Using Named Entity Recognition technique to classify News Articles

Ankit Saxena, Shailendra Patil

Abstract
This project deals with classifying the news articles. In this project we take the ”The Reuters-21578, Distribution 1.0” data. There are total 135 classes into which the news has been classified. The training data has already been classified, our task is to classify the articles from test data into its corresponding classes. We have applied three different methods to train the data. First we implemented the model by training it using Support Vector Machine (SVM), but before applying the SVM we normalized and tokenized the data in all articles and then trained the model using SVM. In the second approach, we used Named entity recognition. In this part we first did the Part of Speech (POS) tagging of words and based on POS tags we did the Named Entity Extraction. In the third approach, we have used the Noun Phrase chunks to normalize the text. Using the tokenized data from the text we have trained our model using SVM. Finally we compare the results of the three methods described above.

Keywords
Classification — Support Vector Machines — Part of Speech Tagging — Chunking — Named Entity Recognition

Contents

Introduction

As the number of news articles and texts are increasing significantly on a daily basis, it is becoming a great challenge for newspapers to categorize or determine the interest of their users. Given the fact that most of the news these days is consumed electronically, we need an algorithm to classify the news automatically. In this project, we have classified the news article using Named Entity Recognition, Noun Phrase chunking and a traditional data normalization technique. Named-entity recognition (NER, also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc [1]. **The Reuters-21578, Distribution 1.0** data is used for our analysis. Using the various techniques for text normalization and tokenization, we have generated multi-label classification models and used them to predict the labels of the articles in the test dataset.

1. Theory

1.1 Tokenization

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining [2].
Example : ”Lets make America great again. The White House is in United States. My friend lives in a white house. A White House is beautiful.”
We will use the same example for future references.
This is how Tokenization applied on the above sentence looks like :

```
['Lets', 'make', 'America', 'great', 'again', '.', 'The', 'White',
'House', 'is', 'in', 'United', 'States', '.', 'My', 'friend', 'lives',
'in', 'a', 'white', 'house', '.', 'A', 'White', 'House', 'is',
'beautiful', '.']
```

Figure 1. Tokenization

Typically, tokenization occurs at the word level. The tokenized words can be used for further analysis, like one can do POS tagging on the tokenized words, and can also perform chunking, named entity recognition etc.
We have used the **word_tokenize** from **NLTK** for Tokenization

1.2 POS Tagging

In corpus linguistics, **part-of-speech tagging** (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph [3].

For the same example considered above, once Tokenization is done, we applied POS tagging on the tokenized words. Below is the results obtained for the same:

```
[('Lets', 'NNS'), ('make', 'VBP'), ('America', 'NNP'), ('great', 'JJ'), ('again', 'RB'),
(',', ','), ('The', 'DT'), ('White', 'NNP'), ('House', 'NNP'), ('is', 'VBZ'), ('in',
'IN'), ('United', 'NNP'), ('States', 'NNPS'), ('.', '.'), ('My', 'PRP$'), ('friend',
'NN'), ('lives', 'VBZ'), ('in', 'IN'), ('a', 'DT'), ('white', 'JJ'), ('house', 'NN'),
('.', '.'), ('A', 'DT'), ('White', 'NNP'), ('House', 'NNP'), ('is', 'VBZ'), ('beautiful',
'JJ'), ('.', '.')]
```

Figure 2. POS Tags

pos_tag from nltk was used for this process. POS tagged words can be further used to obtain chunks, a process known as **Chunking**.

1.3 Chunking

Chunking is also called shallow parsing and it's basically the identification of parts of speech and short phrases (like noun phrases). Part of speech tagging tells you whether words are nouns, verbs, adjectives, etc, but it doesn't give you any clue about the structure of the sentence or phrases in the sentence. Using chunking technique, we can get the Noun Phrase Chunks(NP), Verb Phrase Chunks(VP) etc. Chunking find its application in Named Entity Recognition [4].

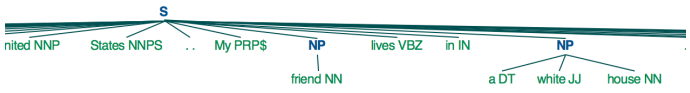

Figure 3. NP Chunks

In the example mentioned earlier, a white house is one among the NP phrases. We have used regular expression on POS tagged data to get the NP phrases.

1.4 Named Entity Recognition

Named-entity recognition (NER, also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc [1]. **ne_chunks** from **NLTK** was used for this purpose. The various named entity tags are shown below [5]:

NE Type	Examples
ORGANIZATION	*Georgia-Pacific Corp., WHO*
PERSON	*Eddy Bonte, President Obama*
LOCATION	*Murray River, Mount Everest*
DATE	*June, 2008-06-29*
TIME	*two fifty a m, 1:30 p.m.*
MONEY	*175 million Canadian Dollars, GBP 10.40*
PERCENT	*twenty pct, 18.75 %*
FACILITY	*Washington Monument, Stonehenge*
GPE	*South East Asia, Midlothian*

Figure 4. NER Tags

For the example considered, below are few Named entity recognized in the text.

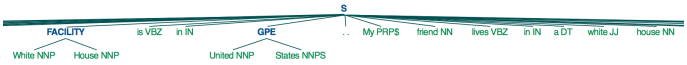

Figure 5. NER Tree

Here **White House** is tagged as **FACILITY**. But in the sentence we have "a white house", here it is not treated as **FACILITY**. It can be seen that the Named Entity recognition depends on the context as well.

1.5 Support Vector Machines

In our project we are using **Support Vector Machine(SVM)** for training and making predictions. In machine learning, SVM's are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier [6].

In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

Below is typical example for two classes classification and the number of splits it can have. It can be seen that there are many possible hyperplanes that correctly classifies the data. The goal of SVM is to find the optimal hyperplane among all the possible hyperplanes.

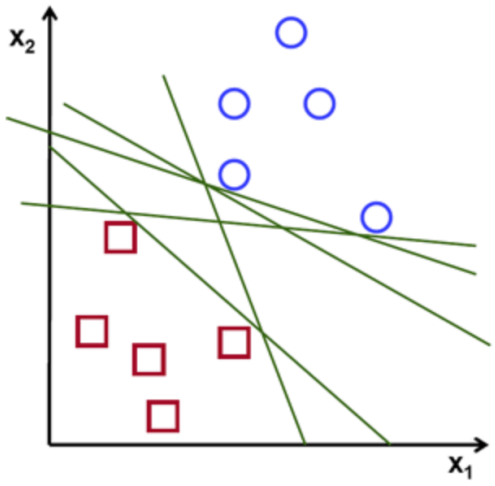

Figure 6. Classification

2. Data

2.1 Data Background

The Reuters-21578, Distribution 1.0 [7] dataset is used for our analysis. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from **Reuters Ltd.** (Sam Dobbins, Mike Topliss, Steve Weinstein) and Carnegie Group, Inc. (Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg) in 1987.

In 1990, the documents were made available by **Reuters** and **CGI** for research purposes to the Information Retrieval Laboratory (W. Bruce Croft, Director) of the Computer and Information Science Department at the University of Massachusetts at Amherst. Formatting of the documents and production of associated data files was done in 1990 by **David D. Lewis** and **Stephen Harding** at the Information Retrieval Laboratory.

Further formatting and data file production was done in 1991 and 1992 by David D. Lewis and Peter Shoemaker at the Center for Information and Language Studies, University of Chicago. This version of the data was made available for anonymous FTP as "Reuters-22173, Distribution 1.0" in January 1993. From 1993 through 1996, Distribution 1.0 was hosted at a succession of FTP sites maintained by the Center for Intelligent Information Retrieval (W. Bruce Croft, Director) of the Computer Science Department at the University of Massachusetts at Amherst.

2.2 Data Description

The dataset consists of **22 data files**, along with **SGML Document Type Definition** file that describes the data file format, and **six** other files describing the categories used to index the data. The documents have a **unique ID** number which is assigned in chronological order to the data files starting from 1, with each data file consisting of **1000** documents ordered by this unique ID.

The data files are in **SGML** format. Each file begins with a document type declaration line. All individual articles are marked with SGML tags, <REUTERS> is the starting tag and < /REUTERS> is the ending tag. The **document id** can be identified by the **NEWID** value within the REUTERS tag. This value is unique for every news article.

For every **document id**, we have one or more topic(s) defined for it, which is/are present within the <TOPICS> tag. There are a total of **135** unique topics into which the documents are classified.

These 135 topics are saved in **"all−topics−strings.lc.txt"** file. Every document used in our training model is labeled with one or more of these 135 values.

The content of the document is found within the <BODY> tag. Every document id has a BODY, which contains the article content. In our analysis, Topics, Doc id and Body are tags that are of prime importance.

2.3 Data Preprocessing

The biggest task in any Text Mining problem is Data Preprocessing. Before we apply any algorithms, we first need to clean the data. The 22 files has the data in SGML format. So we first need to extract the data without any tags. So first we clean the data such that for every document id, we only take the body of the document along with topic associated with it and save it as **csv (comma separated value)** file. The body of the document is converted as bag of words and then saved into the csv file. Figure1 is the representation of how the csv file looks and how it is saved.

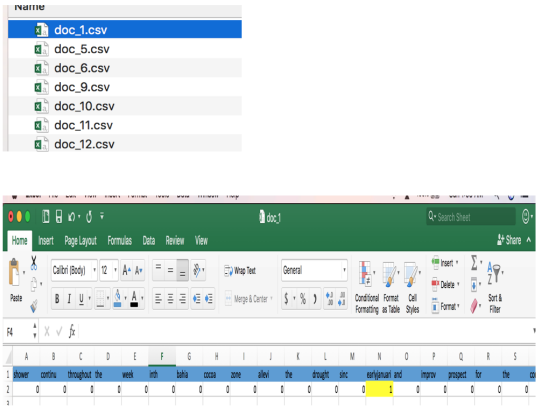

Figure 7. Representation of data after removing tags

As seen, **doc_1.csv** represents the csv file created from document id 1, the file name general syntax is **doc_documentid.csv**. The content of the csv file is as shown. The **blue** highlighted row is the bag of words from that particular document, row **1** in this case. And the second row is a list of **135** values repre-

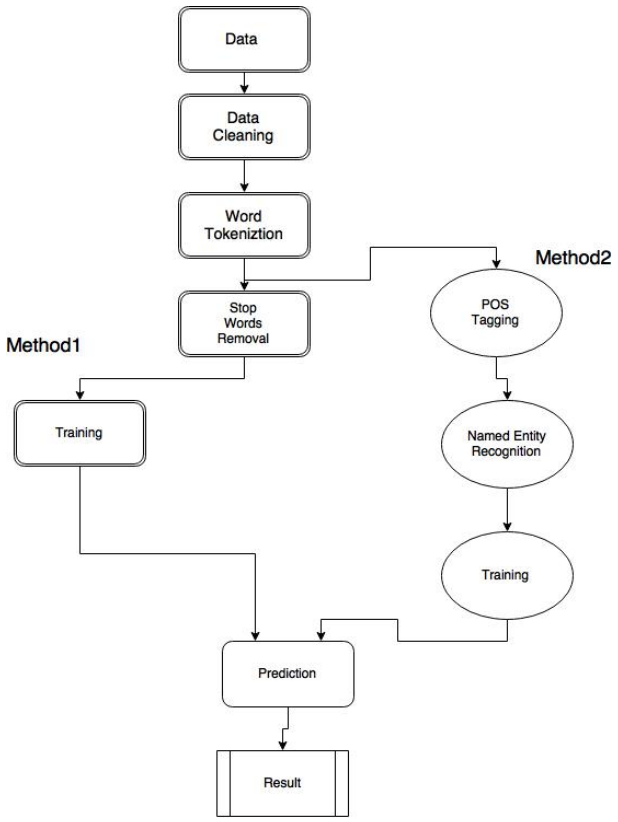

Figure 8. Flow Chart

senting the topics. These 135 values list are in the same order in which topics are saved in **"all−topics−strings.lc.txt"** . The second row values will be either **0** or **1**. If the value of a column is 1, then the corresponding column number should be checked with column number as row number of **"all−topics−strings.lc.txt"** and that particular topic or topics would be the **classification** of the given document. In the figure, **yellow** entry represents 1 and it is in **14th column**. When we check the textbf"all−topics−strings.lc.txt" 14th row, we find it is **cocoa**. So the **document id 1** classification is **cocoa**.

3. Experiment

In this section we discuss the flow of the experiment along with the implementation process. Figure 6 shows a typical **flow chart**. Here we implement our project in two different methods as show in the flow chart:

1. **Method1**: Here we used a traditional approach of preprocessing, like tokenizing the words and removing stop words and applying the learning algorithm.

2. **Method2**: We use NP Chunks and Named Entity Recognition for data preprocessing and then apply the learning algorithm.

The flow chart can be explained as:

1. **Data**: In this problem, we have a total of 135 labels for the newspaper articles. Each article can be labeled with more than one label.

   In order to classify the documents, we need to learn or build a model from the documents that are already labeled. We need to extract meaningful information or words here, from these documents and use their actual labels as a means for the accuracy check.

2. **Data Preprocessing**: Below explanation we consider data cleaning, word tokenization, POS tagging, named entity recognition as a part of Data Preprocessing. We have a dataset with multiple labels under the category 'topics'. We have used One-vs-Rest technique with SVM (Polynomial and RBF kernel) for text classification. First, a vocabulary was made for the words that occur in the documents that are already labeled and this vocabulary has been used as a feature vector for the model.

   The dictionary has been made in three different ways, first with simple preprocessing, second with Named Entity extraction, and the third one with Noun Phrase chunking.

   For the three approaches, we have used **OneVsRestClassifier** from the **sklearn.multiclass** library to classify the documents into multiple labels. The predicted output is a vector of size 135 labels for each document, where label(i) = 1 denotes that the document belong to that i^{th} label of topics.

   The data in the dataset is present inside XML tags with SGML file format. The documents are either categorized with a label under 'topics' or not. We have used only the documents that are categorized under at least one 'topic'. Each SGML file has been parsed and the useful data has been extracted using the XML tags. The 'body' tag data was extracted and data cleaning steps were applied on it. The data cleaning techniques differ for the three methods used for vocabulary building. The vocabulary is generated for the 135 labels under the 'topics' category by taking the union of the vocab list. Each document has been stored in the 'files' directory for further processing.
   File to be executed: $1\_create\_mapping\_*.ipynb$.

   After generating the individual vocabulary files for each label, we create a consolidated vocabulary or the union of all the vocabs. This works as our feature vector. A vector of size equal to the words in the consolidated vocabulary is created and we enter "1" for the word index which occur in the body tag, for each document. The consolidated vocabulary is stored in the file, "*vocab_main.csv*" [8].

File to be executed: 2_*create_vocab.ipynb*.

Using the saved labeled documents we have created a feature matrix with each document as a row in the matrix and the column are the feature vectors corresponding to the consolidated vocabulary. The feature matrix along with expected output is stored in the matrices form on the local drive, as "*X_train.mat*" for the feature matrix and *Y_train.mat* for the expected output. The size of the "*X_train.mat*" is number of documents * size of consolidated vocabulary. The size of the "*Y_train.mat*" is number of documents * number of labels. File to be executed: 3_*train_data.ipynb*.

We have made the consolidated dictionary in three different ways, first with simple preprocessing (traditional approach), second with Named Entity extraction, and the third with Noun Phrase chunking.

(a) **Simple preprocessing** The following techniques have been used to normalize the text in the **body** tag in the given sequence: Convert the text to lowercase, remove all the HTML tags, remove the numbers, process URL, process email address, process dollar sign, remove punctuations. After these steps, we have Tokenized the text by splitting with ' '. The non-alphanumeric characters are removed and all the word stem are extracted for each word, using the **PorterStemmer** from **nltk.stem** package. The stop have not been removed from this list of stem words.

(b) **Named Entity extraction** The following techniques have been used to normalize the text in the **body** tag in the given sequence: Remove all the HTML tags, remove the numbers, process URL, process email address, process dollar sign, remove non-alphanumeric characters. After these steps, we have found the Named Entities, using the **ne_chunk** from nltk package. After this, we have converted all the extracted Named Entities to lowercase.

(c) **Noun Phrase chunking** The following techniques have been used to normalize the text in the **body** tag in the given sequence: Remove all the HTML tags, remove the numbers, process URL, process email address, process dollar sign, remove non-alphanumeric characters. After these steps, we have found the NP chunks, using the **pos_tag** from nltk package and regular expression. After this, we have converted all the extracted NP chunks to lowercase.

Using the saved matrices, we have split the data into train and test with the help of *train_test_split* library from
sklearn.cross_validation. We have 70 % of the data for training purpose and 30% for testing. **OneVsRestClassifier** from the **sklearn.multiclass** library is used to train the model. First, we have used Polynomial kernel to train the model and then used RBF kernel. The value for accuracy, precision and recall are calculated. File to be executed: 4_*SVM_classify.ipynb*.

To create a classifier for a new dataset or unlabeled data, we need to follow the above steps and train the model on the entire training data above instead of just 70%. The test data would be the unlabeled data.

3. **Training and Prediction** Using the saved matrices, we have split the data into train and test with the help of *train_test_split* library from
sklearn.cross_validation. We have 70 % of the data for training purpose and 30% for testing. **OneVsRestClassifier** from the **sklearn.multiclass** library is used to train the model. First, we have used Polynomial kernel to train the model and then used RBF kernel. The value for accuracy, precision and recall are calculated. File to be executed: 4_*SVM_classify.ipynb*.

   To create a classifier for a new dataset or unlabeled data, we need to follow the above steps and train the model on the entire training data above instead of just 70%. The test data would be the unlabeled data. Using the model trained on test data , we predict the classed of unlabeled data.

4. Observations

For the three different consolidated dictionaries, we have observed the Precision and Recall for the two models, and calculated the time taken to train the model and predict the labels. Out of the 135 labels, we have provided the Precision-Recall curve for the first label as an example and the Precision-Recall curve for all the labels combined (for RBF model).

1. **Simple preprocessing**
   Vocabulary Size: 3711 words

   SVM (Polynomial)

Accuracy	Precision	Recall	Time Taken(s)
99.0651	1.0	0.000292	780.1743

   SVM (RBF)
   Time Taken(sec): 2751.6169

2. **Named Entity extraction**
   Vocabulary Size: 590 words

   SVM (Polynomial)

Accuracy	Precision	Recall	Time Taken(s)
99.0651	1.0	0.000292	122.5908

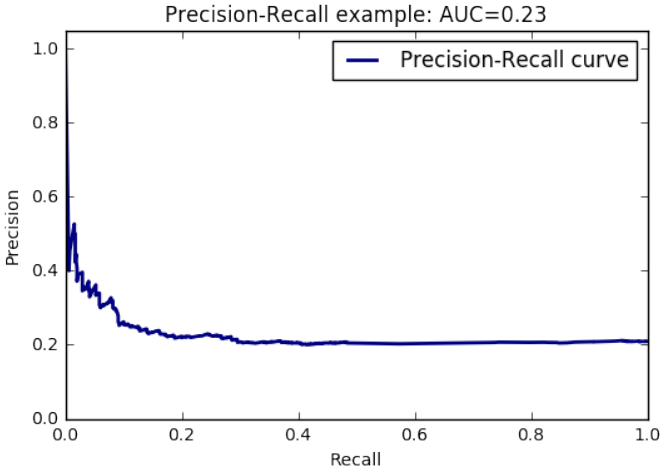

Figure 9. PR curve for Simple preprocessing (Label 1)

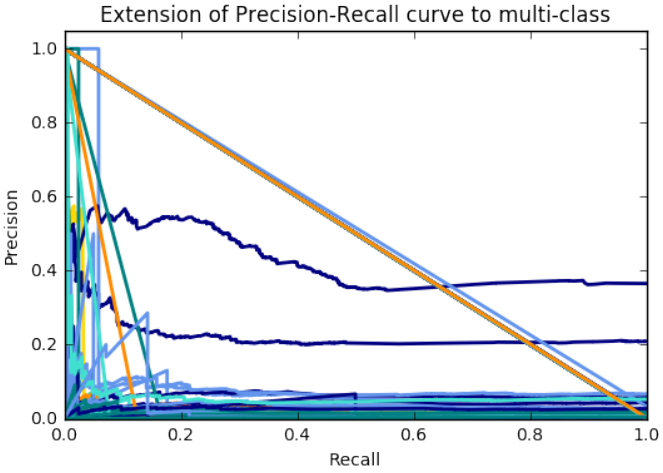

Figure 10. PR curve for Simple preprocessing (all labels)

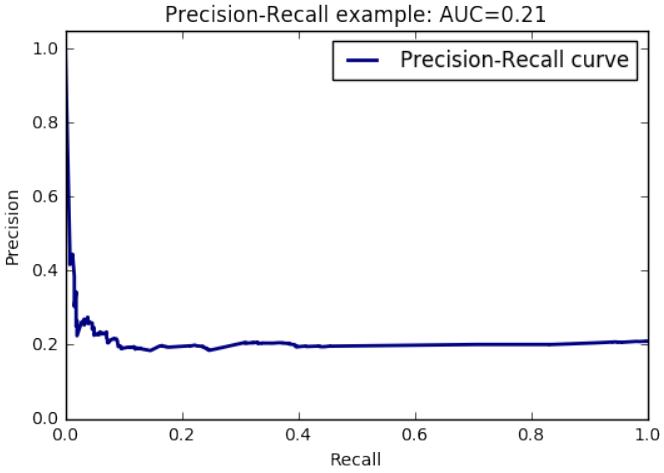

Figure 11. PR curve for Named Entity extraction (Label 1)

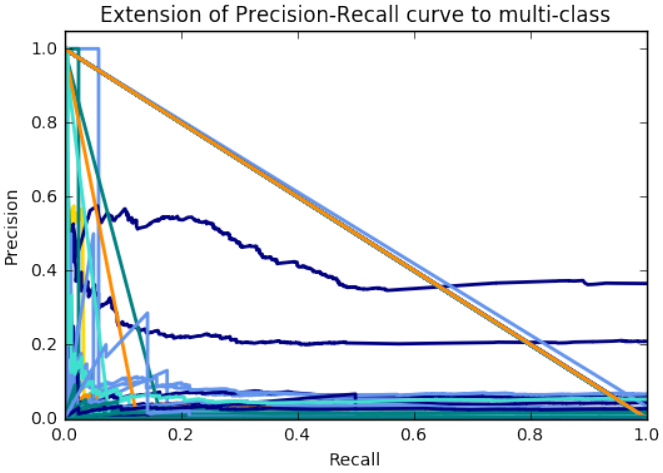

Figure 12. PR curve for Named Entity extraction (all labels)

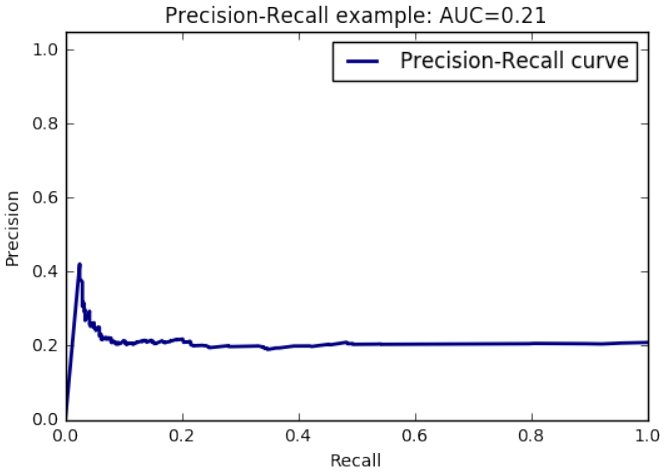

Figure 13. PR curve for Noun Phrase chunking (Label 1)

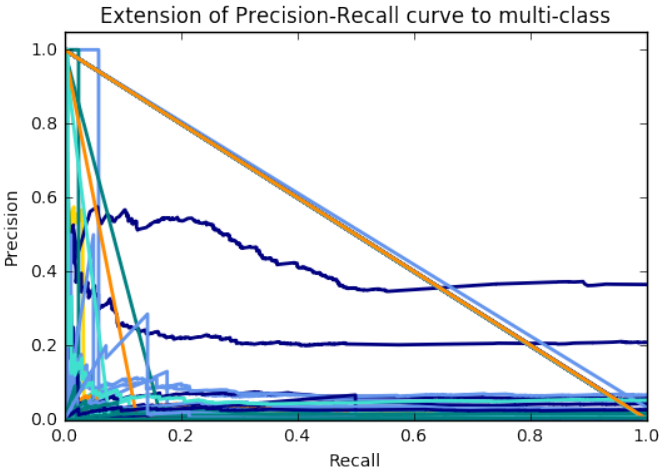

Figure 14. PR curve for Noun Phrase chunking (all labels)

SVM (RBF)
Time Taken(sec): 480.3060

3. **Noun Phrase chunking**
   Vocabulary Size: 1094 words

   SVM (Polynomial)

Accuracy	Precision	Recall	Time Taken(s)
99.0646	1.0	0.000288	232.6776

   SVM (RBF)
   Time Taken(sec): 885.3276

Interpretation: Recall is a performance measure of the entire positive part of a dataset whereas Precision is a performance measure of the positive predictions. The value of AUC (area under the curve) lies between 0 and 1 and a higher value denotes a better prediction model.

In our experiment, we found the value of AUC to be very small. All the three dictionaries have presented similar results, but the prediction with Named Entity recognition takes less time and consumes less storage compared to the other two [9].

5. Future Work

We would like to improve the efficiency of the data cleaning process by refining the text normalization process, fine tune the model parameters, to improve the precision and recall of the models.

We would also like to implement other Machine Learning techniques like Bayes Classifier, Artificial Neural Networks to train the model and compare the results with our observations.

Acknowledgments

We have put in considerable efforts to reach this stage of the project but it would not have been possible without the continuous guidance and support of many other individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to our Professor Markus Dickinson, for his supervision as well as for providing the necessary resources required for the completion of this project report.

References

[1] Wikipedia. Named-entity recognition — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 3-May-2017].

[2] Wikipedia. Tokenization (lexical analysis) — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 3-May-2017].

[3] Wikipedia. Part-of-speech tagging — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 3-May-2017].

[4] StackOverflow. In Natural language processing, what is the purpose of chunking?, 2017. [Online; accessed 3-May-2017].

[5] Ewan Klein Steven Bird and Edward Loper. Natural Language Processing with Python, 2017. [Online; accessed 3-May-2017].

[6] Wikipedia. Support vector machine — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 3-May-2017].

[7] David D. Lewis. Reuters-21578, 2017. [Online; accessed 3-May-2017].

[8] Karan Chopra. A One-VS-Rest SVM classification for documents in Reuters-21578 Data set, 2017. [Online; accessed 3-May-2017].

[9] Takaya. Introduction to the precision-recall plot, 2017. [Online; accessed 3-May-2017].