USING ENUMERATIONS FOR WORD CLUSTERING

Lorand Dali, Nada Lavrač
Department of Knowledge Technologies
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 1 477 3144;
e-mail: lorand.dali@ijs.si, nada.lavrac@ijs.si

ABSTRACT

This paper presents an application of exploratory data analysis consisting in a method of finding related nouns and clustering them using the information inherent in enumerations occurring in text. The results obtained are presented in a demo, and acces to the functionality is published as RESTful web services.

1 INTRODUCTION

The basic idea on which the rest of this paper relies is that people tend to use related nouns in an enumeration. For instance such enumerations would be: zebra, giraffe, elephant, lion; or train, bus, car, plane; or Germany, France, Spain, Italy. One will rarely (if ever) see an enumeration of totally unrelated nouns like: zebra, helicopter, February, Germany.
The aim of the work presented in the paper is to make use of the information inherent in enumerations. Possible uses could be:

- Given a noun find the nouns which are most related to it
- Determine the extent to which two nouns are related
- Grouping of similar nouns
- Ordering of nouns from general to specific
- Document classification

This paper addresses the problem of finding related nouns and clustering nouns in groups based on relatedness.
The following sections will describe the data, the algorithms used on it, and the results obtained.

2 THE DATA

The data consists of enumerations of nouns as they occur in the Reuters RCV1 news article corpus., which contains over 12 million sentences. 500 000 enumerations of at least 3 nouns were extracted, and in these enumerations occur about 220 000 distinct nouns. The decision to take into account only enumerations with length at least 3 came from the wish to keep the data to a manageable size. And also I think that the longer an enumeration is, the more related the terms in it are. More data about the data is shown inTable 1. In the left half of the table is shown how many sentences contain a certain number of enumerations (of at least 3 nouns).

Enumerations per sentence		Enumeration lengths	
0	12 119 657	2	3 270 921
1	460 587	3	380 012
2	16 550	4	77 070
3	1 098	5	22 530
4	749	6	8 482
5	20	7	4 612
6	12	8	3 004
7	3	9	1 075
8	3	10	938

Table 1 Statistics about the data

It can be seen that most (almost all) of the sentences do not contain any suitable enumerations at all. The right half of the table shows how many enumerations are of a certain length. It becomes clear that the vast majority of the enumerations are of length 2, of which none were taken into account for the experiments done. This means that I have used only about 10% of the available data.

2.1 Data Preprocessing

The question which is answered here is how to obtain, from a passage of text which contains an enumeration, a list of nouns appearing in the enumeration. In order to achieve this two preprocessing steps were done: part of speech tagging, and noun phrase chunking. For example the text '*milk and toast and honey*' contains an enumeration which we would like to extract. The part of speech tagger will transform this text into *<noun><conj><noun><conj><noun>*, and the nounphrase chunker to *<NP><conj><NP><conj><NP>*. Now it is easy to recognise sequences of noun phrases separated by conjunctions or commas to obtain the list of nouns i.e. [milk, toast, honey]
Other preprocessing steps are converting all words to lower case, and trying to bring nouns to their base form using WordNet[1].

For the part of speech tagging CRFTagger[2], and for the nounphrase chunking CRFChunker[3] have been used.

2.2 Graph Representation

The enumeration data is stored as an undirected graph where the nouns are the nodes. Two nodes are connected with an edge if they were enumerated together at leas once. The weight of the edge is how many times the two nouns have been enumerated together. Figure 1 shows an example of graph which was obtained from the following enumerations:

[*brandy, whiskey, rum, gin*]
[*chips, coke*]
[*whiskey, pizza, beer, chips*]
[*cheese, beer, pizza*]
[*chips, beer, coke, popcorn*]

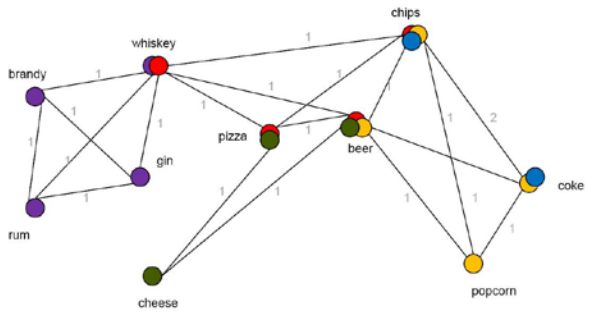

Figure 1 Graph representation

3 ALGORITHMS USED

3.1 Random Walk

Random walk is a process in which at each step we are at a node in a graph, and we randomly choose another node from the graph to be in at the next step. The node chosen for the next step has to be a neighbor of the node we are in at the current step, and the probability of it being chosen is determined by the weight of the edge which connects it. At each step there is also a probability of stopping i.e. ending the process of random walk.

Using random walk we can compute for each node the probability of ending in any other node if starting from there. This probability tells us how related, or how associated, a word is to another. If from a certain word there is a high probability of reaching another given word after a random walk, then this means that the first word is highly related to the second one. It has to be noted that in general it is not true that if a word is strongly related to another then the other one will also be strongly related to the first one. So for each word pair the relation is described by two numbers, one for each direction.

For the graph described in this paper random walk has been computed using Monte Carlo sampling. From each node a large number of walks (2000 in this case) have been started, and the probabilities were estimated using the relative frequencies of the obtained destinations.

3.2 Clustering

Clustering is a way of grouping similar entities together. The similarity between two entities is expressed by a distance. The smaller the distance, the more similar the two entities are. In clustering each entity belongs to exactly one cluster, and the goal is that members of the same cluster have a small distance to eachother while the clusters are far from eachother.

For being being able to compute distances, the entities have to be given as data points (vectors), or alternatively a distance matrix, showing the distance from each entity to each entity, can be directly given.

In the clustering application presented here the entities to be clustered are nouns (the nodes in the graph). The distance between two nouns is computed as the average between the two associations between the words. The goal is not to cluster the entire set of words, but rather a given subset. The clustering method of choice was the hierarchical aglommerative clustering which has as output a tree with the leaves the elements which must be clustered. It works like such that at the beginning each entity is a cluster of its own and at each step the two clusters which are nearest to eachother are merged, until only one cluster remains. The inter-cluster distance has been computed as the average of distances between the distances of all pairs of elements where one element is from one cluster and one is from the other.

For example if we cluster the words *apple*, *train*, *doctor*, *helicopter*, *lawyer* and *orange* we obtain the tree shown in Figure 2

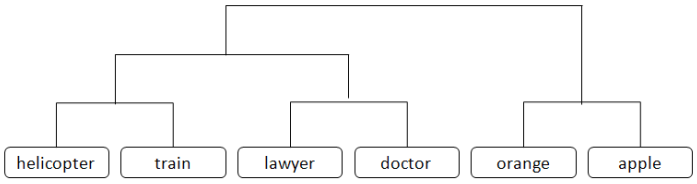

Figure 2 Hierarchical clustering

It can be seen that fruit (apple and orange) professions (lawyer and doctor) and vehicles (helicopter and train) are clustered together. Moreover from the resulting three clusters the one with vehicles and professions are closest to eachother.

The clustering has been done using the C Clustering Library[4] through a Python interface called Pycluster[5].

4 DEMO

The results of the work are presented in a demo in which the user can do three things:

He can explore for any given word what words it has been enumerated with, and to what words it is mostly relayed.
He can check the relations between any pair of words.
He can give a set of words to be clustered hierarchically like shown in Figure 2.
The functionality used in the demo is also published as RESTful web services.

5 CONCLUSIONS AND FUTURE WORK

Time has come to draw the conclusions. A method of finding words related to a given word and to group related words together has been tried. Although no evaluation has been done we can intuitively conclude that the method works quite well, and that indeed there is much information in enumerations.
A point which is worth discussing is the choice of algorithm used to compute relatedness. The random walk probabilities estimation through random sampling needs very little memory, no matter how big the data. The runtime is long if a large number of samples is taken, but it turns out that just 2000 samples give quite satisfactory results. Moreover the algorithm presented is very suitable for parallel computing as it can be easily divided into independent jobs (one for each node) which can be executed in any order. Also the method is scalable as it allows easy addition of new nodes after all probabilities have been estimated. The other alternatives would be to compute the random walk probabilities with matrix multiplications or with singular value decomposition. With both of these methods the difficulties caused by the limited memory must be surpassed. A parallel implementation is not as straightforward, and also online addition of new nodes is more tricky in these cases.
As future work document classification based on enumeration data could be tried. I expect that such a classifier would be more accurate, or at least would need less training data.
Also, using the asymmetry of the relatedness measure computed by the random walk, an ordering of words from general to particular could be tried, as specific words tend to be more strongly associated with general words than the other way around.

ACKNOWLWDGEMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC under NeOn (IST-4-027595-IP) and ACTIVE (IST-2008-215040).

References

[1] WordNet: An Electronic Lexical Database
[2] Xuan-Hieu Phan, "CRFTagger: CRF English POS Tagger", http://crftagger.sourceforge.net/, 2006.
[3] Xuan-Hieu Phan, "CRFChunker: CRF English Phrase Chunker", http://crfchunker.sourceforge.net/, 2006.
[4] The C Clustering Library, The University of Tokyo, Institute of Medical Science, Human Genome Center
[5] Pycluster, http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm.