Phrase detection

Project proposal for Machine Learning course project

Suyash S Shringarpure
suyash@cs.cmu.edu

1 Introduction

1.1 Motivation

Queries made to search engines are normally longer than a single word in length. In fact, [3] show in an analysis of Altavista query logs that approximately more than half of the queries have length more than one. Conventional IR methods propose intersection of the occurence lists for each word in a phrase, using various methods to reduce the time required for this task. Thus, the average response time of a search engine can be reduced by indexing terms which have length more than one. However, in a index which has N words, there are potentially N^2 bigram phrases and N^3 trigram phrases and so on. Clearly it would be infeasible to index all possible bigrams, trigrams etc. We would therefore like to obtain such phrases which are "meaningful"- which we define as their co-occurence being not merely due to chance.We will explore a probabilistic method of finding meaningful phrases in the Twenty-Newsgroups text corpus.

1.2 Problem Definition

The input is a text corpus containing (preferably) a large number of documents. The aim is to extract from them "phrase" - which are also described as collocations in NLP literature. These refer to sequences of words which occur together more times than we would expect co-occurence due to chance. To do this, we will convert the bigrams into feature vectors using features described later. However, all the data that we have is unlabeled. We will use some heuristics (described later) to eliminate ngrams that are obviously not phrases. The problem can be solved by both unsupervised and semi-supervised learning methods. We will only solve the case of $n = 2$ ie, bigram phrases, due to computational limitations. We will try to cluster/classify (using unsupervised and semi-supervised methods respectively) these features vectors. Class 0/negative examples will be used to refer to bigrams that are not phrases and class 1/positive examples will be used to refer to bigrams that are phrases.

2 Related Work

2.1 Phrase detection in general

There is very little published literature on finding phrases in general from a corpus. A paper by [1] describes the use of a neural network to perform unsupervised phrase detection. They use a concept called "mutual significance" . The mutual significance of two tokens *i* and *j* is defined as

$$S(i,j) = \frac{p(i.j)}{p(i)p(j)}$$

with the probabilities replaced by the ratio of the count of the token to the the total number of tokens in the corpus. This is also commonly referred to as “pointwise mutual information” or PMI.

2.2 Noun Phrase Detection

Noun phrase detection is a topic that has been studied a lot in the NLP community. There are many different methods proposed for noun phrase detection, so of which we will mention. Traditional noun-phrase detection used rule-based systems, parse trees etc to detect noun phrases from corpora. A lot of methods involve using part-of-speech tagging on the corpus. However, we would like a method that is scalable to large corpora and hence these methods are not very useful. For this we will make use of statistical tests and other similar probabilistic and linguistic ideas in our phrase detection algorithm.

3 Method

3.1 Types of learning methods used

We will use both unsupervised and semi-supervised learning methods for phrase detection. Both are briefly described below:

3.1.1 Unsupervised learning

Since the ngrams extracted from the corpus are available unlabeled, using a clustering algorithm is the obvious approach. Each ngram can be converted into a feature vector(using features described later), and a simple clustering algorithm such as k-means(with k=2) can be run on the data. This, however, might involve setting the feature weights by hand, which is not desirable. The performance of the algorithm might also be affected by the fact that the data is skewed (ie. there are much more non-phrases than phrases).

3.1.2 Semi-Supervised Learning

In this method, some of the more obvious phrases and non-phrases can be labeled by hand. An EM algorithm could be used with, say, a logistic regression classifer to then classify the remaining ngrams extracted from the corpus. The advantages of this method are:

- Feature weights need not be set by hand but can be learned in the EM algorithm.
- The method will return a weight vector for the features that will make classification of a new ngram easy by just finding the dot-product of the feature vector for the new ngram with the obtained weight vector.

3.2 Features used

The features of each ngram that will be used to compute the feature vector are listed below:

- Parts of speech tags(or a “reject-words” list): The idea of using part of speech tags is an old one, as mentioned previously, but is slow and not scalable. This will be replaced by having a “reject-words” list containing prepositions, articles, punctuations etc. If all words in a ngram come from the list, then clearly the ngram is not a good phrase. In the experiments, a stopword list of 145 stopwords is used. The list can be found at the end of the report. We will not show any results with a POS tagger since the increase in computational time required for tagging makes the method infeasible for having an efficent algorithm. We use the fraction of words in the bigram that belong to the stopword list as a feature taking values $0, 0.5, 1$.

- Fraction of numbers: We use as a feature the fraction of words in the bigram that are pure numbers(integers or decimal numbers). Possible values are 0,0.5 or 1.
- Fraction of capital words: Another feature used is the fraction of words in the bigrams that contain only capital letters. This feature also takes values from among 0,0.5 and 1.
- Scores from significance tests: All the significance tests used test the null hypothesis that the words in the bigram occur independently against the alternative hypothesis that their occurence is dependent on each other. We use the scores from the test directly as features, and they take all real values. The tests we use are:
  - The t-Test.
  - Pearson's chi-square test.
  - Pointwise mutual information

  The advantage of using these tests as features is that they use only occurence information for the bigram and its components, which is easily accesible to us from the unigram index(since it is required only once for feature computation).
- Occurence in a query log: A feature that we will not be using, but one that would be very useful, is the occurence of an ngram in a query log. By thresholding the occurence above a pre-set level, information could be obtained about whether a ngram is queried often enough.(Clearly, a meaningless ngram would be queried very rarely, if ever).

4 Experiments

4.1 Preprocessing of data

The cleaning of the twenty-newsgroups data involves removal of the email headers from the messages in the Twenty Newsgroups dataset so that they contain only text. The proposal described that used Lucene could be used to index the data and obtain counts for the ngrams and their component words. However, Lucene is a little more complex and has more functionality than required in the project. A simpler way of obtaining all the ngram statistics we require from the corpus is the Ngram Statistics Package(NSP) by Ted Pedersen(University of Minnesota). In our dataset, we will consider only those bigrams which occur at least twice in the twenty newsgroups data.

4.2 Heuristic labeling

For the semi-supervised learning, we use some heuristics to label mostly negative examples. The heuristics are:

- Capital bigrams: All bigrams which consist of only capital letters are labeled 0.
- Numbers: All bigrams of the form “number number” are labeled 0.
- Stopword/Punctuation bigrams: All bigrams containing stopwords/punctuations are assumed to have label 0 and their feature vectors are excluded from the data.

Also, 5 meaningful terms are chosen by inspection and labeled 1 for the semi-supervised learning. The reduced dataset contains 186,829 bigrams, down from an original size of 1,124,260. The drastic reduction in size is mainly due to the presence of punctuations or stopwords in the bigrams. The results shown below are on this data.

4.3 Results and Evaluation

Ideally, we would like to evaluate the classification of bigrams as phrases/non-phrases by observing whether positive labeled bigrams have high frequency in query logs. Another method might be to observe the number of results a search query for a particular bigram returns. However, both these methods are complex(due to logistic reasons), hence we will only visually inspect the results for evaluation purposes.

4.3.1 Unsupervised learning-Kmeans

We used Weka for running Kmeans on the data. The results from the algorithm are:

Class	Bigrams assigned
0	25161(13%)
1	161668(87%)

Table 1: Kmeans clustering

These results seem to suggest that Kmeans does a loose clustering of the 1-cluster.ie. lots of non-significant bigrams are also being classified as phrases. For eg, about 6000 bigrams starting with numbers are labeled as phrases by the algorithm. Table 2 shows some bigrams labeled 1(alphabetically near the “t” bigrams for the EM) by the algorithm.

4.3.2 Semi-supervised learning- EM with logistic regression

We use the modification of the Expectation maximisation proposed by [2] to train a logistic regression classifer on the data. As mentioned earlier, this directly gives us feature weights that can be used to evaluate whether a new ngram is a phrase or not. We evaluate this method by tabulating the top bigrams sorted in descending order based on the probablity of being a phrase(after removing non-dictionary words and junk bigrams). The algorithm is: We run two variations stated below:

```
Run logistic regression on labeled examples
Use trained classifier to label all available examples
while Log-likelihood is increasing do
    Update weights.
    Relabel all examples with new weights.
end while
```

- Constrained: In this method, at the end of each relabeling step, the class values for the labeled examples are forced to their corresponding labels.
- Unconstrained: Here the algorithm runs without having to maintain consistency of labels for the labeled examples.

Tables 3 and 4 summarize the results of clustering by the variations.

Bigrams in cluster 1
tetanus toxoid
tetanus toxoids
textbooks printed
textbook errata
texts Misner
texts initially
texts qualitywise
textual data
textual types
textural analysis
textures aerial
textures associated
textures datafiles
texture files
texture library
texture map
texture mapped
texture mapping
texture maps
texture rule
texture rules
text ASCII

Table 2: Bigram near "t" marked as phrases by kmeans

Class	Bigrams assigned
0	107270
1	79559

Table 3: Clustering by constrained EM

4.3.3 Evaluation

While both variations give a similar split of the data into the two classes, the constrained method is marginally better than the unconstrained one. The unconstrained method labels around 9000 bigrams starting with numbers as phrases, while the constrained methods labels only 1000 such bigrams as phrases. Tables 5 and 6 show the top dictionary phrases among the bigrams labeled by each algorithm.

5 Conclusion

Using statisticial tests to generate features gives us a fast and reasonably accurate algorithm for determining meaningful collocations in text. A semi-supervised EM algorithm gives tigher results than unsupervised Kmeans and is advantageous since it can be coupled with almost any simple classifer. The collocatiosn thus obtained can be indexed based on their frequency in query logs to improve speed of indices in a search engine. However, we could significantly improve performance by using query logs and a dictionary to filter out meaningless ngrams, a task we are currently doing by inspecting the output.

Class	Bigrams assigned
0	110110
1	76719

Table 4: Clustering by unconstrained EM

Bigram	Probability
texture maps	1.000000
texture mapping	1.000000
texture mapped	1.000000
textures datafiles	1.000000
textures aerial	1.000000
textural analysis	1.000000
texts qualitywise	1.000000
texts Misner	1.000000
textbook errata	1.000000
textbooks printed	1.000000
tetanus toxoids	1.000000
tetanus toxoid	1.000000
tests EEG	1.000000
testimonies concerning	1.000000
testicular cancer	1.000000
testament passages	1.000000
terrorize westerners	1.000000
terrorist zionists	1.000000
terrorist underworld	1.000000
terrorist ultimatum	1.000000
terrorist hideout	1.000000
terrorist camp	1.000000
terrorist assassins	1.000000

Table 5: Top phrases with constrained EM

References

[1] R.C. Murphy. Phrase detection and the associative memory neural network. *Neural Networks, 2003. Proceedings of the International Joint Conference*, 4:2599– 2603, 2003.

[2] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[3] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

Bigram	Probability
textures datafiles	1.000000
textbook errata	1.000000
tetanus toxoids	1.000000
tetanus toxoid	1.000000
terrorize westerners	1.000000
territoriality instincts	1.000000
terrestrial ores	1.000000
terminating resistors	1.000000
terminated unexpectedly	1.000000
terminally irony	1.000000
tentative pending	1.000000
tendon sheath	1.000000
tenant farmers	1.000000
tempest shook	1.000000
teleoperated prospecting	1.000000
teenage spotty	1.000000
teenage offenders	1.000000
ted frank	1.000000
technological advancements	1.000000
tear gas	1.000000

Table 6: Top phrases with unconstrained EM