TL;DR Automatic Text Summarizer

Rachit Jain

140905464

rachit.jain2706@gmail.com

Aneesh Joshi

140905442

aneeshyjoshi@gmail.com

ABSTRACT

With the increased data that is present today, there is an increasing need for short summaries. We suggest a Naïve Bayes approach to Automatic Text Summarization(ATS) using term frequency-inverse sentence frequency(tf-isf), sentence length and number of nouns in a sentence. These features will give us an idea of which sentences are summary sentences and which are not.

PROBLEM STATEMENT

With the large amount of data that is present today, there is an increased need for summarization of data. It is faster to read a summary of any article rather than reading the whole article.

Summarization, as done by humans, involves reading and understanding an article, website, or document to find the key points. For humans, generating a summary is a straightforward process but it is time consuming.

As of now there exists applications like inShorts which uses humans to summarize news into an article of at most 60 words. We plan on improving this by automating the summarization.

INTRODUCTION

Automatic summarization is the process of reducing a text document with a computer program in order to create a summary that retains the most important points of the original document.

There are two approaches to automatic summarization namely extractive and abstractive:

The **extractive** approach works by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. These summaries do not guarantee a good narrative coherence, but they can represent an approximate content of the text.

In contrast, the **abstractive** approaches build an internal representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original. Ideally, abstractive auto summarization is preferred as they more meaningful summaries. But making such a summarizer is a hard task. We have focused on making an extractive summarizer.

LITERATURE SURVEY

Currently, most of the research is oriented towards extractive summarization as abstractive summarization is an AI hard problem and goes in the domain of Natural Language Understanding.

A common link between all summary techniques is the use of term frequency which believes that if a word occurs more often, it must be important.

We considered several approaches like:

i.) Multilayer Perceptron
ii.) Text Rank Algorithm
iii.) Word Embeddings through Word2Vec and GloVe

SCOPE

An automatic text summarizer has a wide range of approaches.

It can be used:

- In summarizing legal cases which tend to be long and repetitive.
- To summarize news articles and get the most important sentences.
- To get query based summarizations to study large texts
- For Information retrieval systems of Social Networks.

GAPS

The current techniques are focused a lot on statistical data like tf-idf but this doesn’t capture the semantics of the sentence.

We have partially overcome this by considering features of number of nouns in a sentence.

DATASET DESCRIPTION

We collected a dataset which contains Australian legal cases from the Federal Court of Australia (FCA). The cases were downloaded from AustLII (http://www.austlii.edu.au). The dataset included all files from the year 2006,2007,2008 &2009.The dataset was built to experiment with automatic summarization.
Catchphrases are found in the document. We used the catchphrases as gold standard for our summarization experiments. Citation sentences are found in later cases that cite the present case. We use citation sentences for summarization. Citation catchphrases are the catchphrases (where available) of both later cases that cite the present case, and older cases cited by the present case. Citation classes are indicated in the document, and indicate the type of treatment given to the cases cited by the present case.

The dataset was present in an XML format which had to be preprocessed and put into a text file. Two folders were created which contained the full text contained the actual text and training class which contained the summary for each document.

OBJECTIVE:

To obtain a semantically coherent summary of a given large text document. The amount of reduction is based on a compression ratio which is specified by the user of TL;DR.

METHODOLOGY

TL;DR works in the following phases:

1. **<u>Preprocessing:</u>**
   The data was Australian Legal Cases which provided the data in an XML format. This XML file was converted to a text file which was free of all XML tags and unnecessary data.
   a.) The extracted text was tokenized into sentences using the Natural Language ToolKit (nltk).
   b.) Stop words like ‘a’, ‘the’ were removed from the sentences.
   c.) The sentences were tokenized into words.
   d.) All characters except [a-zA-Z] were removed

2. **<u>Feature Extraction:</u>**
   The following features were selected:
   a.) <u>Term Frequency:</u> It is the number of times a word occurs in the text corpus. This feature selection is based on the idea that a more frequent word tends to be more important.

b.) Inverse Sentence Frequency:

$$ISF(t_i) = \log_2(N) - \log_2(n_i) + 1$$

where

N: total number of sentences

n_i : the number of sentences with term i

t_i : term

c.) Number of Nouns:

This is the number of nouns that occur in a given sentence. The idea behind this selection is that a sentence with more nouns tends to be more important.

d.) Sentence Length:

This feature was selected to differentiate between sentences. Small and large sentences tend not to be in the summary.

3. Data Normalization:

The extracted data is then normalized so as to bring all of it in the same scale. Ensuring standardized feature values implicitly weights all features equally in their representation.

4. **Setting Sentence Features:**
   Since we are applying sentence summarization, we need features for each sentence. We chose to represent each sentence by the average of the values of the features extracted above wherever applicable.
   For example, tfisf for a sentence will be the mean of tfisf of all its words.
   After this each feature value is compared with the mean of that feature. If the feature is above the mean it is set to 1 else it is set to 0.
   So our resultant feature vector is of the form:
   [0,1,0] for [2,4,1] with average values as [4,1,3]

5. <u>Training</u>:

We then trained a Bayes Classifier to differentiate between summary words based on the given data.

P(label | features) = P(label) * P(features | label) / P(features)

P(label) is the prior probability of the label occurring, which is the same as the likelihood that a random feature set will have the label. This is based on the number of training instances with the label compared to the total number of training instances. For example, if 60/100 training instances have the label, the prior probability of the label is 60 percent.

P(features | label) is the prior probability of a given feature set being classified as that label. This is based on which features have occurred with each label in the training data.

P(features) is the prior probability of a given feature set occurring. This is the likelihood of a random feature set being the same as the given feature set, and is based on the observed feature sets in the training data. For example, if the given feature set occurs twice in 100 training instances, the prior probability is 2 percent.

P(label | features) tells us the probability that the given features should have that label. If this value is high, then we can be reasonably confident that the label is correct for the given features.

Where the label refers to whether the sentence is a summary sentence or not.

The feature vector is:

[tf*isf, number_of_nouns, sentence_length]

RESULT AND ANALYSIS

Result:

We split our data into training and testing sets.

We found that we on an average we have an **accuracy of 67.66%**

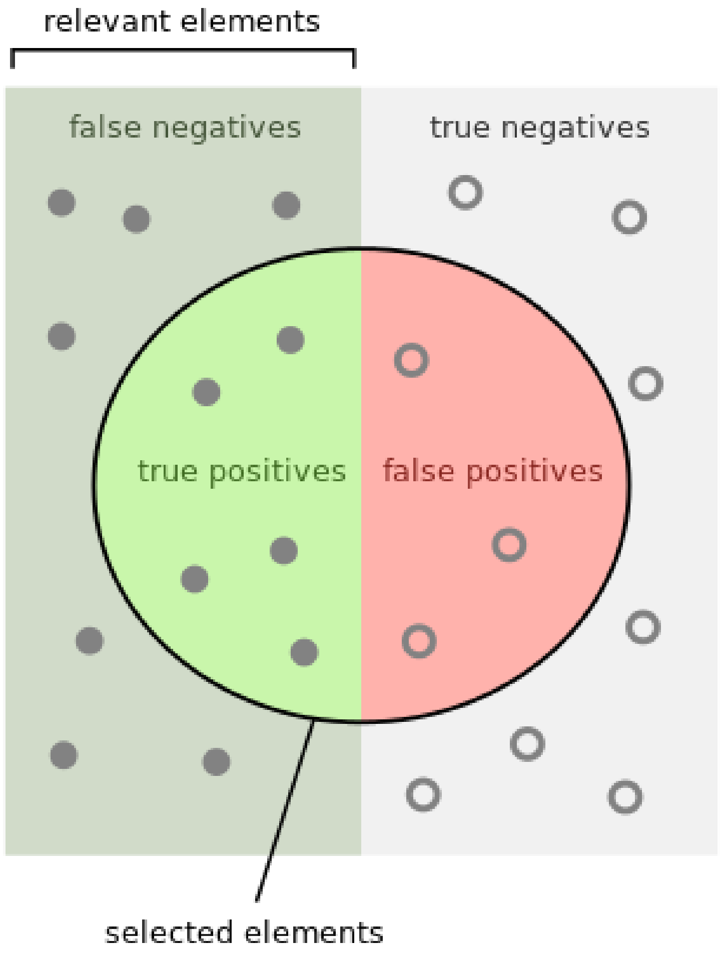
Analysis:

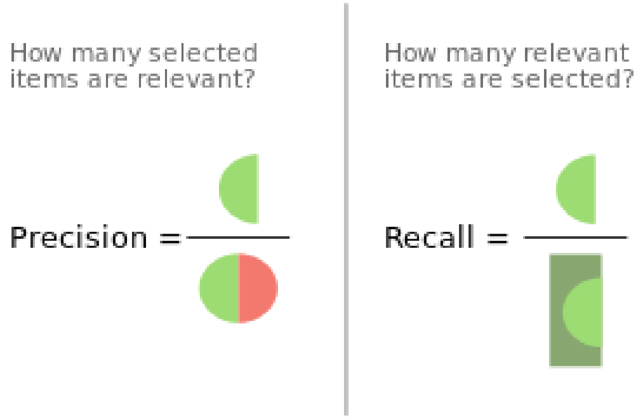
We are analyzing the data based on the following measures:

1. **Precision** is the fraction of retrieved instances that are relevant
2. **Recall** is the fraction of relevant instances that are retrieved.
3. **F1 Score**: The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system.

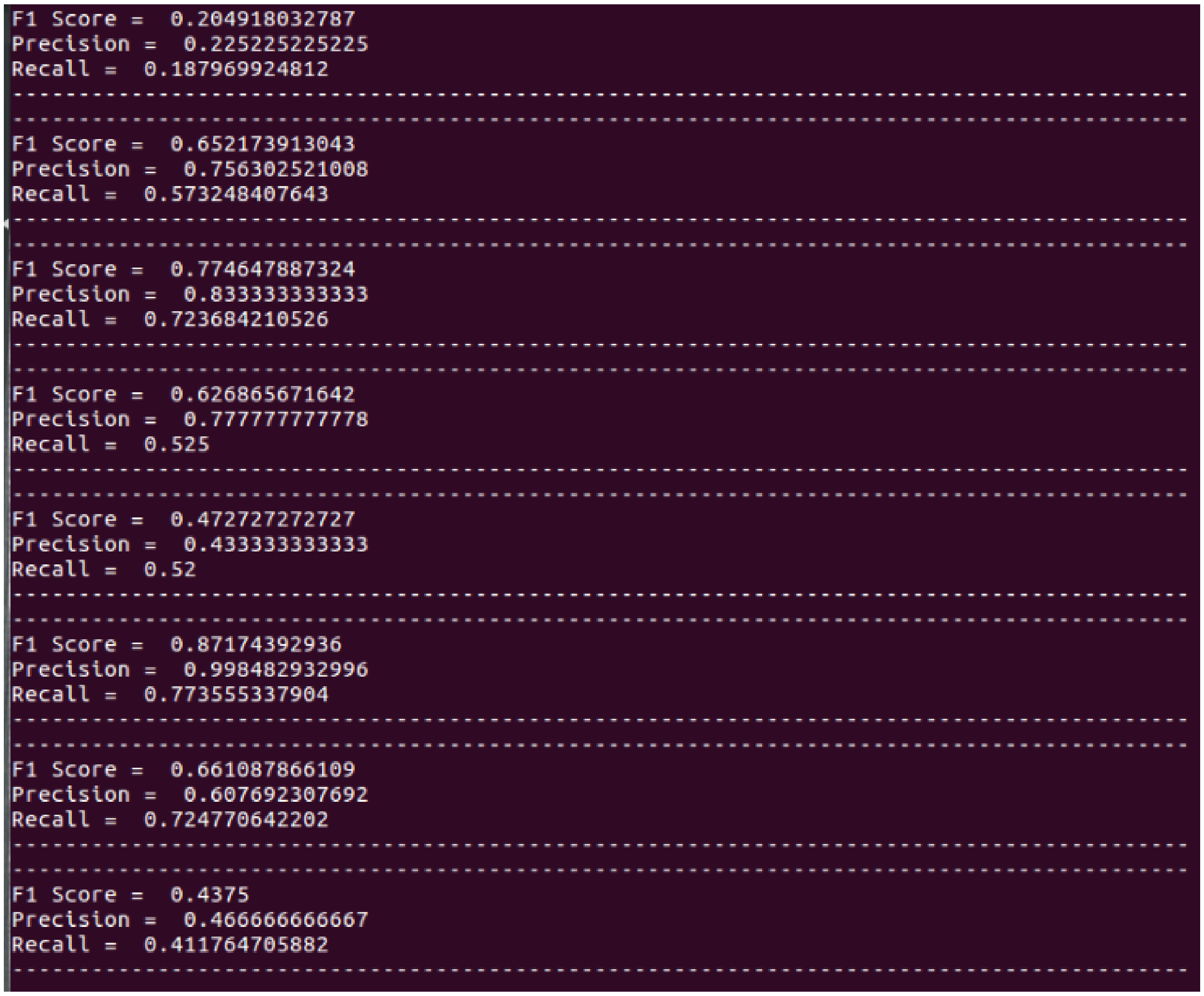
$$\text{F-measure} = \frac{2*\text{precision}*\text{recall}}{\text{precision} + \text{recall}}$$

<u>To calculate Precision and Recall:</u>




How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =

On running our classifier on the testing data we got the following values for precision, recall and f1 score:



```
F1 Score =  0.204918032787
Precision =  0.225225225225
Recall =  0.187969924812
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.652173913043
Precision =  0.756302521008
Recall =  0.573248407643
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.774647887324
Precision =  0.833333333333
Recall =  0.723684210526
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.626865671642
Precision =  0.777777777778
Recall =  0.525
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.472727272727
Precision =  0.433333333333
Recall =  0.52
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.87174392936
Precision =  0.998482932996
Recall =  0.773555337904
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.661087866109
Precision =  0.607692307692
Recall =  0.724770642202
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
F1 Score =  0.4375
Precision =  0.466666666667
Recall =  0.411764705882
--------------------------------------------------------------------------------
```

CONCLUSIONS

Our Naïve Bayes Approach leads to an acceptable summary which has an accuracy of 67.66%. Statistical approaches like term frequency inverse sentence frquency and semantic features like number of nouns make for good features for classifying sentences.

FUTURE WORK

This summarizer could be further improved by increasing more thresholds for the features. Now we are setting it to either 1 or 0 but another possible way is to split it into different ranges to get more values like 0,1,2,…

This threshold could be decided by a Neural Network.

REFERENCES

- Python Text Processing with NLTK Toolkit 2.0
- http://sebastianraschka.com/Articles/2014_naive_bayes_1.html
- http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- http://www.cs.duke.edu/courses/spring14/compsci290/assignments/lab02.html
- http://bdewilde.github.io/blog/2014/09/23/intro-to-automatic-keyphrase-extraction/