A Report on

Topic Detection

CS 6320

Natural Language Processing

Dr. Mithun Balakrishna

by

Bala Chandra Yadav [bxy140430] & Ravindhar Reddy [rxt140930]

Problem Description:

The recent explosion in textual information has created a need for methods to automatically organize text documents. New technologies such as the World-Wide Web and advances in speech-recognition have fuelled a significant growth in the range of textual information. This growth has engendered a drive for techniques to organize and classify large amounts of text. In this project, we try to address the problem of automatically organizing text documents into groups related by topic. The identification of the topic(s) that a document addresses increases our understanding of that document, the characteristics of the collection as a whole and the interplay between distinct topics. Considering this, we focus on detecting the topic of the given document.

Proposed Solution:

It is found that many of the current applications in place for topic detection are based on the clustering of the keywords from each topic. Though many new approaches have emerged over time for topic detection, the clustering algorithms continues to outperform these methods. Guided by this basic principle and fuelled with Natural Language Processing (NLP) techniques including Lexical, Syntactic and Semantic features, the current project has been implemented leveraging various NLP technologies and clustering algorithms. As an initial step, we have analysed various algorithms and techniques for each of the phase; to state a few we have considered TF-IDF's model as our initial add-on to base line model and extensively compared the model with another implementation of Normalized TF-IDF's, the resultant of which was again extensively analysed and compared with Multinomial Naive Bayes; Upon a considerable amount of variations, we decided to focus on Multinomial Naive Bayes as our clustering algorithm. Using NLP features grouped with Multinomial Naive Bayes is our model to effectively and efficiently detect the topic of the given document.

Examples:

1. Wizards repel Magic for season-opening win

The ball bounced on the rim seven times, glanced off the backboard, and bounced on the orange metal six more times as the seconds ticked off the Washington Wizards' 88-87 triumph. The anticipation from sold-out Amway Center escalated as the ball hung on the rim and a chance to keep a fresh season unblemished hung in the balance.
Finally, two arms, wrapped in different shades of blue sleeves, appeared in the picture. One tapped the orange leather ever so slightly as it sat on the cylinder and rolled off. The referees halted the sequence to declare the guilty arm had been navy, that of Wizards guard Bradley Beal, and counted the basket for the Orlando Magic, granting the hosts a one-point lead.

The above article has to be labelled as "Sports"

2. A search engine for cyberthreats

Keeping track of all the latest cyberattacks and who was behind them is hard for the average person, but it can be tough for those whose full-time job involves security, too.
That's where one Virginia start-up sees a business opportunity — managing the vast troves of data on threat intelligence when security experts need it the most.
ThreatQuotient, which operates out of AOL's Fishbowl Labs incubator in Dulles, is the brainchild of Wayne Chiang and Ryan Trost, both former employees of defense contractor General Dynamics.

The above article has to be labelled as **"Tech"**

Full Implementation Details:

Baseline System:

The baseline system randomly detects one of the topic from the list of trained topics. It does not take the leverage of NLP techniques and the classification is a disaster. We have intentionally kept the baseline to a toss of a coin model, as we used bag of words model to compare the accuracy of our actual model which is Multinomial Naïve Bayes. The Baseline system labels the same file with a different label when executed again.

Improvement Strategy:

Implementing the crude Multinomial Naïve Bayes wasn't yielding the accuracy as expected; so as our initial improvement strategy, we have reduced the document size (i.e. reduced no of paragraphs inside each document), which reduced the overlap of words with different topics, in effect increasing efficiency of our model. Also reducing the size of document fostered the effect of few NLP features like semantics. The semantic features like word disambiguation started showing more results after reducing the document size i.e. using the synonyms on reduced size documents helped to distinguish between topics more accurately.

Examples:

*The new representative of the **Obama administration** in **Haiti**, Ambassador Peter F. Mulrean said on Thursday that he wished for **elections** being organized by the **Provisional Electoral Council** (CEP) of Pierre Louis Opont to continue. Mulrean made the declaration while entertaining a **party** at his private residence in Port-au-Prince on Thursday. His words come at a time when an overwhelming number of Haitians and organizations, across many sectors, are calling for the **electoral** process to be scrapped and a transitional **government** for the organization of the **contests**.*

We can notice that, classifying this document using the keywords such as "Obama", "Haiti", "elections" yields better results than labelling the document randomly using the baseline method.

Finding Head word

As a process of leveraging the NLP features, we begin with finding the head words for the phrases and assign a specific weight to the words classified as 'head'. Particularly in this domain of documents where phrases or partial sentences does not have much weight to decide on the topic, the document is stating about. We did not see much of improvement by increasing the weight of head words. We

did see a few instances where head words did have a positive effect towards accuracy but in total, the head word recognition did not play a major role in increasing the accuracy.

Tokenize

The next process in our implementation was to tokenize the document, this being the basic and primary step we have given considerable amount of time to analyse and determine if we have accounted for all variations in tokenizing i.e. we could find that our data set does have many junk characters which was affecting our model and so removing those junk characters before we further process was helping and also increasing the weight of individual terms. We have added a slight variation in the tokenizing process for training and testing, i.e. for training data set, we consider formatting the data at line level and word level during tokenizing. The formatting at line level includes stopword removal, junk character removal, numeric data handling and few other string manipulations. The word level formatting includes to bring the case to uniform level, trimming, numeric data handling, range handling etc. The tokenize phase on test set does a straight tokenizing without completely formatting the line as the line will be further processed with exact case and characters for extracting other NLP features.

Examples:

*The new representative of the **Obama administration** in **Haiti**, Ambassador Peter F. Mulrean said on Thursday that he wished for **elections**…*

On Tokenization of the above document, we will get

[*The, new, representative, of, the, **Obama, administration,** in, **Haiti**, Ambassador…]*

As shown, rather on working on the document as a whole, tokenization will help us in finding the key words.

POS Tagging

Because the task in hand is topic detection and most of the functional words eg. ‘because’, ‘so’ does not have an impact on topic detection; we ignore the functional words. We leverage the POS Tagging feature to determine whether the word is either a functional or a content word. We saw a tremendous difference in accuracy after including the POS Tagger. The analysis helped us to understand that even though we remove stopwords, there will be words which can further classified as important and non-important; depending on the task and requirement we can either ignore the non-important words or have a less weight to those words. In our current implementation, we ignore the functional (non-important) words.

Examples:

How**/ WRB** are**/VBP** you**/PRP** today**/NN**. I**/PRP** am**/RB** doing**/VBG** great**/JJ**

Here the content words are: are, today, doing and great are consider as content words and the remaining words as functional words and these functional words are not considered as a part of the modelling.

Named Entity Recognition

Named Entity Recognition helps us in finding PLACES, ORGANIZATIN and PERSONS in a given sentence. We have implemented it using Stanford Named Entity Recognizer.

Obama/ PERSON
Google/ ORGANIZATION
Richardson/ PLACE
The basic idea is a few persons and places are strongly associate with a topic. Like, how Obama is associated with Politics and Jordan with the basketball. This feature is used while labelling a test document as if we found a match between the word in the test sentence and the given topic and if the word is a Named Entity, we give more weight to the topic unlike a non-named entity word matching.

Examples:

*At least 14 Haitians dead, dozens injured and dozens of political arrests, are not worth $30 million to the **Obama** administration. It said this week it wished the electoral process it is funding in **Haiti**, that which has registered historic levels of fraud, to continue to its end.*

*The new representative of the **Obama** administration in Haiti, Ambassador **Peter F. Mulrean** said on Thursday that he wished for elections being organized by the Provisional Electoral Council (CEP) of **Pierre Louis Opont** to continue….*

Here the paragraph gets more weightage for being labelled as "Politics" as the words "Obama" ," **Peter F. Mulrean** " are mostly associated with "politics" and is a named entity.

Synonym Usage:

As we reduced our document size, the synonyms of the words started playing a major role in topic detection, the effect of including the synonym consideration was prominent. It is understandable that a word which is crucial in topic detection may have multiple meanings (polysemy) and the same word may not occur in a test document but if a word which is a synonym to the crucial word appears in the test document, the test document is most probably belongs to the same topic and considering synonyms will help us find such cases.

Examples:

We can find that the word 'car' has following synonyms:
automobile
cable_car
auto
railcar
machine
motorcar
railroad_car
elevator_car
railway_car
gondola

While trying to label a document from auto domain, even if it doesn't contain the word *car* exactly, matching with synonyms will help us classify the document correctly as auto.

Stemming & Lemmatization

The goal of lemmatization and stemming is to reduce inflectional or derivational forms of a word to base form. But Stemming is much simpler, smaller and usually faster than lemmatization and for our application stemming outperformed lemmatization in terms of both performance and accuracy. To illustrate this, consider the following cases:

Word	Lemma	Stem
Meaning	Mean	Mean
Meaningful	meaningful	Meaning
Meaningfully	meaningfully	Meaningfully
meaningfulness	meaningfulness	Meaning
Meaningly	Meaningly	Meaningli
Meaningness	meaningness	Meaning
Variations: 6	Variations: 6	Variations: 4

As seen above Stemming is more effective in reducing the inflectional forms to their base forms.

Multinomial Naïve Bayes

As we know, Naïve Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem with strong and naïve independence assumptions. Being one of the most basic text classification techniques, the implementation of it was very simple and straight forward. Despite the naïve design and oversimplified assumptions that this technique uses, Naïve Bayes as ascertained was performing well in the current topic detection implementation. Because we had probabilities in least decimals, applying the logarithm to the Naïve Bayes equation worked well in our case.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

Here ***Prior***: *The probability that the consider category occurs*

Likelihood: *The probability that the consider word belongs to the category*

Evidence: *The probability that the considered word occurs*

Our multinomial implementation is:

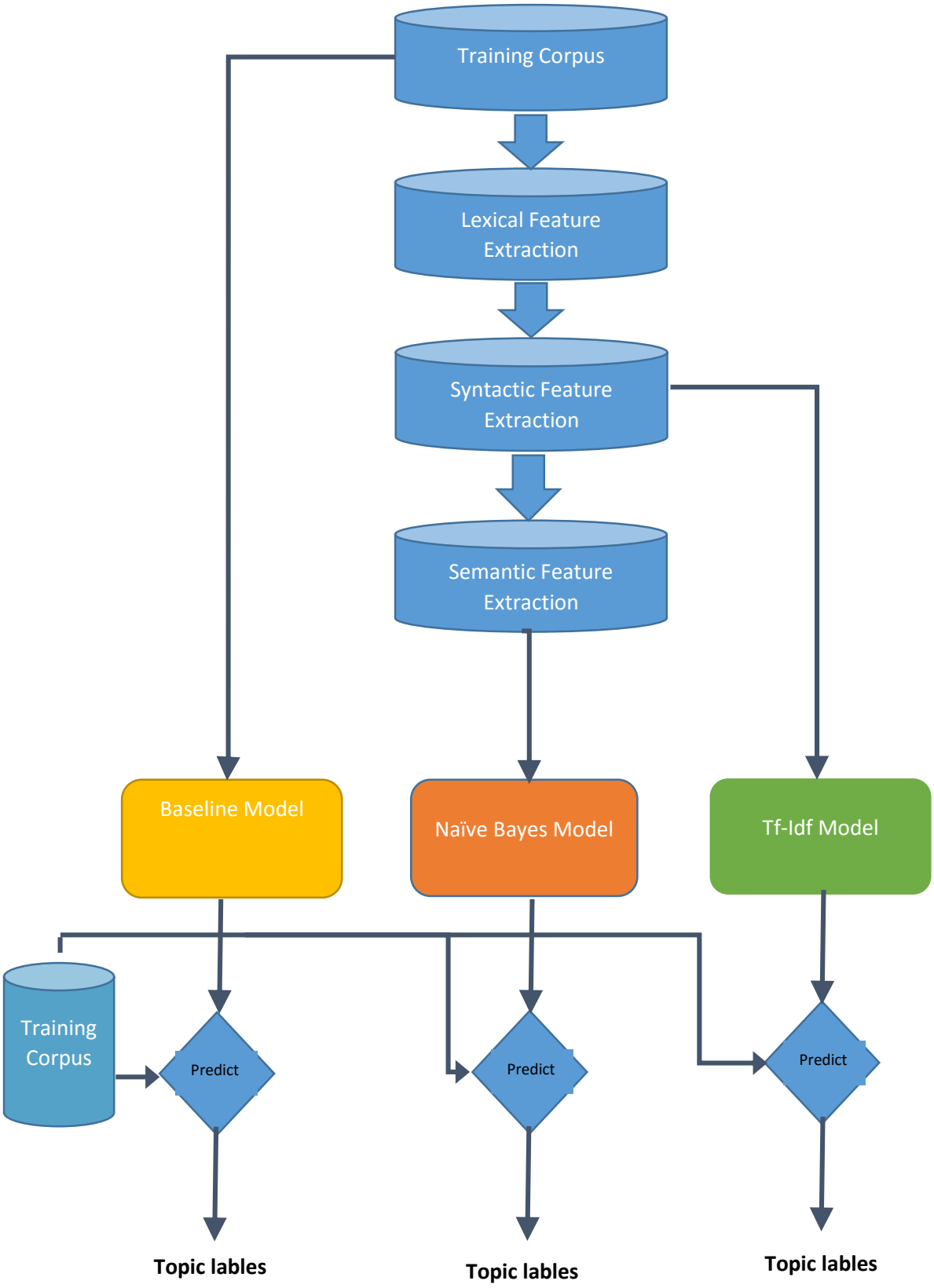
$$\text{Topic}_{MAP} = \text{argmax}_{c \in \text{Topics}} \; P(x_1, x_2, x_3, \ldots x_n \mid c)\, P(c)$$

$$\text{Topic}_{NB} = \text{argmax}\; P(c_i)\, \pi_{x \epsilon X}\, P(x/c)$$

In our implementation, as the probability values are very low, we have applied log function to normalize them.

Architectural Diagram

Below is the overall architectural diagram for our model.



Results:

The current implementation was done based on newsgroup dataset with topics

alt.atheism
comp.graphics
misc.forsale
rec.autos
rec.sport.baseball
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns .

With considerable amount of training, the model was detecting the topic for the documents which aren't alone news articles but also for few emails, we cast this credit to NLP techniques we have implemented in our implementation.

We have tested the program on 5 documents from each category, following are the accuracy results:

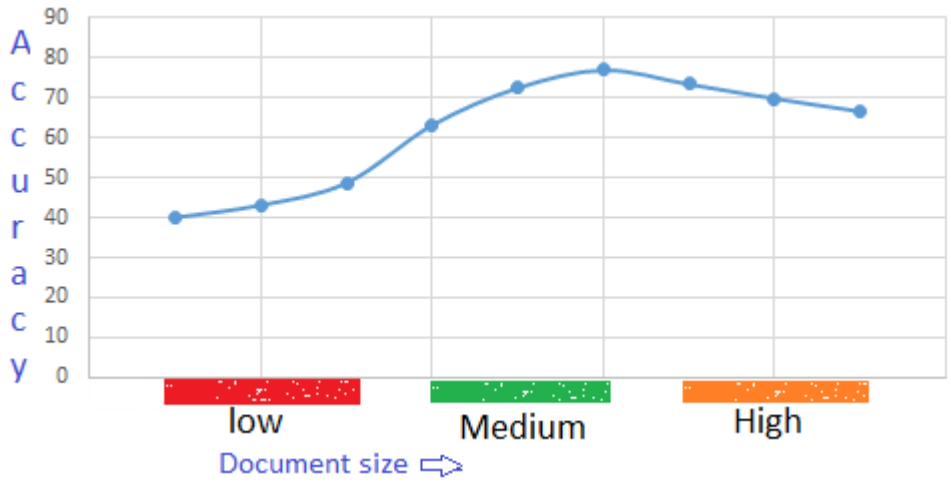
Baseline method: 20.56 (is keep on changing)

Tf-idf method: 50.21

Naïve Bayes method: 77.25

Analysis:

1. It is found that the accuracy varies based on the document size. When the document size is very high, results are not better because of word overlapping. On decreasing the size, the accuracy got better. But if we decrease too much, the accuracy started going down as it may lead to overfitting.



Problems Encountered

1. The dataset which was taken for the current model had many junk characters and the text was also corrupt in most the documents yielding us unwanted and unexpected results. We had to improve our

filtering criteria and prior to even filter, we had to pre-process the data set as a cleansing step. Though we have not included that module as we have done that pre-processing once on the entire data set. But it took us considerable amount of time to figure out the issue as the documents which were corrupt were a few in between many.

2. Another daunting issue was the overlap of words between topics, most of the articles had very coarse grained topics and up on that most of our probability calculations were very close. We addressed this issue by reducing the size of the documents to half. This has really helped to improve the accuracy of our model.

3. The other problem where we spent most of our time was when did not see any improvement after including few of the NLP features (discussed in earlier sections), Though there was not a much of improvement, the analysis we have done for the same helped us to understand the topics better. Realizing the reason for not having an improvement was itself a learning point for us from this implementation.

Pending Issues & Potential Improvements

1. The current implementation is rigid in a way that it will try to classify the document to one of the topics it has, it can be extended to potentially detect/generate the topic from the document's content.

2. Because we have developed the model based on news group articles, the classification approach we have taken works in most of the cases but if the data set is not from news group but instead from another source, the sentimental analysis will play a major role and we see that integrating Latent Dirichlet allocation will effectively improve the model. This is more of an extension rather than a pending issue.

Programming Tools

Language: Java

Compiler/IDE: JDK, Eclipse

Libraries/Tools: Stanford CoreNLP, Stanford Named Entity Recognizer (NER), Stanford Log-linear Part-Of-Speech Tagger, MIT Java Wordnet Interface – Princeton Wordnet