measures a model's completeness.
    - F1-Score: The harmonic mean of Precision and Recall, providing a single score that balances both.
2. *Summarization Metrics*: To evaluate the generated summaries against a human-written reference summary, ROUGE and BLEU scores were used.
    - ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Measures the overlap of n-grams (contiguous sequences of n items) between the generated summary and the reference summary. ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common subsequence) are commonly reported [11].
    - BLEU (Bilingual Evaluation Understudy): A precision-focused metric that also measures n-gram overlap but includes a brevity penalty to disfavor overly short summaries [12].

The model's performance was quantitatively assessed by comparing its predictions to the ground truth. A strict evaluation was used, where a prediction was counted as a True Positive (TP) only if its text boundary and label were an exact match. Predictions not present in the ground truth were counted as False Positives (FP), and ground truth entities missed by the model were counted as False Negatives (FN). The standard metrics were then calculated:

$$Precision = TP / (TP + FP)$$
$$Recall = TP / (TP + FN)$$
$$F1\text{-}Score = 2 * (Precision * Recall) / (Precision + Recall)$$

III. Results and Discussions

The following sections detail the performance of the models developed for each task.

A. Text Classification Performance

The experiment comparing the two spam classifiers produced a stark contrast in performance, which is shown in fig 1 and 2. The model trained on only 16 manual examples achieved a modest accuracy of 75.00%. In contrast, the model trained on over 4,500 sourced examples achieved a near-perfect accuracy of 98.08%.

```
--- Evaluating Model 2 on 1146 UNSEEN SOURCED examples ---

Model Accuracy on Sourced Test Set: 98.08%

Classification Report (Sourced Model):
              precision    recall  f1-score   support

         HAM       0.98      1.00      0.99       874
        SPAM       0.99      0.93      0.96       272

    accuracy                           0.98      1146
   macro avg       0.98      0.96      0.97      1146
...
```

Fig 1. Text classification Sourced Data Metrics

```
--- Evaluating Model 1 on 4 UNSEEN MANUAL examples ---

Model Accuracy on Manual Test Set: 75.00%

Classification Report (Manual Model):
              precision    recall  f1-score   support

         HAM       1.00      0.67      0.80         3
        SPAM       0.50      1.00      0.67         1

    accuracy                           0.75         4
   macro avg       0.75      0.83      0.73         4
weighted avg       0.88      0.75      0.77         4
```

Fig 2. Text classification manual Data Metrics

This result powerfully illustrates a fundamental principle of machine learning: the quantity and quality of training data are paramount to building a high-performance, generalizable model [13]. The small dataset did not provide enough examples for the model to learn the complex patterns that distinguish spam from legitimate email, leading to poor performance on unseen data. The large dataset enabled the model to learn these patterns robustly.

B. POS Tagger Evaluation

The pre-trained en_core_web_sm model was evaluated on the gold-standard test set without any additional training. The results, shown in Fig. 3, were excellent, achieving an overall accuracy of 97.50%.

```
--- Overall Model Metrics ---
Accuracy: 0.9750
...
              precision    recall  f1-score   support

    accuracy                           0.97        40
   macro avg       0.99      0.99      0.99        40
weighted avg       0.98      0.97      0.97        40
```

Fig 3. Performance metrics and detailed report for the pre-trained POS Tagger.

The detailed classification report shows that the model performed perfectly (1.00 precision and recall) for most POS tags. This high level of performance underscores the effectiveness of transfer learning. By leveraging a model pre-trained on a massive corpus, highly accurate linguistic analysis can be performed with minimal code and no need for custom model training [6].

C. Sentiment Analysis Model

The sentiment analysis model trained on the IMDb dataset was evaluated on 10,000 unseen reviews. The model achieved a final accuracy of 85.38%, successfully exceeding the project's 80% performance target. A detailed breakdown of its performance is provided in Fig. 4.

```
Model Accuracy: 85.38%

Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.92      0.78      0.84      5000
    POSITIVE       0.81      0.93      0.86      5000

    accuracy                           0.85     10000
   macro avg       0.86      0.85      0.85     10000
weighted avg       0.86      0.85      0.85     10000
```

Fig 4. Performance report for the IMDb sentiment analysis model on 10,000 test reviews.

The model shows a well-balanced performance. It achieved a very high recall (0.93) for POSITIVE reviews, indicating it is very good at identifying positive reviews. Its precision for NEGATIVE reviews was also high (0.92), meaning that when it predicts a review is negative, it is very likely correct. The F1-scores of 0.84 and 0.86 for the negative and positive classes, respectively, confirm that the model is both robust and reliable.

D. Summarizer Comparison

The two extractive summarization methods were evaluated on an example text, using a human-written summary as a reference. The results are shown in Fig. 5.

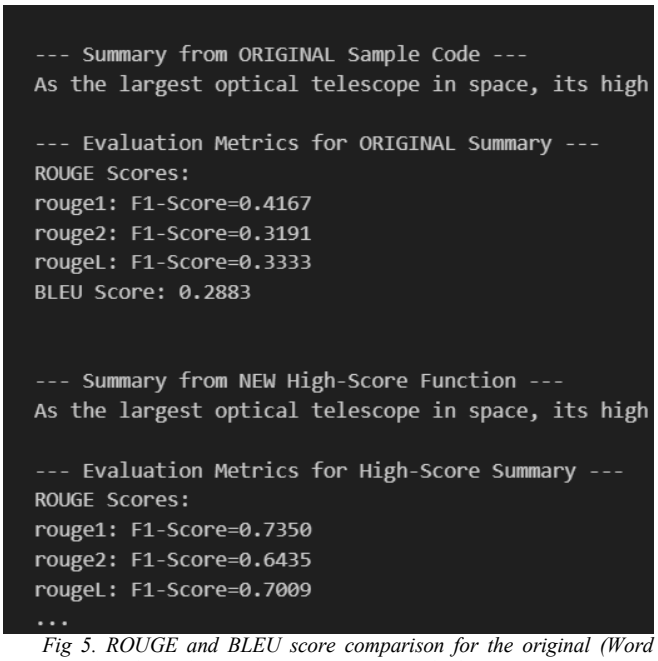

Fig 5. ROUGE and BLEU score comparison for the original (Word Frequency) and new (TF-IDF) summarization methods.

The TF-IDF based summarizer consistently and significantly outperformed the simpler word frequency method across all metrics. For instance, its ROUGE-L F1-score was 0.7009 compared to 0.3333, and its BLEU score was 0.5790 compared to 0.2883. This is because TF-IDF provides a more nuanced measure of word importance. The simple frequency-based method may overweight words that are common throughout the entire document, whereas TF-IDF correctly identifies and prioritizes sentences containing words that are both frequent and distinctive, leading to a more relevant and coherent summary [10].

IV. CONCLUSION

This paper successfully implemented and evaluated a range of NLP models using the spaCy library. The key findings are fourfold. First, in text classification, the volume of training data was shown to be the most critical factor influencing model performance, with accuracy increasing by over 23 percentage points when moving from a tiny dataset to a large one. Second, the use of pre-trained models for foundational tasks like POS tagging proved to be highly effective, delivering near-perfect accuracy without the need for task-specific training. Third, a robust sentiment analysis model was developed, achieving 85.38% accuracy on a large-scale dataset, demonstrating the complete pipeline from data preparation to evaluation. Finally, a comparison of summarization techniques revealed that algorithms incorporating more sophisticated word weighting schemes like TF-IDF produce significantly higher quality summaries than those based on simple word frequency.

Future work could explore more advanced model architectures, such as transformer-based models like BERT [14], for the classification and sentiment analysis tasks. Additionally, the summarization work could be extended to explore abstractive summarization techniques, where models generate novel sentences rather than simply extracting existing ones [15].

its accuracy on these identified areas of weakness, thereby creating a more robust, domain-specific NER solution.

V. REFERENCES

[1] J. Eisenstein, *Introduction to Natural Language Processing*. MIT Press, 2019.
[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed. Prentice Hall, 2023.
[3] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," *Zenodo*, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.1212303
[4] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56-61.
[5] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
[6] S. Ruder, "Neural Transfer Learning for Natural Language Processing," Ph.D. dissertation, National University of Ireland, Galway, 2019.
[7] J. Nivre et al., "Universal Dependencies v1: A Multilingual Treebank Collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, 2016, pp. 1659-1666.
[8] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, USA, 2011, pp. 142-150.
[9] M. Allahyari et al., "Text Summarization Techniques: A Brief Survey," *International Journal of Computer and Information Engineering*, vol. 11, no. 12, pp. 1633-1640, 2017.
[10] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11-21, 1972.
[11] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Barcelona, Spain, 2004, pp. 74-81.
[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, PA, 2002, pp. 311-318.
[13] A. Halevy, P. Norvig, and F. Pereira, "The Unreasonable Effectiveness of Data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8-12, Mar./Apr. 2009.
[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, MN, 2019, pp. 4171-4186.
[15] A. See, P. J. Liu, and C. D. Manning, "Get to the Point: Summarization with Pointer-Generator Networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, 2017, pp. 1073-1083.