Language Detection and Identification in Natural Language Processing

Anonymous ACL submission

Abstract

In this paper, we will analyze a variety of more traditional machine learning (ML) models and many other deep learning (DL) ones as well. These models are used to accomplish a language detection task with the Language Detection dataset (Saji, 2021) from Kaggle. The goal was to compare the performance of the simpler models (traditional ML) versus the more complex ones (DL models). Our findings were quite surprising to us, given that the more simpler ML models perform quite well and often times perform better than the more complex DL ones. The models that were built and trained were a multinomial naive Bayes (MNB) with both N-gram representation and TF-IDF representation, a support vector machine (SVM) model, a multinomial logistic regression (MLR) model, a random forest (RF) model, a basic recurrent neural network (RNN) model, a simple LSTM model, a simple GRU model, a Bi-LSTM model, a Bi-GRU model, a Bi-LSTM/Bi-GRU model, a BERT-based transformer model, a convolutional neural network (CNN) model, and pre-trained libraries such as langdetect by Python and fastText by Facebook (Joulin et al., 2022). Variations of some of these models were also built. These variations can be found in our submitted code.

1 Introduction

The motivation behind this extensive research and analysis project is very simple. As a team, we thought it would be interesting to build and implement a variety of machine learning models and deep learning models that were either discussed in class or not at all and compare their performance on a certain NLP task. The chosen task was language detection. More precisely, given a sentence, the model must correctly identify the language in which the sentence is written. For example, if the model receives as input "Hi, how are you?" the model must output "English". Language detection was chosen given that many datasets already exist for this sort of task. Furthermore, we also wanted to see how certain models perform when not used on the English language but on other languages like French, German, Italian, and many others.

Our main research question is as follows: "What are the comparative strengths and weaknesses of traditional machine learning models versus deep learning models in identifying languages from short text samples?" while our hypothesis before starting the project was as follows: "Deep learning models like BiLSTMs outperform traditional machine learning models like Naive Bayes or SVM in language recognition tasks due to their ability to capture complex patterns in textual data."

As we shall see in this paper, surprising results are obtained. This hypothesis was tested in a very methodical manner. Indeed, as a team, we implemented a variety of different ML and DL models. After being implemented, all models were trained and tested on the same language detection dataset (Saji, 2021) which we will discuss in more depth in section 3.1. We then compared each model's accuracy with one another to try and determine which model was best performing. Other analysis methods were implemented such as classification reports and confusion matrices which can be found in our submitted code, however these were unfortunately not generated for all models.

2 Related work

In this section, we will be going over a bit of the literature present in this domain. We will also discuss how these works differ from ours.

2.1 Enriching Word Vectors with Subword Information (Bojanowski et al., 2017)

In this paper, Bojanowski et al. enhance traditional word embeddings by incorporating character-level subword information. Unlike conventional methods that treat words as standalone entities, their approach breaks words into character n-grams and

represents a word as the sum of its subword embeddings. This allows the model to handle out-of-vocabulary words and perform better on morphologically rich languages like German or Russian, where words have complex inflections. Their experiments demonstrate improved results for tasks like word similarity and syntactic analogies, particularly for rare words and small datasets. While effective at syntactic tasks, the model still faces challenges in fully capturing semantic relationships between words. This method is highly relevant for language identification tasks, especially for short texts or languages with limited data.

2.2 Bag of Tricks for Efficient Text Classification (Joulin et al., 2016b)

Joulin et al. propose fastText, an efficient and lightweight approach to text classification that combines a bag-of-words model with n-gram features and a simple linear classifier. The authors optimize training and prediction speed by embedding n-grams into a smaller feature space and using a hierarchical softmax to scale well for large datasets. Despite its simplicity, fastText achieves accuracy comparable to deep learning models while being significantly faster and more memory-efficient. The experiments demonstrate its effectiveness across multiple text classification tasks, proving that adding n-gram features improves performance. The paper shows that for practical applications where computational resources are limited, fastText can serve as a reliable and scalable alternative to deep learning models.

2.3 FastText.zip: Compressing Text Classification Models (Joulin et al., 2016a)

In this work, Joulin et al. introduce FastText.zip, an optimized version of fastText that focuses on compressing model size to make text classification more resource-efficient. By applying techniques like product quantization to compress word embeddings and vocabulary pruning to retain only important features, they significantly reduce the model's memory footprint. The experiments show that FastText.zip achieves compression rates of up to 1000x without compromising much on accuracy, making it suitable for low-resource environments such as mobile devices. The compressed models maintain competitive performance across various text classification benchmarks while being lightweight and fast to deploy. The study highlights that carefully optimized traditional models like FastText.zip can deliver practical, real-world efficiency without relying on the computational demands of deep learning.

2.4 Character-level Convolutional Networks for Text Classification (Zhang et al., 2016)

In this paper, the authors explore the effectiveness of Character-level CNNs for a variety of natural language processing tasks. They compare their CNN model against other more classical natural language methodologies such as bag of words models, n-gram models, TF-IDF representations as well as more complex models such as ConvNets and RNNs. Through the results of this paper, the authors show that CNNs can be quite effective for tasks like sentiment analysis and topic classification but the performance on these two tasks depend heavily on a multitude of factors. These factors include dataset size (where more data is ideal), choice of alphabet, the use of user-generated data, and whether or not the texts present in the dataset used for training and testing the models have been curated or not.

2.5 Differences

A lot of the papers summarized above do not discuss the task of language identification, which is the first major difference with what our project focuses on. Furthermore, most of these papers do not do an exhaustive comparison between all possible models that can be used for a task like language detection. In our case, we tried to test the most models possible to be able to compare their performance with one another. This would allow us to see how different models perform in general, but also on specific languages. Furthermore, for most of the papers discussed in the previous section, a lot of their focus and work was concentrated on making a very efficient and high-performing model, which is great, but it is not what we set our focus on. Our focus was simply on model performance and how different models perform on an identical task which is, in our case, language identification.

3 Method

In this section, we will be going over the dataset that we used in this project. We will also be discussing how we preprocessed our data, the environment in which we ran our experiments, and we will be discussing the models and experiments we ran on them.

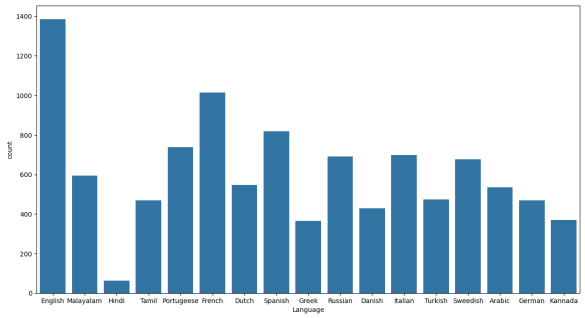

Figure 1: Distribution of the dataset entries based on their language.

3.1 Dataset

The dataset we used in this project is the Language Detection dataset (Saji, 2021) from Kaggle. This dataset is a rather small language dataset that contains about total of 10 000 entries for 17 different languages. Not all languages contain the same amount of entries. The 17 languages in this dataset are English, Malayalam, Hindi, Tamil, Kannada, French, Spanish, Portuguese, Italian, Russian, Swedish, Dutch, Arabic, Turkish, German, Danish and Greek. The author collected this language data by scraping it from Wikipedia. A visual representation of the dataset entries and how they are distributed can be viewed in figure 1.

3.2 Data Preprocessing and Environment

The dataset was preprocessed in a variety of ways. First of all, missing values within the dataset were dropped as well as duplicate values to clean up our data. Then, we proceeded to remove special characters and numbers from within the dataset given that these two types of characters are not language specific and would not aid in our language identification task. Finally, when appropriate, N-gram and TF-IDF vectorization representation were used to enhance model performance with certain models such as MNB, which will be discussed in section 4. Furthermore, when necessary for more complex models, target values were encoded using LabelEncoder() from Python. The datasets used were 60% for training, 20% for validation and 20% for testing. Finally, for our training and testing environment, all experiments and implementations were done in Google Colab using the Python programming language. The GPUs used to run our models were the ones offered by Google Colab. Lastly, labels were given to each of the 17 languages after preprocessing the dataset. Label 0 is associated to Arabic, label 1 to Danish, label 2 to Dutch, label 3 to English, label 4 to French, label 5 to German, label 6 to Greek, label 7 to Hindi, label 8 to Italian, label 9 to Kannada, label 10 to Malayalam, label 11 to Portuguese, label 12 to Russian, label 13 to Spanish, label 14 to Sweedish, label 15 to Tamil, and label 16 to Turkish. These labels will be useful when discussing the performance of each models.

3.3 Models and Experiments

A plethora of models were implemented. The models we will be focusing on are the following: multinomial naive Bayes, support vector machines, multinomial logistic regression, random forest, Bi-LSTM, BERT-based transformer, basic RNN, Bi-LSTM and Bi-GRU with both dropout and batch normalization, CNN for language identification, langdetect library built within Python, and fastText by Facebook. These models will be divided into three groups: Traditional ML models, DL models, and Pretrained models. Multinomial naive Bayes, support vector machines, multinomial logistic regression, and random forest will be considered the more traditional ML models. Bi-LSTM, BERT-based transformer, basic RNN, Bi-LSTM and Bi-GRU with both dropout and batch normalization, and CNN for language identification will be considered the DL models. Finally, langdetect and fastText will be considered to be the pretrained models.

Models with slight variations from the ones just discussed were also developed. However, we decided to concentrate on the most interesting ones from our point of view. All models that were not discussed here are present in our submitted code with their respective performance metrics.

The experiments we ran on these models were all relatively simple. Essentially, when appropriate, we trained, validated, and tested the models on the Language Detection dataset (Saji, 2021). Hyperparameter tuning was also done when appropriate to get the best performance available from all implemented models. When possible, confusion matrices and classification reports were then generated to compare results between all models. Accuracy measures were used to determine which models performed better than others.

4 Results

In this section, we will be discussing the results of our experiments and their general trends. All

Table 1: Accuracies for a Variety of Traditional Machine Learning Models for Language Identification

Model	Accuracy
Multinomial Naive Bayes	98.20%
Support Vector Machines	98.10%
Multinomial Logistic Regression	**98.49%**
Random Forest Classifier	98.35%

Table 2: Accuracies for a Variety of Deep Learning Models for Language Identification

Model	Accuracy
Bi-LSTM	95.41%
BERT-based transformer	**98.93%**
Basic RNN	94.24%
Bi-LSTM and Bi-GRU	95.82%
CNN for language identification	95.08%

Table 3: Accuracies for a Variety of Pretrained Models for Language Identification

Model	Accuracy
Langdetect	**95.51%**
FastText	89.24%

models are evaluated based on their accuracy, that is, their capability in identifying the language in which a text is written.

4.1 Traditional Machine Learning Models

Multinomial naive Bayes, support vector machines, multinomial logistic regression, and random forest will be analyzed in this section. For MNB's performance, the model has an overall test accuracy of 98.20% while using bi-grams and trigrams. It performs well for virtually all languages with a slight performance drop-off for label 14. The overall test accuracy drops to 90.60% when using TF-IDF. Recall and F1 score values for labels 2 and 3 are not good. For SVMs, overall test accuracy is 98.10% with great performance on all languages. A slight drop-off can be seen for label 13. For MLR, we have an overall test accuracy of 98.49% with minimal drop-offs between languages. For RF, we have an overall accuracy of 98.35%. Precision, recall, and F1-scores are high for most classes, particularly for classes 6, 9, 10, 12, and 15 but there are minor drops in precision and recall for classes like 1, 11, 13, and 14. As we can see from all these results, with regards to the traditional ML models, MLR has the highest performance accuracy.

4.2 Deep Learning Models

Bi-LSTM, BERT-based transformer, basic RNN, Bi-LSTM and Bi-GRU with both dropout and batch normalization, and CNN for language identification will be analyzed in this section. For Bi-LSTM's performance, we have an overall test accuracy of 95.41% but label 16 has a very low precision score (76%) while labels 1, 9 and 11 have a slight drop-off in performance compared to all other labels. For the BERT-based transformer's performance, we have an overall test accuracy of 98.93%. This model performs extremely well on essentially all labels, but label 1 does have the lowest precision value (95%). For the basic RNN's performance, we have an overall accuracy of 94.24%. We have high precision, recall, and F1-scores for most classes, particularly 0, 3, 4, 6, 8, 14 but label 7 and label 12 show some weaknesses, likely due to class imbalances. For the Bi-LSTM and Bi-GRU with dropout and batch normalization model performance, we have an overall test accuracy of 95.82%. We notice that dropout layers significantly reduce overfitting while maintaining strong accuracy and most classes achieve precision and recall scores above 90%. Label 12 and label 16 do exhibit slightly lower F1-scores however (88% and 93% respectively). For the CNN's performance, we get an overall test accuracy of 95.08% where high accuracy is achieved across most labels. Label 9 does have a relatively lower precision (55%). Moreover, some labels like 1 and 16 have slight recall drops, indicating potential misclassifications.

4.3 Pre-trained Libraries

Langdetect and fastText will now be analyzed in this section. For langdetect, we have an overall accuracy of 95.51% while for fastText we get an overall accuracy of 89.24%. Langdetect performs quite well, beating out a variety of other models implemented by our group while fastText performs worse than all the previously discussed models.

4.4 Remarks and Trends

As we can see, the more traditional machine learning models perform quite well, and often times better than more complex deep learning models. This can be due to a variety of factors. As a matter of fact, usually, simpler models are able to generalize quite well with smaller datasets like with the one we are currently using. Deep learning mod-

els, on the other hand, need much more data to learn the patterns and particularities of input values which can explain why they perform worse than the simpler models. This can also explain why the BERT-transformer model performs so well: BERT is a pretrained deep learning model and it can still perform well even in our rather simple training and testing environment. Furthermore, we do notice a trend related to the class imbalances from the dataset we used. Indeed, oftentimes, most models would perform poorly when making positive predictions about classes with low quantities of data. This was expected as the classes are imbalanced. Another surprising remark which was made was the performance of fastText. While langdetect performed quite well with an overall accuracy of 95.51%, fastTest struggled compared to all other models with an overall accuracy of 89.24%. As a group, we thought that the pretrained models would all perform relatively well but fastText was our worst performing model and the only one not getting above 90.00%. Finally, the full classification reports and confusion matrices of all models (when applicable) can be observed in the submitted code for a more in depth view of their performance metrics.

5 Discussion and Conclusion

All in all, the main conclusion that we can pull from our study is that for simpler tasks such as language identification, more complex deep learning models are not always necessary to have exceptional performance. The one exception to that in our paper would be the BERT transformer model which performs extremely well in all situations with all languages. Indeed, it is our best performing model with an accuracy of 98.93%, with multinomial logistic regression coming in a close second place with an overall accuracy of 98.49%. Our initial hypothesis was proven to be false, given that, in general, the more complex models do not perform better than the more simpler ones, at least in our experimental setting.

There are multiple limitations to our approach. Firstly, our dataset wasn't particularly big, which may explain the performance of certain deep learning models that often require a lot of data to perform to the best of their capabilities. Additionally, we tested on 17 different languages when we could have perhaps done testing on more by including multiple datasets together. Furthermore, with more time and computational power, we could have further hypertuned the models to obtain better performance. In the same vein, our study was very broad and not specific. We did not do any deep dives into any particular models, but rather did a more surface level analysis of our topic of study.

For future work, it would be interesting to extend all these models to different sorts of datasets that contain more data but also more languages. This would allow for a better evaluation of all models and extend their usage to even more languages. Moreover, some models can be chosen for a much deeper analysis when compared to what we did, given that our analysis was meant to be more general.

6 Statement of contributions

This project was a collaborative effort among three members. Sangmin Lee led the implementation and analysis of traditional machine learning models, including Multinomial Naive Bayes, SVM, and Logistic Regression. Mathieu-Joseph Magri focused on deep learning models such as Bi-LSTM, CNNs, and BERT, as well as integrating pre-trained libraries like langdetect and fastText. Minjae Kim contributed to the implementation of hybrid models, wrote about related work, and assisted in writing and refining sections on deep learning models. All members worked together on data preprocessing, experimental design, and hyperparameter tuning, ensuring consistent evaluations. The introduction, discussion, and conclusion were co-authored by the team, with all members involved in editing and finalizing the manuscript.

7 Ethics Statement

This research was conducted in compliance with ethical standards. The dataset used is publicly available and does not contain sensitive or private information. The project avoids discriminatory or harmful outcomes by ensuring fair evaluation across languages and focusing solely on the technical aspects of model performance. Our code and findings are shared transparently to promote reproducibility and further research in the field.

8 Acknowledgements

All of us would like to thank the entire staff for the wonderful semester! COMP 550 was a very fun class to take!

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2022. fasttext.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Basil Saji. 2021. Language detection: A dataset for language detection.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification.