CSCI - B 659: Assignment #1

Due on Sunday, February 26, 2017

Prof. Cavar

poosingh/rraavi/vpatani

Contents

Text Corpus Selection **3**

Approach 1 - Bag of Words **4**
- Model Selection . . . 4
- Optimise Feature Extraction . . . 4
- Making the Training/Testing Model . . . 4
- Results . . . 4

Approach 2 - Naive Bayes **5**
- Model Selection . . . 5
- Optimisation Feature Extraction and Modeling Data . . . 5

Approach 3 - Naive Bayes - TF-IDF **6**
- Optimisations and improvements . . . 6

Comparison **7**

References **8**

Text Corpus Selection

We selected our corpuse of text to be **Driver.**

From the below given meanings (**NLTK**):

1. The operator of a motor vehicle
2. Someone who drives animals that pull a vehicle
3. A golfer who hits the golf ball with a driver
4. (Computer science) a program that determines how a computer will communicate with a peripheral device
5. A golf club (a wood) with a near vertical face that is used for hitting long shots from the tee

We are only interested in the **first**, **second** and **fourth** meaning of the words for our disambiguation.
We have followed 3 approaches to disambiguate the given word:

- Bag Of Words Approach
- Naive Bayes
- Naive Bayes with TF - IDF

Approach 1 - Bag of Words

Model Selection

Our Bag Of Words approach comes from the family of Lexical Sample Task. This contains a small set of pre selected words on which we train our model.
We first build a vocabulary of words at the begining, find synonyms and antonyms for it, and put them in a vocabulary collection. This is our feature set. Each vocabulary word (or its related meaning) represents a position in the feature vector.

Optimise Feature Extraction

While preparing the vocabulary, we use stemming (sometimes stemming adversely affects our output but we selected to go ahead with it anyway) and lemmatisation.
We pass each word of the vocabulary through Stemming and Lemmatisation, followed by looking for similar (synonyms and antonyms) words. This will also contain the word to be disambiguated.

Making the Training/Testing Model

We read each line of the training set and check for similar words from the vocabulary and if it does exist then we mark that location of feature vector as 1, which in the hindisght means that this sentence represents some relation (maybe it is negatively co-related, but it is) and it is good to learn.
We prepare the test set in a similar fashion.

Results

Once TimBL trains and test data is given, we need to interpret the result.
If on given test data, TimBL classifies the data as 0, then it has correctly classified that data point and 1 means it has incorrectly classified that data point. We had fairly variable results.
As a whole test data set, it has given us an accuracy within in a range of 0% to 60%, due to unpredictive nature of IB1 algorithm.
On an average we get 5/10 results correctly classified with the correct definition of that word.

Approach 2 - Naive Bayes

Model Selection

The Naive Bayes assumption comes with a small caveat that the independence assumption discards any relation between features (words) or such entities. We cannot strongly (or at all) represent relation between features and that may sometimes be a big problem but we have gone ahead and used it.

Optimisation Feature Extraction and Modeling Data

For this classification, we generate frequency profiles for each class and use term frequency for evaluating the scores. To generate the frequency profiles, we tokenize the words for which we use NLTK. Consider a scenario where token ends with a punctuation and the same token doesnt end in punctuation. While counting the frequency for this token, we might want to consider as one token occurring twice, instead of considering them as two different tokens altogether. For this reason, we use regular expressions to pick only the characters in the alphabet and tokenize them and then determine the term frequency. Also, we wont be calculating the term frequencies for stop words. Once we have the frequency class profiles, we proceed to try and classify the unknown text.
The unknown text will be tokenized and for each token of the unknown text, we find out the occurrence of that token in the class and then calculate accordingly. This calculation must be done for all the tokens of the unknown text against each class. Once we have two final scores, we evaluate to find out the higher score and decide which class the unknown text is more likely to fall under.

Approach 3 - Naive Bayes - TF-IDF

Optimisations and improvements

One way to optimize our classification is by refining how we calculate the frequency scoring. One other way to do this by using term frequency inverse document frequency, simply put tf-idf scoring.

$$TF - IDF = tf \times idf$$
$$idf = \log(N/df)$$
$$where, N \rightarrow Totalnumberofdocuments$$
$$df \rightarrow documentfrequency$$

The idea is that, there might be words occurring more frequently and having no significance to the context. To avoid having higher scores for such words, we use tf idf to limit. It reduces the scoring of such words, depending on the word/term appearing in multiple documents of collection. Also, in our case, the two text files will be like two Text Collections and each line is like one document.

Comparison

References

1. Stanford Slides
2. PYWSD
3. Wikipedia - Naive Bayes
4. Stanford NLP

References continued on next page...