Name: Abhishek Kumar NetId: ak6210
Name: Raj Kamal NetId: rk3098

Project Report

Language Detection using n-grams

With increasing internet presence of people all over the world, detecting language of text content has emerged as a crucial problem. In this project, we have attempted to create a model for learning languages from the text. The work is inspired from an article on language detection: https://bugra.github.io/work/notes/2014-12-26/language-detector-via-scikit-learn/

We have used n-grams method for text classification with value of n from 1 to 5. An n-gram is a contiguous series of n items from the text. It is used in various applications such as natural language processing, speech recognition and computational biology. In this case, strings of different lengths are selected randomly from the data to form n-grams. The frequencies of their occurrence are calculated while training the model. Every language has some sequences of characters for prefixes and suffixes which are quite commonly used. The strings in n-grams look for such sequences and that helps in classifying which language particular text belongs to. The n-grams use Markov model for approximation.

Our model can detect 7 languages: English, Chinese, Italian, Portuguese, Dutch, Spanish, French

The data has been taken as text files from different Wikipedia articles in all these languages. The method *getTextData* in class DetectLanguage accepts type of data (train/test) as parameter and returns the text data in form of list.

```
def getTextData(self, type):
    directory = 'data/' + type
    X = []
    y = []
    files = [lang for lang in os.listdir(directory)]
    self.languages = [lang.split('.')[0] for lang in files]
    for nf, file_name in enumerate(files):
        file_path = os.path.join(directory, file_name)
        with open(file_path, encoding="utf8") as text_file:
            text = text_file.readlines()
            y += [nf] * len(text)
            X += text
    return X, y
```

To train the data, feature extraction module from sklearn is used. We create a vector of different n-grams and then train the model using pipeline with logistic regression as our linear model.

```
vectorizer = feature_extraction.text.TfidfVectorizer(ngram_range=(1, 6), analyzer='char')
lang_model = pipeline.Pipeline([('vectorizer', vectorizer), ('clf',
linear_model.LogisticRegression())])
lang_model.fit(Xtr, ytr)
```

Name: Abhishek Kumar NetId: ak6210
Name: Raj Kamal NetId: rk3098

Then we use the trained model for predicting language. The test data has single line text files of various languages. After prediction, the different metrics are measured.

```
yhat = lang_model.predict(Xts)
confusion_matrix = metrics.confusion_matrix(yts, yhat)
```

Metrics for model:

	precision	recall	f1-score	support
ch	1.00	1.00	1.00	1
du	1.00	0.67	0.80	3
en	0.50	1.00	0.67	1
es	1.00	1.00	1.00	1
fr	1.00	1.00	1.00	2
it	1.00	1.00	1.00	1
pr	1.00	1.00	1.00	1
avg / total	0.95	0.90	0.91	10

In future, this model can be further developed to be integrated with a web application as a service and can be used to make a language translator.