Language Classification

Team name: Cereal Killer

Paul Wu

#26XX

jiapeng.wu@mail.mcgill.ca

Andre Kaba

#260493293

andree.kaba@mail.mcgill.ca

Jingyun Liu

#260657058

jingyun.liu@mail.mcgill.ca

1. **INTRODUCTION**

We present a classification model to discriminate between 5 different languages. Short sentences or snapchats of conversations as samples from each language are given as training set, and we experimented with both linear and nonlinear classifiers (and ensemble methods?), as well as different approaches of preprocessing and feature extraction of the raw text input, including manual removal of special characters and unigram lexical features. Results show that XYZ classifier achieves the highest accuracy of XX% and an F1 score of XX on validation set, with appropriate preprocessing.

1. **RELATED WORK**

One of the main focuses of language identification task is discriminating similar languages. Zampieri et al. (2014) (<http://www.aclweb.org/anthology/W14-5307>) concluded and compared various methods, where the most commonly used features are n-gram character or words features. More specifically, there have been approaches using bag-of-words for language identification (2013)(<http://ieeexplore.ieee.org/document/6705230/> ) and n-gram language models combined with Part-Of-Speech distribution(2013) (<http://pers-www.wlv.ac.uk/~u22984/papers/taln2013.pdf>).

Another focus of language identification task is applied to social media data such as Twitter. (<http://www.aclweb.org/anthology/W14-1303>) Most commonly used feature extraction methods includes ad-hoc rank-order statistic over character n-grams proposed by Scheelen et al. (2003) (Frank Scheelen, 2003. libtextcat. Software available at http://software.wise-guys.nl/libtextcat/.). Lui et al. (<http://www.aclweb.org/anthology/Q14-1003>) presented a multi-lingual detection model where features are constructed by character n-grams and selected using Information Gain(IG), reaching an F1 score of 0.9

In this project, however, we are given limited data regarding the text length and language variety. Thus we shall only experiment with the most common feature extraction methods mentioned above: n-gram language model over the characters.

1. **PROBLEM REPRESENTATION**
2. *Data Preprocess*

We observe that in the both training and testing data, there exists a mixture of corrupted and outliner characters that obviously do not belong to the language. Three approaches were compared: 1) remove outside of vocabulary for each language separately, in reference to the alphabet dictionaries in the language; 2) only remove emojis and urls while leave the others unchanged, since emoji and url are universal and clearly should not belong to any particular language; 3) do nothing other than converting everything into lower case, since the characters given in test case are all in lower cases. We concluded that 3), doing minimal removal of the corrupted characters results in the best performance, thus is the final choice for predicting on test data.

1. *Feature Extraction.*

Previous works have shown that it n-gram language model with tfidf weighting is one of the most effective way of encoding inputs. We chose character instead of word representations since word representations would have a much higher dimension caused by word diversity across different languages, whereas character contains more compact information. Observing that the test data were single characters drawn from different language distributions instead of complete words or sentences, it makes more sense to encode the input at a character level using unigrams.

We observe that there are larger variety of characters in training set than test set. Therefore, we experimented on restricting the vocabulary to be the set of characters that appear in both training and testing set, in order to eliminate irrelevant and unnecessary noises for inference. In theory this operation reduces dimensionality of the feature space, and thus should improve the performance on some models such as decision tree, since the tree would have fewer split boundaries to consider and would need fewer splits. However, in practice we observed no significant improvement over the test set.

1. **ALGORITHMS**

Training data is split into 10 folds for cross-validation. We collect accuracy and F1 score by each classifier for hyperparameter decisions. We experimented with Principal Components Analysis (PCA) for reducing the dimensionality of the data, but since the input dimensionality is already rather small, the performances show no significant improvement after the operation

We implemented the following classifiers:

1. *Linear classifiers*

**Naive Bayes**: Naive Bayes models have been widely used for clustering and classification. We assume that each feature in our data input follows a multinomial distribution, since each entry is a discreet real number representing the occurrence of one character. It is unreasonable to assume distributions such as Gaussian or Bernoulli, since the feature entries represents occurrence, and are not continuous nor binary.

The model estimate for each feature using Maximum Likelihood Estimator (MLE), and discriminates between classes by calculating for each label y. Because MLE is prone to overfit the observed distribution, we also experimented with different amount of smoothing, since Maximum a Posteriori (MAP) estimator results in a more generalized classifier.

Direct multiplication of fractions over times is numerically unstable and can cause underflow issues. In our implementation, we take natural log over all probabilities and do addition when calculating joint probability.

**Logistic Regression**:

1. *Nonlinear classifiers*

**KNN**: In vanilla KNN, inference time grows linearly with the number of training data point and dimension of features. Since the input matrix is sparse and lots of entries are 0, computation power were wasted on dot product operation. Thus we incorporated an Invert Index Based method (<https://arxiv.org/pdf/1011.2807.pdf>). The idea is to only do calculation on features where the point has a valid value and skip all the 0’s.  However due to high dimensionality of the input, this method still does not scale up. Thus although theoretically doable, we are unable to evaluate KNN efficiently with other models.

**Decision Tree**:

**Random Forest**:

1. **TESTING**

We conducted some basic analysis on the performance of different classifier with various choices of preprocessing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Naive Bayes (laplace smoothing) | Logistic Regr (L2 reg) | Decision Tree | Random Forest |
| No preprocess, no tfidf | 0.855 | 0.859 | 0.855 | 0.911 |
| No preprocess, with tfidf | 0.717 | 0.877 | 0.85 | 0.912 |
| Remove emoji/url, with tfidfl |  |  |  |  |
| Remove all special | 0.818 |  |  |  |
| Vocab intersection |  |  |  |  |

Table X: Comparison across models with different preprocessing

1. **DISCUSSION**

Unigram language model in general lacks robustness because it takes no consideration of any information given around target character, but it makes sense here since test set is composed of single characters drawn independently across language distributions. If the test data given are snapchats like phrases or sentences, we will have to consider more sophisticated language models for feature extraction, and maybe even classifiers that model the interaction between features.

1. **STATEMENT OF CONTRIBUTION**

Jingyun Liu:

* Implemented Naïve Bayes, KNN classifier
* Drafted most parts of the report

Jiapeng Wu:

* Data preprocessing and feature extraction.
* Implemented Logistic Regression
* Construct the structure and write the report

Andre Kaba:

* Implemented Random Forest, Decision Tree

We hereby state that all the work presented in this report is that of the authors.

**REFERENCES**