Extracting semantic clusters from the alignment of definitions

Gerardo SIERRA
Instituto de Ingeniería, UNAM
Apdo. Postal 70-472
México 04510, D.F.
gsm@pumas.iingen.unam.mx

John McNAUGHT
Department of Language Engineering, UMIST
PO Box 88
Manchester M60 1QD, UK
John.McNaught@umist.ac.uk

Abstract

Through the alignment of definitions from two or more different sources, it is possible to retrieve pairs of words that can be used indistinguishably in the same sentence without changing the meaning of the concept. As lexicographic work exploits common defining schemes, such as genus and differentia, a concept is similarly defined by different dictionaries. The difference in words used between two lexicographic sources lets us extend the lexical knowledge base, so that clustering is available through merging two or more dictionaries into a single database and then using an appropriate alignment technique. Since alignment starts from the same entry of two dictionaries, clustering is faster than any other technique.

The algorithm introduced here is analogy-based, and starts from calculating the Levenshtein distance, which is a variation of the edit distance, and allows us to align the definitions. As a measure of similarity, the concept of longest collocation couple is introduced, which is the basis of clustering similar words. The process iterates, replacing similar pairs of words in the definitions until no new clusters are found.

Introduction

Clustering methods to identify semantically similar words are usually divided in relation-based and distribution-based approaches [Hirawaka, Xu and Haase 1996]. Relation-based clustering methods rely on the relations in a semantic network or ontology to judge the similarity between two concepts, either by measuring the shortest length that connects two concepts in the hierarchical net [Agirre and Rigau 1996], or by comparing the information content shared by the members under the same cluster [Morris and Hirst 1991, Resnik 1997]. However, even although these ontologies describe a huge number of members for a cluster, few words of a category may be interchangeable in the same context and then used as members of the same cluster. This means that not all words in a category are necessary.

Conversely, distribution-based clustering methods depend on pure statistical analysis of the lexical occurrences in running texts. A major drawback is that distribution-based methods require us to process a large amount of data in order to get more reliable results. Moreover, the use of large corpora is not always practical, due to economic, time or capabilities factors. Gao [1997] states that the problem for statistical alignment algorithms, such as those based on the facts described by Gale and Church [1991], is the low frequency of words that occur in parallel corpora. The consequences for lacking large corpora include results based on low-frequency words, which are quite unrepresentative for clustering.

From a methodological point of view, there is, in addition to the above two approaches, a little known approach called the analogy-based approach. This employs an inferential process and is used in computational linguistics and artificial intelligence as an alternative to current rule-based linguistic models.

1 Analogy-based clustering

Jones [1996] suggests corpus alignment as a feasible analogy-based approach. In order to align two sentences in the same language, Waterman [1996] uses a technique for measuring the similarity between lexical strings, named *edit distance*. This matches the words of two sentences in linear order and determines their correspondence. For example, given the following two definitions for alkalimeter:

- An apparatus for determining the concentration of alkalis in solution [CED]
- An instrument for ascertaining the amount of alkali in a solution [OED2]

Alignment may identify which words in these definitions are equivalents of each other. A quick observation of the sentences lets us identify three pairs of words: (apparatus, instrument), (determining, ascertaining) and (concentration, amount).

The appeal of using definitions as corpora for alignment is founded on two reasons. Firstly, dictionaries contain all necessary information as a knowledge base for extracting keywords [Boguraev and Pustejovsky 1996]. Secondly, it is much easier to find the sentences for aligning, since definitions are distinguished by entry headword.

Taking into account Waterman´s studies, we propose an analogy-based method to identify automatically semantic clusters. The difference in words used between two or more lexicographic definitions enables us to infer paradigms by merging the dictionary definitions into a single database and then using our own alignment technique.

2 Clustering algorithm

The overall structure of the clustering algorithm is shown in figure 1, and its description is given below.

2.1 Processing definitions

Our algorithms are used in an overall system called "onomasiological search system" (OSS), whose aim is to allow the user to find terms by giving a description of a concept. Lexicographic and terminological definitions constitute the main lexical resources. Our algorithms cluster words that are used in the same context, thus operate on pairs of definitions for a same entry word, drawn from two different dictionaries. If dictionary *I* does not have an entry word that exists in dictionary *J*, then this entry word is omitted from consideration. In order to balance the number of strings when an entry word in the dictionary *I* has two or more senses, the entry word in dictionary *J* is repeated as many times as necessary to equal the number of senses of dictionary *I*. We thus derive two files *I* and *J* containing an equal number of strings S_1 and S_2, respectively. Each string consists of an entry term followed by its definition, the definition giving only one sense of the entry term. For each string S_1 there is a string S_2.

Our experiments focus on 314 terms for measuring instruments extracted with their definitions from CED [1994] and OED2 [1994], resulting in 387 strings from each dictionary.

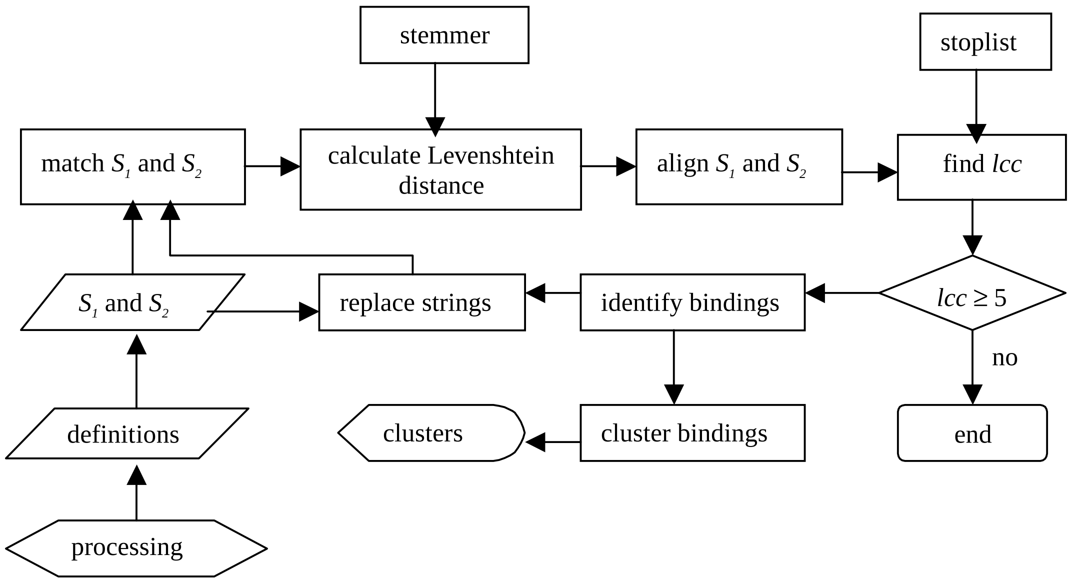

Figure 1 Clustering algorithm

The strings consist of the entry term and the definition, so that etymology, part of speech, inflected forms of the entry term, examples and other information were deleted. Subject-field labels, such as ‘astronomy’ and ‘meteorology’, were preserved, either in full or slightly abbreviated, as they are helpful to resolve which sense of a word to choose, and usually constitute a fundamental property of the concept.
It should be noted that none of the 387 strings suffered any additional transformation, apart from a few cases in order to complete a definition when it had been broken in two parts by the dictionary editor, such as when a core meaning appears just once at the beginning of several subsequent senses. Although some abbreviations (‘U.S.A.’), initials of proper names (‘C.T.R. Wilson’) and possessives (‘sun's rays’) will come out as two or more words after deleting punctuation marks and therefore can alter the efficiency of the algorithm, they were preserved to observe their effect.

2.2 Aligning definitions

In order to compare two strings of words, we use the Levenshtein distance [Levenshtein 1966], a similar method to the edit distance. This method measures the edit transformations that change one string into other. The Levenshtein distance arranges the strings in a matrix, with the words of S_1 heading the columns and those of S_2 heading the rows. A null word is inserted at the beginning of each string S_1 and S_2, in position i=0, j=0. The matrix is filled with the costs of insertion, deletion and substitution using the following formula :

$$\mathrm{D}(\mathrm{a}_i, \mathrm{b}_j) = \min \begin{cases} D(a_i, b_{j-1}) + D_{ins}(b_j) \\ D(a_{i-1}, b_j) + D_{ins}(a_i) \\ D(a_{i-1}, b_{j-1}) + D_{sub}(a_i, b_j) \end{cases}$$

Where the cost of insertion, $D_{ins}()$, is 1, and the cost of substitution, $D_{sub}()$, is 0 or 1, according to whether a_i and b_j differ or not.
Our experimental results have shown that the application of the Levenshtein distance using stem forms gives better matches than using full forms. Therefore, we shall fill the matrix with the cost for the stem forms, although the strings preserve the full forms both for the following steps and in the output table. We used the stemming algorithm of Porter [1980], which removes endings from words.
Building on the Levenshtein distance, Wagner and Fisher [1974] propose a dynamic programming method to align the elements of two strings. Their procedure to return the ordered pairs of the alignment starts with the last cell of the matrix with cost[n][m] and works back until either i or j equals 0, according to which of its neighbours a cell was derived from. If it is derived either from the previous horizontal or vertical cell ([i-1][j] or [i][j-1] respectively) then the difference in cost is just 1, otherwise it is derived from the diagonal.

2.3 Extracting triplets

The alignment gives us a list of triplets formed by (ff_i, ff_j, cost[i][j]), in decreasing order according to cost[i][j], where ff_i, and ff_j are full forms from the strings S_1 and S_2, respectively. There are three possible pairings of words:
“Equal couple” is defined as the pair (ff_i, ff_j) of full forms such that the corresponding stem forms are equal ($sf_i = sf_j$).
“Matched couple” is a pair (ff_i, ff_j) such that $sf_i \neq sf_j$. This couple represents a potential pair of similar words.
“Null couple” is a pair (ff_i, ff_j) such that sf_i or sf_j is missing.
With respect to the Levenshtein distance, the equal couple means these words do not need any change to make both equal, while for the matched couple we shall replace one word with the other progressively, and for the null couple we must either insert one word into the given string or delete it from the given string.
The purpose of clustering is to match different pairs of words (matched couples), thus neither pairs of equal words (equal couples) nor pairs with a null word (null couples) are relevant.

2.4 Measuring similarity

As a measure of the similarity between a matched couple, we quantify the surrounding equal couples above and below it. This concept is similar to the “longest common subsequence” of two strings suggested by Wagner and Fisher [1974], which is defined as the common subsequence of two strings having maximal length, although in our case both strings differ by the single matched couple. By analogy, we use *longest collocation couple*, henceforth abbreviated *lcc*, since we refer to couples instead of a single string. Besides, the word "collocation" is more representative for a pair of words and their neighbourhood, being the core of two longest common subsequences. We define longest collocation couple as the maximal sequence of pairs of words formed by equal couples surrounding a matched couple.

Given the alignment of the strings S_1 and S_2 consisting of a list of triplets formed by (ff_i, ff_j, cost[*i*][*j*]), in decreasing order according to cost[*i*][*j*], where ff_i, and ff_j are, respectively, full forms from S_1 and S_2, the *lcc* is the longest consecutive sequence of triplets (ff_i, ff_j, cost[*i*][*j*]) formed by one matched couple, such that it meets 3 conditions:

- The cost difference between the first triplet and the last triplet is 1.
- There is no null couple.
- The matched couple is neither the first nor the last triplet.

By these conditions, only the matched couple becomes the core of a *lcc*: we constrain a matched couple to be between two or more equal couples, and eliminate the possibility that the matched couple appears at the beginning or end of a phrase.

As a result, we get a new triplet (ff_i, ff_j, lcc_{ij}), where (ff_i, ff_j) is the matched couple and lcc_{ij} is the length of the longest collocation couple. As an example, for the definitions of "dynameter" in table 1, there is only one matched couple, "determining-measuring", whose *lcc* is 9 (the extent of the *lcc* is indicated by arrows).

ff_i	***ff_j***	***cost[i][j]***	
telescopes	telescope	2	
--	a	2	
of	of	1	←
power	power	1	
magnifying	magnifying	1	
the	the	1	
determining	**measuring**	1	lcc = 9
for	for	0	
instrument	instrument	0	
an	An	0	
dynameter	dynameter	0	←

Table 1 Triplets for "dynameter"

Ranking all triplets found by *lcc* in decreasing order, we observe that the greater the value of *lcc*, the greater the similarity between the words of the matched couple.

2.5 Removing function words

So far, function words and other noise words will also be clustered by our algorithms. In general, such words interfere in the identification of clusters and can give more wrong than good results. We use a stoplist to automatically identify any pair of words where a non-relevant word appears and exclude it, on the grounds that they are not very useful words for clustering. Thus, when the program comes across a matched pair of different words in a context and if that matched pair contains a word from the stoplist, then the pair is rejected. Essentially, this is the same thing as using a tagger and looking at the tags as well as the words, since one would not want to choose a noun pairing with a determiner or a relative.

By inspection, we observe that, after stoplist discrimination, the best potential clusters are found at higher values of *lcc*. Our experimental results show us that a length of *lcc* equal to 5 is a reliable threshold. Although there are also good matches for values equal to 4 and 3, the majority of these are duplicates of higher values.

2.6 Clustering

We introduce the term *binding* to represent a candidate cluster, i.e. two words that may be used in the same context without changing the meaning of a definition. A binding is a matched couple (ff_1, ff_2) formed by the full forms ff_1 and ff_2, after stoplist discrimination, drawn from the strings S_1 and S_2, respectively, in such a way that the stem forms are equivalent, in a determined context, according to a determined threshold. The threshold associated with a binding is the length of the *lcc*, and we consider only bindings of matched couples where $lcc \geq 5$.

Each binding can be considered as an initial cluster. Clusters represent sets of words that are used with the same meaning in particular contexts. In a consecutive sequence of bindings, it may happen that a stem form occurs in two or more different bindings. In this case, one can cluster all bindings with a common stem form according to the transitive property.

In order to cluster bindings, we use an algorithm consisting of three loops. First, it assigns a cluster number to each binding, so those bindings with a common word have the same cluster number. Secondly, it clusters bindings with the same cluster number, but removes duplicate stem forms in the same cluster. Thirdly, it checks if it is possible to merge new clusters with those of previous cycles. This process will typically result in a set of overlapping clusters, reflecting the natural state where concepts may belong to more than one conceptual class.

2.7 Cycling

As bindings represent pairs of words such that the stem forms can be substituted in a particular context without changing the meaning, $sf_1 = sf_2$, we can replace any of the full forms $\boldsymbol{ff}_i$ with the full forms $\boldsymbol{ff}_j$ according to each binding, so that the corresponding definition preserves the same meaning. After substituting bindings, we observe that several pairs of words will now typically present a high *lcc* score, even those pairs of words which initially did not yield matches with any word. It is then advantageous to replace thus the bindings in the definitions and to repeat the entire process until no new clusters are found. The first cycle runs from the reading of definitions up to merging of clusters. All subsequent cycles will start by replacing retained bindings in the definitions, thus each subsequent cycle works with new data.

3 Experimental results

The current clustering algorithm was developed by analysing definitions on the following basis:

- Language dictionaries. The use of language dictionaries has been preferred because there are enough to extract data from. As they are in machine-readable form, it is possible to copy definitions, avoiding likely mistakes while typewriting.
- Corpus on 314 "measuring instruments". This domain has the advantage that it is easy to search for the terms that correspond to it, as they usually end in "-meter", "-scope" or "-graph". As a consequence of applying the clustering program to the 387 strings, it is evident that the majority of clusters were related to "measure" and "instrument".
- Alignment of two strings. We have shown that two sources of data (pairs of definition) are sufficient for clustering to yield good results.
- No manipulation of data. After identification of the term and the definitions, these were truncated to 200 characters and punctuation marks were removed. No words in definitions were replaced or moved, to "tidy up" the data, before being submitted to the main process.
- Stemming algorithm. The stemmer algorithm presents both overstemming and understemming, but nevertheless the clustering program yields good results.
- Stoplist discrimination. The stoplist has been used as a tagger, i.e. as a filter to avoid matching words with different parts of speech.
- Bindings for $lcc \geq 5$. The best clusters have been observed for bindings with $lcc \geq 5$, and the results presented are good.

Table 2 presents some cluster results after two cycles of the clustering procedure starting from the Levenshtein distance. In addition to these clusters, 14 other clusters of two or three elements were obtained.

1.	apparatus instrument telescope
2.	analyse ascertaining determining estimating location measuring recording takins testing
3.	amount concentration intensity percentage proportion rate salinity strength

Table 2 Cluster results for "measuring instruments"

The procedure then stops, as no more matched words with $lcc \geq 5$ have been found for our data. The following sections analyse variations of these considerations.

3.1 Using multiple resources

General language dictionaries present the advantage of using well-established lexicographic criteria to normalise definitions. These criteria, as for example the use of analytical definitions by genus and differentia, have been nowadays implemented by terminological or specialised dictionaries, with the addition of a richer vocabulary and the identification of properties that are not always considered relevant in other resources. Unfortunately, these are more oriented to a specific domain, so that it is sometimes necessary to search in two or more resources to compile the data.

We used many online lexical resources, some of them available on the Internet. This allowed us to easily use different databases to extract

semantic clusters. As an example, for the term "barometer" we selected from the Internet 17 sources from general language dictionaries, terminologies and specialised dictionaries, in addition to OED2 and CED.
Table 3 demonstrates the use of our clustering program for the 19 definitions of "barometer", considering a *lcc* ≥ 5, stoplist discrimination, and a previous modification of the original strings as indicated in the paragraph below.

1.	air atmospheric
2.	device instrument
3.	determining measures shows

Table 3 Clusters for "barometer"

From this table, we see there are only 3 clusters, but comparing these with the strings we can observe that these clusters are complete with low recall and high precision. No more clusters can be extracted from the strings, there are no more relevant words in the strings that still can be clustered, and there are no unnecessary words in any of these clusters.

3.2 Modifications of the strings

In order not to manipulate the strings to retrieve biased clusters, definitions were not modified beyond the pre-processing described. In fact, entry words were chosen randomly, but always in the domain of measuring instruments. Although good precision is observable in the clusters, there are still some relevant words in the strings that are semantically similar to some of those of the clusters. For example, the word 'device' is frequently used instead of 'instrument', but because of the definition of *lcc*, the matched couple (device instrument) rarely can be a binding for clustering, as the preceding determiner of each word is different. The former use 'a', while the latter use 'an' and unfortunately the stemmer did not stem 'an' to 'a', thus (an a) do not form an equal couple.
However, before stoplist discrimination was introduced, the matched couples (any an) and (any a) present a *lcc* ≥ 5, so that by our clustering algorithm they should belong to the same cluster and then one can replace one with the other in the strings. By running the program without stoplist discrimination, one can observe two clusters related to function words:

Cluster 1: a an any the

Cluster 2: for that which

According to these premises, table 4 shows clusters by first replacing all the strings according to these clusters of function words. The italicised words are the new words added to the list for the clustering algorithm presented in table 2, where most of the words added are correctly used as equivalent words.

1.	apparatus *device* instrument *meter* telescope
2.	analyse ascertaining *astronomical counting detecting* determining estimating *indicates* location *making* measuring *provides* recording takins testing

Table 4 Clusters after replacing clusters of function words

4 Conclusion

We presented an innovative clustering algorithm expressly created to identify automatically overlapping and non-hierarchic clusters. This is an analogy-based method, as one can acquire knowledge of an unfamiliar linguistic object by extracting the right amount of linguistic knowledge from examples of similar objects. The difference in words used between two or more lexicographic definitions enabled us to infer paradigms by merging the dictionary definitions into a single database and then using our own alignment technique.
An advantage of our clustering method over other statistical or analogy-based methods is that it is not over-dependent on the availability or amount of data from which clusters are extracted or on the use of an ontology. Alignment algorithms based on Levenshtein distance are not statistical by nature, so that they do not require large amounts of data and can return clusters even when alignment between words is very rare. The major advantage, however, is that our method complies with the need to identify pairs of words that can be replaced one for another without affecting the meaning of a concept. It achieves this by aligning definitions that express the same concept with different words. The final clusters were evaluated by our overall system, OSS, via hypothetical queries with a paradigm expansion based on the clusters. No further evaluation was available beyond direct observation, since the accuracy of semantic clusters from other lexical resources is

arguable, and we would have to evaluate then first and then define which of them is the best as a point of comparison. Although our technique is intended for use within OSS, the results yielded are of relevance to those interested in sense disambiguation, in classification and in other areas where clusters of similar words are exploited.

The clustering algorithm here proposed gives us reliable clusters using a stemmer algorithm, stoplist discrimination, $lcc \geq 5$ and no manipulation of the strings. A better performance of the program would be achieved by using equivalence of function words, and a tagger for part of speech recognition. The former was demonstrated and lets us retrieve words that usually are not matched as they do not have a high *lcc* value. The latter lets us exclude matching words with different categories. This however requires further research.

We can think of some other further manipulations the strings can undergo to improve the retrieval of similar words. For example, reducing to a single word form two or more abbreviations of a proper name (‘T.S. Eliot’) or of an acronym (‘U.S.A.’). A major manipulation of strings that undoubtedly can improve the retrieval of clusters is trying to normalise the syntactic elements of the strings. Therefore, possessives can be transformed to noun phrases. For example, ‘direction of the wind’ can be replaced by ‘wind direction’ or ‘carpenter’s square ‘ to ‘carpenter square’. Similarly, as suggested by Waterman [1996], one can try to align the same part of speech categories after using a tagger, so that bindings of different categories are rejected.

References

Agirre, E., and Rigau, G. 1996. “Word Sense Disambiguation using Conceptual Density”. *Proc. COLING-96. The 16th International Conference on Computational Linguistics*, Copenhagen, 16-22.

Boguraev, V. and Pustejovsky, J. 1996. “Issues in text-based lexicon acquisition”. *Corpus processing for lexical acquisition*. B. Boguraev and J. Pustejovsky (eds.). Cambridge: The MIT Press.

[CED2] 1994. *Collins English dictionary*. Glasgow: Harper Collins Publishers.

Gale, W.A. and Church, K.W. 1991. “A program for aligning sentences in bilingual corpora”. *Proc. of 29th Annual Conference of the ACL*, 177-184.

Gao, Z.M. 1997. *Automatic extraction of translation equivalents from a parallel Chinese-English corpus*. PhD Thesis, UMIST.

Hirakawa, H., Xu, Z., and Haase, K. 1996. “Inherited Feature-based Similarity Measure Based on Large Semantic Hierarchy and Large Text Corpus”. *Proc. COLING-96. The 16th International Conference on Computational Linguistics*. Copenhagen: Center for Sprogteknologi.

Jones, D. 1996. *Analogical Natural Language Processing*. London: UCL Press.

Levenshtein, V.I. 1966. “Binary codes capable of correcting deletions, insertions, and reversals”. *Cybernetics and Control Theory* 10 (8), 707-710.

Morris, J., and Hirst, G. 1991. “Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text”. *Computational Linguistics* 17(1), 21-48.

[OED2] 1994. *Oxford English dictionary*. Oxford: Oxford University Press and Rotterdam: Software B.V.

Porter, M.F. 1980. “An algorithm for suffix stripping”. *Program* 14(3), 130-137.

Resnik, P. 1997. “Disambiguating noun groupings with respect to WordNet senses”. *Proceedings of the 3rd Workshop on Very Large Corpora*, MIT.

Wagner, R.A., and Fisher, M.J. 1974. “The String-to-String Correction Problem”. *Journal of the Association for Computing Machinery* 21(1), 168-173.

Waterman, S.A. 1996. "Distinguished Usage". In *Corpus Processing for Lexical Acquisition*. B. Boguraev and J. Pustejovsky (eds.) Cambridge: The MIT Press.