Clustering documents with vector space model using n-grams

Klas Skogmar, d97ksk@efd.lth.se
Johan Olsson, d97jo@efd.lth.se

Lund Institute of Technology

Supervised by:
Pierre Nugues, Pierre.Nugues@cs.lth.se

Abstract

This paper describe a method to cluster documents using the linear space algorithm together with unigrams, bigrams, trigrams and n-grams in an attempt to enhance the clustering performance compared to unigrams only. Our tests did not reveal an improvement, but shows that the technique has a potential, especially in certain specific areas like finding citations or versions of the same document. The extra effort required for implementation and the speed loss could make it less interesting however.

Introduction

This paper is the result of a project done for a course in language processing at Lund Institute of Technology. The project was mandatory, but the topic of the project was optional. We chose to use our newly acquired knowledge on the benefits of n-grams versus unigrams, but on a different area, namely: document clustering.

We wanted to use basic standard algorithms for everything, because we wanted to focus on the differences that n-grams versus unigrams could make for the accuracy of clustering. Therefore we chose the vector space model for determining similarities between documents, and k-means for clustering.

We expected that the benefits of our suggested solution would be that it enhanced the accuracy of clustering, that it worked with all available solutions (with some minor modifications), and that it gave better retrieving of keywords (which we will not consider in this paper).

The accuracy should increase in documents, where similar words occur, but when the ordering is different. An example is "Avoid a written door code…" and "… code written to avoid the Trojan back door…", where they would be considered quite similar using unigrams, but not as similar using bigrams or trigrams. In the example above "Trojan back door" is an example of the benefits of trigrams over bigrams.

The proposed algorithm will not give as many benefits in languages where words are compound words, as in Swedish (in contrast to English).

As far as we know, most existing algorithms only use unigrams, although the advantages of n-grams have proven most successful in many language processing areas.

The problem

Clustering is the ability to automatically group similar documents, only using the text in the document itself. To do this one needs to determine the similarity between documents. There is also a need to create reference documents when combining two or more documents.

Clustering is done in many areas of computer science, and there are a lot of different techniques, like algorithms, neural networks, genetic algorithms, et cetera. The different techniques used depend on the application area.

There are many applications for clustering documents. Some examples are: Internet search engines, knowledge management systems or document databases. Similarities between documents could be used for searching the local computer for documents or automatically acquiring meta-data from documents for use in version management systems.

Usually clustering of documents can be combined with several other techniques to enhance the clustering accuracy. These techniques can be: determining grammar, omitting common words or putting weights to the words.

Our solution

We chose to use the k-means clustering algorithm. This algorithm requires a distance value between documents. For this we used one of the most widely

used techniques for determining similarities: the Vector space model.

The reason for the choice of these algorithms is that they are very common and therefore known by most computer linguists. The fact that they are well documented make them easy to implement and easy to read for others. Also, we wanted to focus on the benefits of n-grams compared to unigrams.

We chose to implement the project in Java. In addition to the fact that it is object oriented and that we are familiar with the language, we could also show our progress using an Applet on the Internet. Since this is a commonly used language, others will be able to copy our source code and modify it for their own purposes. Since we had used regular expressions in the course, we thought it would be good to include that functionality in this project as well. This makes the parsing of the documents very flexible. We used Java's standard implementation of regular expressions (included since Java 1.4).

Vector space model

The vector space model is widely used for determining similarity between documents [5].
This method sees each document as a vector in a word-space. The dimension (number of axis) of this word-space is the number of different words in the two documents. The number of occurrences of each word determines the "length" of that axis for that document. Then the vector space model determines the angle (cosine coefficient) or some other value (for example Jaccard or Dice coefficients) [4] between the documents. This represents the similarity between the documents.

When using n-grams instead of unigrams, each axis consists of more than one word. Since we used Java as the programming language, we did some simple inheritance to solve this.

We chose to use the cosine coefficient, which determines the angle between the documents. [1] presented the algorithm for VSM as follows:

$$\cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}},$$

where q is the query, d is the document and n the number of different words.

Since we used only the number of words as the coefficient, only those words, which have a word count higher than zero in both documents, will produce a value. Therefore we simplified this algorithm, so that we would only need to multiply the common words in the nominator. This way the algorithm we use does not require any computations on the words that are not part of both the documents.

$$\cos(\vec{d}1,\vec{d}2) = \frac{\sum_{i=1}^{n_{qd}} d1_i d2_i}{\sqrt{\sum_{i=1}^{n_q} d1_i^2}\sqrt{\sum_{i=1}^{n_d} d2_i^2}},$$

where d1 is document 1, d2 is document 2, and n_{qd} the number of common words.

We extract the words from the documents using regular expressions, which make it flexible to redefine the smallest parts that are analyzed. Then we sort using Java's Mergesort for O(nlogn) performance, instead of putting in the words in a sorted list directly, which would give $O(n^2)$ performance.

Clustering

According to [1], the k-means algorithm "*is the conceptually simplest method and should probably be used first on a new data set because its results are often sufficient*". This summarizes the reasons why we used this algorithm.

The k-means method is really simple; first some cluster centers (center of mass) are randomly chosen (we picked randomly chosen documents). Each document is assigned to one of these clusters (defined by the closest centre). Then a new cluster centre is calculated for each cluster. In the next iteration each document is assigned to one of the new cluster centers that were previously calculated. Currently we iterate a given number of times, instead of having a stop criterion.

The calculation of new cluster centers was done using only the words of all the documents in that cluster. An alternative would have been to include all words in all documents instead, but that would have required more computational power.

Results

We have only tested our algorithm on a limited test, due to time restrictions. We have also constructed examples where our algorithm outperforms traditional (unigram) methods. For example our method can see differences in texts with words that are the same, but that come in a different order. In our test texts there are not many cases where this arises. In some areas there are two words that are belonging to each other, though, which make our algorithm work better.

When calculating a Vector space model value between two documents, a choice has to be made between unigrams, bigrams, trigrams, et cetera. Since we wanted to analyze the benefits of n-grams, we did both separate tests using them individually and a weighed sum of the unigrams, bigrams and trigrams.

We found that the Vector space model between arbitrary documents is only applicable using up to 4-grams or 5-grams, unless you want to spot citations or versions of exactly the same documents. We have chosen to only use the first three values (up to trigrams) in our clustering tests.

When we tested to cluster 6 texts from the New Scientist web page, 3 texts about neutrinos and 3 texts about transistors, they were clustered correctly using both unigrams and our extended version, which included bigrams and trigrams. When analyzing the output of the Vector space model (the similarities), we found that our approach using only bigrams changed the outcome of the clustering somewhat in the wrong direction to what we had expected. Clustering with only trigrams made it cluster properly again, as did the combination of the three. One reason to the failure of bigrams could be the small quantity of texts, or the fact that we did not choose texts that were different enough.

Conclusion

Our limited testing shows that looking at more than one word at a time won't necessarily give accuracy, but can potentially give other benefits when comparing documents. In the few texts we have used, there is no obvious advantage of using our suggested algorithm, although it could be our handpicked documents that are not adequately belonging to other areas. The extra time of implementing the

algorithm probably makes it even less interesting. We hope that people can reuse, and make use of, our source code, though.

Besides clustering, n-grams could be used to spot citations from one document to another, or versions of the same document, with some small changes in the source code. The method could also be used to extract key n-grams, instead of extracting keywords from texts. These would probably describe the document even better than single words.

We think there are a lot of potentially interesting areas where n-grams can be used, and clustering is one of them. We have shown that it has potential, but the current benefits are too small. Maybe a combination of n-grams and other techniques will prove to work best? Further testing of our proposed algorithm needs to be done.

Applet, Java source, and test texts

To make it easier to analyze our results, and to present the code used, a web page was created for the project. It is located at: http://www.efd.lth.se/~d97ksk/language. There we have the source code for the project, together with javadoc, an example Applet, the test text samples we used and this report.

References

[1] Christopher D. Manning and Hinrich Schütze, 1999, *Foundations of statistical natural language processing*, MIT Press.

[2] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, Geoffrey Zweig, 1997, Syntactic Clustering of the Web, *Digital SRC Technical Note 1997-015*, http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1997-015.html

[3] Pierre Nugues, 2002, Intr*oduction to Language Processing and Computational Linguistics*, Lecture Notes, Lund Institute of Technology.

[4] Danmarks Tekniske Universitet, 2002, *Introduction: Vector Space Mod*el, Technical report, http://isp.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html

[5] Gerard Salton, A. Wong, C.S Yang, 1975, A Vector Space Model for Automatic Indexing, *Communications of the ACM*, 18(11), November.

[6] John Zukowski, 2002, Java Tech Tips on using regular expressions in Java, *Sun JDC Technical notes*, http://developer.java.sun.com/developer/JDCTechTips/2002/tt0423.html