Evaluating Linguistically Motivated Features in Language Identification of Short Text

Sam Briggs
Department of Linguistics
University of Washington
Seattle, WA 98195
briggs3@uw.edu

Lexie Wang
Department of Linguistics
University of Washington
Seattle, WA 98195
lexwang@uw.edu

Abstract

The paper explores the task of language identification (LangID), which involves determining the natural language(s) in which a given piece of text is written. The task is crucial for natural language processing (NLP) tasks that rely on prior knowledge of the natural language domain, such as part-of-speech tagging and named-entity recognition. We used 12 linguistically motivated features (which can be partitioned into three groups: the top k most frequent characters, tokens, and character bigrams in the given text; frequency indicators over the whole data set; and orthographic features) and implemented three different methods (Multinomial Naive Bayes, Decision Tree, and Logistic Regression) to evaluate their performance on classifying the Big Language Detection Dataset, which contains over 10 million instances written in 404 languages. Results reveal that the decision tree classifier performs best, given the 12 linguistic features.

1 Introduction

Language Identification (LangID) is a Natural Language Processing (NLP) task that involves determining the natural language(s) that a given piece of text is written in.

This task is especially important because many NLP tasks presuppose the natural language domain of the given data. NLP tasks from trivial tasks such as tokenization, lemmatization, and stopword removal to harder tasks, such as part-of-speech (POS) tagging, dependency parsing, constituency parsing, and named-entity-recognition (NER) all rely on prior knowledge of the natural language domain. This task is also important with the proliferation of different languages represented on the internet, where automatic language identification is necessary before any machine translation.

Current methods of Language Identification include shallow methods (TODO: include citations), as well as deep methods (TODO: include citations). In this paper, we use linguistically motivated features to explore feature engineering. We also implement three different methods: Multinomial Naive Bayes (sec. 3.3.2), Decision Tree (sec. 3.3.3), and Logistic Regression (sec. 3.3.4). Finally, we evaluate the performance of the linguistically motivated features over these four selected methods.

2 Dataset

We used the Big Language Detection Dataset found on Kaggle[1]. This data set has just over 10 million different instances, where each instance is a sentence written in a given language mapped to the ISO 639-3 code of the given language. There is 404 different languages represented within the data set.

[1]It is important to note, that although this data set is massive, there are not the same number of instances per label, so the data set is not uniformly distributed.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

To perform train, dev, test split, we randomly shuffled the data, and then partitioned the data set into 60% train, 20% dev, and 20% test data using `sklearn`[2]. This amounts to about 6 million train instances, 2 million development instances, and 2 million test instances.

3 Method

3.1 Preprocessing

We did not do any text preprocessing. Our intuition is that text preprocessing often will result in a loss of information – information that could be useful for language identification. Also most text preprocessing steps, such as tokenization, removing stop words, removing punctuation, etc., would need to know which language the text is written in beforehand – the very task we are trying to perform. We also believe that orthography differences between languages is important to keep when trying to determine the language of a given piece of text.

3.2 Features

To develop our features, we used our linguistic intuitions. Our features can roughly be partitioned into the following three groups:

3.2.1 Group 1: Top k over Text

1) k_1 Most frequent characters in the given text
2) k_2 Most frequent tokens in the given text
3) k_3 Most frequent character bigrams in the given text

Group 1 tries to capture the intuition that the most frequent tokens[3], characters, and character combinations are informative during the language identification task. Intuitively, in a language such as English which delineates words by whitespace, the most frequent tokens (e.g. stop words) will greatly help in identifying the given text. This intuition falls from the fact that the distribution of a language's vocabulary follows Zipf's law (Řehůřek and Kolkus, 2009). For example, even a person who does not speak English might recognize that the sentence *The leopard eats food* is written in English, because of the presence of common words like the determiner *the*.

3.2.2 Group 2: Frequency Indicators over Data Set

Over the whole data set, we created four sets. These four sets contained the n_1 most frequent characters, least n_1 frequent characters, n_2 the most frequent tokens and the n_2 least frequent tokens[4]. The following features are indicators of whether the given text contains a frequent or infrequent character/token.

4) The given text contains a token that appears very frequently within the whole data set.
5) The given text contains a character that appears very frequently within the whole data set.
6) The given text contains a token that appears very infrequently within the whole data set.
7) The given text contains a character that appears very infrequently within the whole data set.

Because target labels in our data set are not evenly distributed, we hoped to capture some information about the given text using frequencies of tokens and characters over the whole training data. Intuitively, the characters and tokens that appear in the text of languages with most frequent instances in the training data would be common, while characters and tokens that appear in the text of languages with the least frequent instances in the training data would be rare. We hoped that these intuitions

[2] `sklearn.model_selection.train_test_split`

[3] Throughout the paper, we will refer to a token as a string delineated by whitespace. A token is the same as an element in the list returned by `string.split()` in python. This means that for languages which do not delineate words by whitespace like Mandarin Chinese, the given text constitutes one token.

[4] We set n_1 and n_2 to be 30, and did not have enough time to test other hyper-parameters. More hyper-parameter tuning is necessary in the future.

would help identify whether a given text was a language that appeared often in the training data, or a language that appeared only a few times in the training data. For example, we hoped that these features would help distinguish English (the most frequent language in the training data) from Central Bikol (ISO 693-3 bcl) which only had one instance in the development data. This group also tries to improve and expand upon the features from group (3.2.1).

These features are also an attempt to patch an issue with using the features from group (3.2.1) for short text. Because each instance of text is so short, often each token in the given text was unique. Therefore grabbing the k_2 most frequent tokens in the given text for feature (2)), often just gave us a random k_2 tokens from the text. This group of features tries to combat this issue by encoding different information about the frequency of the tokens in the given text.

3.2.3 Group 3: Orthography Differences

Our intuition behind the design of group 3 features is that the orthography differences is helpful for capturing the differences between languages, and, hence, will serve as useful features in language identification tasks. For example, languages like English, French, and Italian separate words on whitespace, whereas texts written in Japanese, Chinese, and Korean do not have whitespaces in between strings of characters. Also, since we performed tokenization by splitting on whitespace, texts written in Chinese, Japanese, and so on will just have one long token per instance. Thus, we included the feature (9)) average length of tokens to capture these differences. Additionally, not all languages differentiate between upper and lower case letters. Specifically, all Latin languages have upper and lower case, but many languages like Hindi do not. Similarly, alphanumeric characters are only used my some writing systems, like those of Latin languages. Thus, we believe that the ratio of upper case, lower case, and alphanumeric characters in a given text will capture these differences between distinct writing systems.

8) Ratio of whitespace to all other characters
9) Average length of tokens in the given text
10) Ratio of lower case characters to all characters in the given text
11) Ratio of upper case characters to characters in the given text
12) Ratio of alphanumeric characters to all characters in the given text

3.3 Classification Methods

3.3.1 Baseline

For a baseline, we used majority vote, where we labeled all test data with the most common label in the training data. The most common label in the training data was English (unsurprisingly) and thus we labeled all test data as English.

3.3.2 Multinomial Naive Bayes

We tested our features with the Multinomial Naive Bayes (MNB) classifier, using `sklearn`[5]. MNB is a probabilistic model that assumes conditional independence between features given the class label. Based on that assumption, the model calculates Bayes-optimal estimates of model parameters based on the training data, which are then used to predict test instances (Mccallum and Nigam, 1998). This model has been used to great success in the language identification task before (Lui and Baldwin, 2012). To train the model, we used the partial_fit method and used batches size of about $\frac{1}{6}$ of the training data, which comes out to about 10 million instances[6].

3.3.3 Decision Tree

Another classifier we used to perform our task is the Decision Tree (DT) classifier, which has been used before for the LangID task (Hakkinen and Tian, 2001). We used `sklearn`[7] to build the DT

[5] `sklearn.naive_bayes.MultinomialNB`

[6] We needed to use partial_fit to train the model because the data set is so large that we ran out of memory when training the classifier on our home computers.

[7] `sklearn.tree.DecisionTreeClassifier`

classifier. The DT model is a nonparametric, tree-like model that divides a hypothetical input space into multiple regions through recursive binary splitting at each node. The DT model splits the data set into subsets based on the feature values that reduce entropy. We found that using a maximum depth of 30 created the best results.

3.3.4 Logistic Regression

Furthermore, we also employed logistic regression (LR) using `sklearn`[8]. The logistic regression algorithm has been applied in various fields for classification tasks (Hoang, 2019). We used the training algorithm of Stochastic Gradient Descent (SGD)[9] to train our LR classifier. We used batch sizes of about $\frac{1}{6}$ of the training data, which comes out to about 10 million instances. We found that running 60 batches created the best results.

4 Results

We employed three different hyperparameters, which are k_1, k_2 and k_3 as used in group 1 features (3.2.1). By analyzing the initial results on the development set, we found that the highest accuracy results for every single classification model is achieved with $k_i = 9$, which is the highest k value we tested.

We achieved an accuracy score of 0.15 with our baseline model (sec. 3.3.1). The Multinomial Naive Bayes model performs slightly better than the baseline, achieving an accuracy score of 0.2. The Decision Tree model achieved the best performance among all of the classification models we tested, with an accuracy score of 0.75 when maximum depth is equal to 30. The logistic regression model only achieved an accuracy score of 0.07, which is less than that of the baseline model.

5 Discussion

One challenge we faced was the data set is extremely large, with more than 10 million instances, which leads to the problem of computational resource constraints. For example, we encountered issues with the space and time complexity of the code, and our giant vectors wouldn't fit into the memory of our home computers, so we had to run the models on the Linguistics Department's supercomputer.

As we ran the models on the development set, we found that the highest accuracy scores are achieved with k values[10] equal to 9, which is the highest value we tested. This leads us to believe that the higher the k values, the more informative the feature vectors are, and the better our models are able to perform.

Additionally, many of the features we designed are correlated. For example, the features (10) ratio of lower case characters to all characters and (11) ratio of upper case characters to all characters are highly correlated, since these two features both aim at detecting the presence of casing in a given piece of text and tend to vary together. Feature correlations as such limit the performance of our models, as these correlations could mask the interactions between different features. It would be ideal to remove correlated features and use distinct features.

Moreover, our features don't seem to perform well with probabilistic models like Multinomial Naive Bayes (MNB). Since MNB is a probabilistic classifier, it is usually used with word count features (e.g. count vectorization). The combination of bag of words features and the MNB is commonly used for text classification tasks. However, the features we designed are not probabilistic features. Thus, it is reasonable that MNB, when used in combination with our linguistic features, achieved a poor accuracy score. To achieve higher accuracy scores with probabilistic models, it would be best to use count-based features like bag of words. In addition, our Logistic Regression (LR)also performed poorly. It is also reasonable that LR performed poorly given our linguistic features, since the feature vectors were not designed to be linearly separable.

[8] sklearn.linear_model.SGDClassifier

[9] We needed to use the SGD version of logistic regression because the data set is so large that we ran out of memory when training the classifier on our home computers.

[10] k values as corresponding to features (1) through (3)

The Decision Tree model performs the best, amongst all classifiers we tested our features on. This result is intuitive because the intuition and logic of the DT model are in line with the intuition behind our design of the linguistic features. Our features are designed based on the intuition of dividing the dataset into subgroups based on feature values. These features capture the intra-group similarities and inter-group differences between subgroups of data. Thus, it is intuitive that the DT model will achieve the best performance given our linguistic features.

In the future, with the goal of improving the accuracy of our classifiers in mind, we need to design better features that are distinct from another (in contrast to being correlated). We could also test the performance of our linguistic features with a neural network in an attempt to optimize our features, which could yield better classification results.

A Appendix

A.1 Results Table

Model	k_1	k_2	k_3	Accuracy
Baseline	N/A	N/A	N/A	0.15
Multinomial Naive Bayes	9	9	9	0.2
Decision Tree	9	9	9	0.75
Logistic Regression	9	9	9	0.07

Table 1: Accuracy of our Best Models on Test Data

Here we have the results table. k_1, k_2, and k_3 are hyper parameters that correspond to features (1)), (2)), and (3)) respectively. The *Model* column corresponds to the three different classification methods as seen in sections 3.3.2, 3.3.3, and 3.3.4 as well as the baseline seen in section 3.3.1. The *Accuracy* column is the raw accuracy for the predictions made by each model for test data.

A.2 Source Code

Our source code can be found on GitHub.

References

Hakkinen, J. and Tian, J. (2001). n-gram and decision tree based language identification for written words. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU '01.*, pages 335–338.

Hoang, N.-D. (2019). Automatic detection of asphalt pavement raveling using image texture based feature extraction and stochastic gradient descent logistic regression. *Automation in Construction*, 105:102843.

Lui, M. and Baldwin, T. (2012). langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30.

Mccallum, A. and Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, Madison, USA.

Řehůřek, R. and Kolkus, M. (2009). Language identification on the web: Extending the dictionary method. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, pages 357–368, Berlin, Heidelberg. Springer Berlin Heidelberg.