Linguistic Personality Prediction: An NLP Approach to MBTI Classification

Zeyuan Ding
Center for Data Science
New York University
zd2466@nyu.edu

Haitong Zhu
Center for Data Science
New York University
hz3916@nyu.edu

Yuan Tian
Center for Data Science
New York University
yt2077@nyu.edu

Xin Shu
Center for Data Science
New York University
xs2189@nyu.edu

Siling Xiao
Center for Data Science
New York University
sx2567@nyu.edu

1 Motivation

The Myers-Briggs Type Indicator (MBTI) is widely used in various domains such as business, career counseling, and team development. Traditional MBTI assessments typically rely on self-report questionnaires, which can be time-consuming and prone to response bias.

With the rise of social media and digital communication, individuals increasingly express themselves through text, offering a rich and natural source of behavioral data linked to personality traits. Leveraging advances in natural language processing (NLP), we can explore whether linguistic patterns in written communication correspond to MBTI classifications, potentially enabling more efficient and scalable personality assessment.

2 Dataset

We utilize a preprocessed MBTI (Myers-Briggs Type Indicator) dataset comprising approximately 106,000 text samples. Each data point consists of a 500-word excerpt representing user-generated content, labeled with the author's MBTI personality type. The dataset is constructed by aggregating and truncating posts from various online sources, ensuring uniform sample length.

The original data was sourced from two major contributors: Dylan Storey, who compiled a large corpus of approximately 1.7 million Reddit posts via Google BigQuery, and Mitchell Jolly (datasnaek), who contributed around 9,000 records from the PersonalityCafe forum, with each entry containing a user's most recent 50 posts.

The dataset is publicly available at https://www.kaggle.com/datasets/zeyadkhalid/mbti-personality-types-500-dataset.

2.1 Data Preprocessing

Although the existing dataset on Kaggle has been preprocessed, we find it important to replicate the preprocessing pipeline ourselves to ensure data completeness and consistency.

Furthermore, in addition to the original steps, we apply an additional filter to remove any text that explicitly mentions MBTI types, thereby preventing potential data leakage.

For the LSTM model, we employ a tokenizer to convert textual input into sequences of integers, enabling the model to process language data effectively. In contrast, for the Logistic Regression

The full source code is available at: `https://github.com/dingfengkun/NLP-Predictor-MBTI.git`

and Random Forest models, we utilize Term Frequency–Inverse Document Frequency (TF-IDF), a statistical technique that quantifies the importance of a word within a document relative to a larger corpus.

While TF-IDF does not capture word context as neural networks do, it offers advantages in terms of computational efficiency and model interpretability.

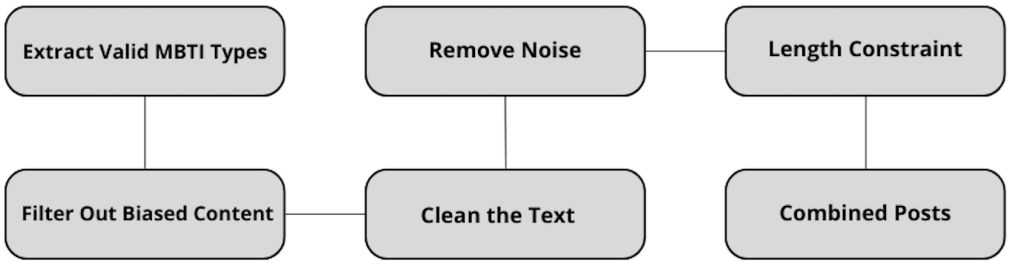

Figure 1: Data Preprocessing Pipeline

3 Exploratory Data Analysis (EDA)

We performed an initial exploratory data analysis (EDA) on the processed dataset and observed the following:

- The three most frequent personality types—INTP, INTJ, and INFJ—collectively account for approximately 50–60% of the dataset.
- Several personality types, such as ESFJ, ESFP, and ESTJ, are underrepresented, which may introduce bias in the learning process.
- The user posts exhibit substantial variation in both length and linguistic complexity.

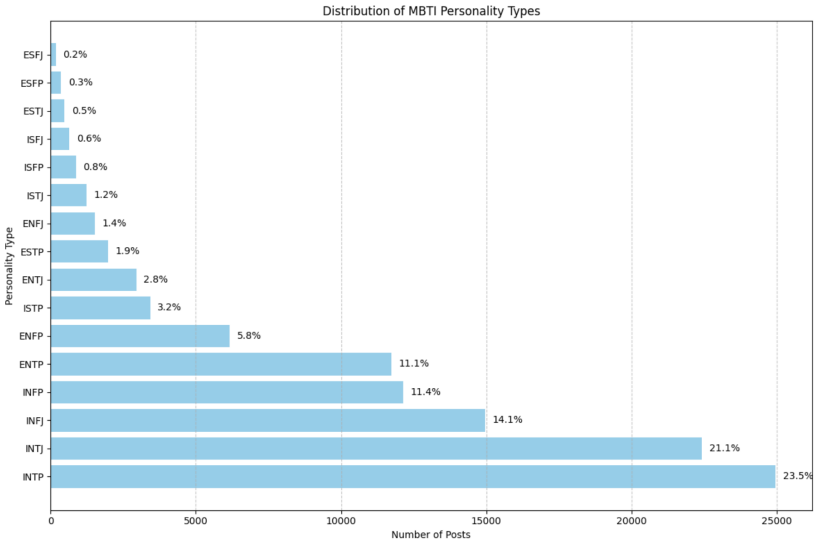

Figure 2: Distribution of MBTI Personality Types in the Dataset.

To address the issue of class imbalance, we adopt the weighted F1 score as our primary evaluation metric. This approach ensures that each class is given proportional importance by accounting for class frequency, thereby mitigating the effects of imbalance.

In addition, to manage the variability in text length and linguistic complexity, we aggregate all posts associated with the same MBTI personality type into a single data point, truncated to a maximum of 500 words.

4 Methodology

4.1 Logistic Regression

Logistic Regression is employed as a baseline model due to its simplicity, interpretability, and strong performance on high-dimensional sparse feature spaces like those generated by TF-IDF. Despite being a linear model, it often performs competitively in text classification tasks.

In our context, we chose Logistic Regression for the following reasons:

- **Efficiency**: It trains quickly, even on large sparse matrices, making it suitable for initial benchmarking.
- **Interpretability**: The learned coefficients provide insight into which words are most indicative of specific personality types.
- **Compatibility with TF-IDF**: Logistic Regression handles sparse, high-dimensional data effectively without requiring dimensionality reduction.
- **Robustness to Imbalance**: The model compensates for class imbalance during training by adjusting the loss weights. This is achieved through the option `class_weight='balanced'` in the implementation.

4.2 Random Forest

To further diversify our modeling strategies, we also employ Random Forest, an ensemble learning method based on decision trees. Compared with Logistic Regression, Random Forest can capture nonlinear relationships between features. It also requires minimal data normalization and offers basic interpretability. These help us understand the influence of certain words on MBTI classification. With strong robustness against overfitting, Random Forest have a good balance between interpretability and predictive power, making it suited for our experiment.

4.3 LSTM

Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN) designed to capture long-range dependencies in sequential data. By utilizing gating mechanisms, LSTM effectively addresses common issues such as vanishing and exploding gradients. In our task, the fixed-length input (up to 500 words) is well-suited for the LSTM architecture to identify and learn meaningful patterns across the sequence.

5 Results

Table 1: Comparison of Model Results

Model Name	Accuracy	Weighted F1-score
Logistic Regression	0.8298	0.83
Random Forest	0.5231	0.47
LSTM	0.8593	0.81

Table 1 presents the performance metrics of three classification models evaluated on the preprocessed dataset. Logistic Regression achieved an accuracy of 0.8298 and a weighted F1-score of 0.83, outperforming the Random Forest model, which attained an accuracy of 0.5231 and a weighted F1-score of 0.47. The LSTM model achieved the highest accuracy at 0.8593, though its weighted F1-score was slightly lower at 0.81, indicating potential imbalance in prediction performance across classes. While the LSTM was expected to outperform Logistic Regression due to its longer training

time and advanced sequential modeling capabilities, the comparatively lower F1-score suggests underutilization of its representational capacity. This performance gap may be attributed to limited hyperparameter tuning, constrained by available hardware resources. Furthermore, class imbalance in the dataset may have adversely affected the LSTM model more significantly than the Logistic Regression model.

6 Optimization

We applied optimization techniques to the following areas:

6.1 Preprocessing

To enhance the efficiency of large-scale text preprocessing, we incorporated multiprocessing to parallelize the cleaning operations across multiple CPU cores. Additionally, stopword localization was implemented to eliminate dependency on external internet sources, thereby reducing latency and ensuring consistency across runs. These adjustments collectively reduced preprocessing time from 85 seconds to 53 seconds, representing a 38% improvement in processing speed.

6.2 Logistic Regression

For the logistic regression model, optimization focused on reducing training time without compromising accuracy. The maximum number of iterations (max_iter) was decreased from 1000 to 300 to eliminate redundant computations. Moreover, full CPU parallelism was enabled using n_jobs=-1, which significantly accelerated model training. As a result, the execution time decreased from 189 seconds to 90 seconds, yielding a 52% improvement.

6.3 Random Forest

Optimization of the Random Forest model targeted the reduction of computational overhead associated with high-dimensional feature spaces. We employed Truncated Singular Value Decomposition (TruncatedSVD) to reduce the dimensionality of the TF-IDF feature matrix. This helps speed up training and improve generalization. In conjunction with enabling multi-threaded parallelism (n_jobs=-1) to utilize all available CPU cores during training. These changes dramatically decreased the training time from 1572 seconds to 35 seconds, corresponding to a 96% improvement in speed.

6.4 LSTM

To improve the training efficiency of the LSTM model, we replaced the original NumPy-based implementation with an optimized PyTorch version. This transition leveraged GPU acceleration and more efficient tensor operations inherent in the PyTorch framework. Despite maintaining equivalent classification accuracy, the model's execution time was reduced by 78%.

7 Conclusion

In the optimization process, the most effective techniques involved the utilization of C language-based packages and multiprocessing. Multiprocessing substantially accelerated data preprocessing tasks, particularly when working with large-scale datasets containing millions of records. For the modeling phase, we adopted high-performance libraries such as `PyTorch`, which is implemented in C/C++ and utilizes `CUDA`. This transition yielded significant runtime improvements, especially in the implementation of long short-term memory (LSTM) networks.

Although we investigated the use of `Cython` for additional optimization, it was observed that many core natural language processing (NLP) libraries are already built upon highly optimized C backends, resulting in limited potential for further performance gains in this area.

In summary, the strategic application of parallel processing and compiled language tools was critical in reducing computational time while maintaining model accuracy.