Training of Document Categorizer using Naive Bayes Algorithm in OpenNLP

Training of Document Categorizer using Naive Bayes Algorithm in OpenNLP

In this Apache OpenNLP Tutorial, we shall learn how to build a model for document classification with the Training of Document Categorizer using Naive Bayes Algorithm in OpenNLP.

Document Categorizing or Classification is requirement based task. Hence there is no pre-built models for this problem of natural language processing in Apache openNLP.

In this tutorial, we shall train the Document Categorizer to classify two categories : Thriller, Romantic. The categories chosen are movie generes. The data for each document is the plot of the movie.

Following are the steps to train Document Categorizer that uses Naive Bayes Algorithm for creating a Model :

- **Step 1** : Prepare the training data. The training data file should contain an example for each observation or document with the format : Category followed by data of document, seperated by space. For example, consider the below line which is from the training file :

```
Thriller John Hannibal Smith Liam Neeson is held captive in Mexico
```

  Here ,Category is “Thriller”Data of the document is “John Hannibal Smith Liam Neeson is held captive in Mexico”.

  Find the complete training file used in the example, here en-movie-category.

- **Step 2** : Read the training data file.

```
InputStreamFactory
```

```
InputStreamFactory dataIn = new MarkableFileInputStreamFactory(new File("train"+File.separator+"en-movie-category.train"));
ObjectStream lineStream = new PlainTextByLineStream(dataIn, "UTF-8");
ObjectStream sampleStream = new DocumentSampleStream(lineStream);
```

- **Step 3** : Define the training parameters.

```
TrainingParameters
```

```
TrainingParameters params = new TrainingParameters();
params.put(TrainingParameters.ITERATIONS_PARAM, 10+"");
params.put(TrainingParameters.CUTOFF_PARAM, 0+"");
params.put(AbstractTrainer.ALGORITHM_PARAM, NaiveBayesTrainer.NAIVE_BAYES_VALUE);
```

- **Step 4** : Train and create a model from the training data and defined training parameters.

```
DoccatModel model =
```

```
DoccatModel model = DocumentCategorizerME.train("en", sampleStream, params, new DoccatFactory());
```

- **Step 5** : Save the newly trained model to a local file, which can be used later for predicting movie genere.

```
BufferedOutputStream
```

```
BufferedOutputStream modelOut = new BufferedOutputStream(new FileOutputStream("model"+File.separator+"en-movie-classifier-naive-bayes.bin"));
model.serialize(modelOut);
```

- **Step 6** : Test the model for a sample string and print the probabilities for the string to belong to different categories. The method DocumentCategorizer.categorize(String[] wordsOfDoc) takes words of a document as an argument in the form of an array of Strings.

```
DocumentCategorizer
```

```
DocumentCategorizer doccat = new DocumentCategorizerME(model);
double[] aProbs = doccat.categorize("Afterwards Stuart and Charlie notice Kate in the photos Stuart took at Leopolds ball and realise that her destiny must be to go back and be with Leopold That night while Kate is accepting her promotion at a company banquet he and Charlie race to meet her and show her the pictures Kate initially rejects their overtures and goes on to give her acceptance speech but it is there that she sees Stuarts picture and realises that she truly wants to be with Leopold".replaceAll("[^A-Za-z]", " ").split(" "));
```

The complete program is provided in the following java file:

DocClassificationNaiveBayesTrainer.java

```
import
```

```
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
import opennlp.tools.doccat.DoccatFactory;
import opennlp.tools.doccat.DoccatModel;
import opennlp.tools.doccat.DocumentCategorizer;
import opennlp.tools.doccat.DocumentCategorizerME;
import opennlp.tools.doccat.DocumentSample;
import opennlp.tools.doccat.DocumentSampleStream;
import opennlp.tools.ml.AbstractTrainer;
import opennlp.tools.ml.naivebayes.NaiveBayesTrainer;
import opennlp.tools.util.InputStreamFactory;
import opennlp.tools.util.MarkableFileInputStreamFactory;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
* oepnnlp version 1.7.2
* Training of Document Categorizer using Naive Bayes Algorithm in OpenNLP for Document Classification
* @author www.tutorialkart.com
*/
public class DocClassificationNaiveBayesTrainer {

    public static void main(String[] args) {

        try {
            // read the training data
            InputStreamFactory dataIn = new MarkableFileInputStreamFactory(new File("train"+File.separator+"en-movie-
category.train"));
            ObjectStream lineStream = new PlainTextByLineStream(dataIn, "UTF-8");
            ObjectStream sampleStream = new DocumentSampleStream(lineStream);

            // define the training parameters
            TrainingParameters params = new TrainingParameters();
            params.put(TrainingParameters.ITERATIONS_PARAM, 10+"");
            params.put(TrainingParameters.CUTOFF_PARAM, 0+"");
            params.put(AbstractTrainer.ALGORITHM_PARAM, NaiveBayesTrainer.NAIVE_BAYES_VALUE);

            // create a model from traning data
            DoccatModel model = DocumentCategorizerME.train("en", sampleStream, params, new DoccatFactory());
            System.out.println("\nModel is successfully trained.");

            // save the model to local
            BufferedOutputStream modelOut = new BufferedOutputStream(new FileOutputStream("model"+File.separator+"en-
movie-classifier-naive-bayes.bin"));
            model.serialize(modelOut);
            System.out.println("\nTrained Model is saved locally at : "+"model"+File.separator+"en-movie-classifier-naive-
bayes.bin");

            // test the model file by subjecting it to prediction
            DocumentCategorizer doccat = new DocumentCategorizerME(model);
```

```
            DocumentCategorizer doccat = new DocumentCategorizerME(model);

            String[] docWords = "Afterwards Stuart and Charlie notice Kate in the photos Stuart took at Leopolds ball and realise that her destiny must be to go back and be with Leopold That night while Kate is accepting her promotion at a company banquet he and Charlie race to meet her and show her the pictures Kate initially rejects their overtures and goes on to give her acceptance speech but it is there that she sees Stuarts picture and realises that she truly wants to be with Leopold".replaceAll("[^A-Za-z]", " ").split(" ");
            double[] aProbs = doccat.categorize(docWords);

            // print the probabilities of the categories
            System.out.println("\n--------------------------------\nCategory : Probability\n--------------------------------");
            for(int i=0;i<doccat.getNumberOfCategories();i++){
                System.out.println(doccat.getCategory(i)+" : "+aProbs[i]);
            }
            System.out.println("--------------------------------");

            System.out.println("\n"+doccat.getBestCategory(aProbs)+" : is the predicted category for the given sentence.");
        }
        catch (IOException e) {
            System.out.println("An exception in reading the training file. Please check.");
            e.printStackTrace();
        }
    }
}
```

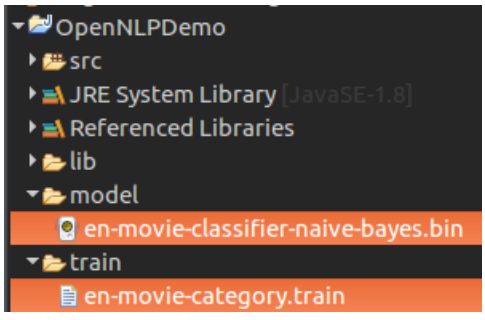
When the above program is run, the output to the console is as shown below :

Program Output

```
Indexing events using cutoff of 0
```

```
Indexing events using cutoff of 0

    Computing event counts... done. 66 events
    Indexing... done.
Collecting events... Done indexing.
Incorporating indexed data for training...
done.
    Number of Event Tokens: 66
        Number of Outcomes: 2
      Number of Predicates: 6886
Computing model parameters...
Stats: (27/66) 0.4090909090909091
...done.

Model is successfully trained.
Compressed 6886 parameters to 6886
3 outcome patterns

Trained Model is saved locally at : model/en-movie-classifier-naive-bayes.bin

---------------------------------
Category : Probability
---------------------------------
Thriller : 2.1694037140217655E-14
Romantic : 0.9999999999999782
---------------------------------

Romantic : is the predicted category for the given sentence.
```

The location of the training file and the locally saved model file are shown in the following picture :



Location of Training file and Generated Model file

Conclusion :

In this OpenNLP Tutorial, we have learnt briefly the training input requirements for Document Categorizer API of OpenNLP and also learnt the example program for Training of Document Categorizer using Naive Bayes Algorithm in OpenNLP used for document classification.

Learn OpenNLP

- OpenNLP Tutorial
- Setup Java Project with OpenNLP in Eclipse
- OpenNLP Models

Detection / Extraction using Java API

- Tokenizer Example
- Sentence Detection Example
- Parts-Of-Speech Tagger Example
- Chunker Example
- Lemmatizer Example
- Named Entity Extraction Example

Training using Java API

- Sentence Detection Model Training
- Name Entity Finder Model Training
- Document Categorizer Training - Maximum Entropy
- Document Categorizer Training - Naive Bayes
- Document Categorizer with N-gram features used
- Language Detector Training Example

Command Line Tools

- Setup and start using Command Line Tools

Useful Resources

- How to Learn Programming