Analogy-based Method for Semantic Clustering

Gerardo Sierra,
John McNaught

An analogy-based clustering method is proposed, through the alignment of definitions from two different sources. The method relies on the assumption that two authors use different words to express a definition. 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.

1 INTRODUCTION

The primary goal of clustering is to collect together into clusters a set of elements associated by some common characteristic. Each element or member within a cluster *A* is strongly associated with each other because they share the same property, while members of other clusters show distinct characteristics from those of *A*. According to Gordon [1981], clustering may alternatively be oriented either to discover the strongest association among members or to seek members which are isolated from each other. Clustering is often based on measurements of the similarity or dissimilarity between a pair of objects, these objects being either single members or other clusters.

Clustering methods to identify semantically similar words are usually divided in relation-based and distribution-based approaches [Hirawaka, Xu and Haase 1996]. The former analyse relations in an ontology, while the latter use statistical analysis. According to the terminology of Grefenstette [1996], these methods can be called knowledge–rich, based on a conceptual dependency representation, and knowledge-poor, based on distributional analysis.

Relation-based clustering methods rely on the relations in a semantic network or ontology to judge the similarity between two concepts. Since an ontology connects concepts, each located in a node, it is then possible to analyse either the taxonomic relations or just the conceptual distance between the nodes. A taxonomy lets us extract semantic relations, such as is-a or a-kind-of

[Chakravarthy 1994], to judge the similarity between two concepts by comparing their parent. A semantic network lets us derive similarity by determining the path-length or number of links between the nodes.

The ontologies WordNet [Miller et al 1990] and Roget's thesaurus [Roget 1987] have been used to cluster similar 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 to the semantic relations extracted from an ontology, distribution-based clustering methods depend on pure statistical analysis of the lexical occurrences in running texts. The basis for the statistical approach is that similarity of words can be judged by analysing the similarity of the surrounding context in which they occur, since it has been observed that two synonym words share similar context when they occur separately.

The use of statistical techniques to find similar words faces difficulties when it is fully automated, and new methods attempt insofar as possible to solve these difficulties. Earlier studies encountered drawbacks with the treatment of independent variant forms, such as spelling variation and inflectional endings of words [Adamson and Boreham 1974]. Although most corpus analysis software allows us to analyse variations of a word in the same utterance, it requires additional effort that reduces the efficiency of the method.

A major drawback is that distribution-based methods require us to process a large amount of data in order to get more reliable results [Habert et al 1996; Arranz 1997]. Moreover, the use of large corpora is not always practical, due to economic, time or capabilities factors. The consequences for lacking large corpora include results based on low-frequency words, which are quite unrepresentative for clustering. Grefenstette [1996] suggests that a mixture of different methods, rather than any single statistical technique, may be adapted to be usefully applied to all ranges of frequency of words encountered in a corpus: for more frequent words, finer grained context discrimination; for less frequent words, using windows of *N* words; for rare words, examining large windows, even to entire document level.

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.

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. However, syntactic differences affect the alignment when using purely edit distance. Therefore, Waterman suggests further extensions to the basic edit distance, such as using a part-of-speech tagger and then splitting the original sentences into more similar sub-sentences.

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 enabled us to infer paradigms by merging the dictionary definitions into a single database and then using our own alignment technique.

2 ALGORITHMS

Our algorithms are used in an overall system 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.

In order to compare two strings of words, we use the Levenshtein distance [Levenshtein 1966], a similar method to the edit distance. Both methods measure the edit transformations that change one string into other. The Levenshtein distance also 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 same formula as the edit distance, but differing in that the cost of substitution is 1, instead of 2 as proposed by the edit distance.

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. Given the strings S_1 and S_2 consisting of full forms, with $|S_1| = n$, $|S_2| = m$ the number of words, and $sf_1[i]$, $sf_2[j]$ referring to the i^{th} and j^{th} stemmed forms of S_1 and S_2, respectively, the algorithm to calculate the cost, *cost[i][j]*, is as follows:

```
for i:=0 to n do cost[i][0] := i;
for j:=0 to m do cost[0][j] := j;

for i:=1 to n do
    for j:=1 to m do begin
    /* if the stem forms are equal */
    if ($sf_1$ [i] = $sf_2$ [j]) then
        cost[i][j] := min(cost[i-1][j] + 1, cost[i][j-1] + 1, cost[i-1][j-1])
    else
        cost[i][j] := min(cost[i-1][j] + 1, cost[i][j-1] + 1, cost[i-1][j-1] + 1)
    end
```

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. Table 1 shows the situation after construction of the matrix for the pair of strings S_1 and S_2; the highlighted cells show the best alignment according to Wagner & Fisher's algorithm.

S_1*: alkalimeter an apparatus for determining the concentration of alkalis in a solution.*
S_2*: alkalimeter an instrument for ascertaining the amount of alkali in a solution.*

Table 1. Levenshtein distance for "alkalimeter"

	--	**alkalimeter**	**an**	**apparatus**	**for**	**determining**	**the**	**concentration**	**of**	**alkalis**	**in**	**solution**
--	**0**	1	2	3	4	5	6	7	8	9	10	11
alkalimeter	1	**0**	1	2	3	4	5	6	7	8	9	10
an	2	1	**0**	1	2	3	4	5	6	7	8	9
instrument	3	2	1	**1**	2	3	4	5	6	7	8	9
for	4	3	2	2	**1**	2	3	4	5	6	7	8
ascertaining	5	4	3	3	2	**2**	3	4	5	6	7	8
the	6	5	4	4	3	3	**2**	3	4	5	6	7
amount	7	6	5	5	4	4	3	**3**	4	5	6	7
of	8	7	6	6	5	5	4	4	**3**	4	5	6
alkali	9	8	7	7	6	6	5	5	4	**3**	4	5
in	10	9	8	8	7	7	6	6	5	4	**3**	4
a	11	10	9	9	8	8	7	7	6	5	4	4
solution	12	11	10	10	9	9	8	8	7	6	5	**4**

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 ($sf_i = sf_j$). With respect to the Levenshtein distance, this means these words do not need any change to make both equal, as their stems are equal.
- **"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. In terms of the Levenshtein distance, we shall replace one word with the other progressively to change one string into the other.
- **"Null couple"** is a pair (ff_i, ff_j) such that sf_i or sf_j is missing. This means we must either insert one word into the given string or delete it from the given string, to change one string into the other.

Table 2 shows the list of triplets for strings S_1 and S_2 derived from the matrix of table 1. It also shows the equal, matched and null couples that have been found.

Table 2. List of triplets for "alakalimeter"

ff_i	ff_j	**cost[i][j]**	**kind of couple**
solution	solution	4	equal couple
--	a	4	null couple
in	in	3	equal couple
alkalis	alkali	3	equal couple
of	of	3	equal couple
concentration	amount	3	matched couple
the	the	2	equal couple
determining	ascertaining	2	matched couple
for	for	1	equal couple
apparatus	instrument	1	matched couple
an	An	0	equal couple
alkalimeter	alkalimeter	0	equal couple

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.

As a measure of the similarity between a matched couple, one can 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:

1. The cost difference between the first triplet and the last triplet is 1. As the *lcc* contains only one matched couple, by this condition the matched couple becomes the core of a longest common subsequence. This implies having at least two triplets, where the cost of the matched couple is a unity more than the couple immediately below.
2. There is no null couple. The pair of words immediately above a null couple is, when it exists, an equal couple, the difference of cost between the two pairs being equal to 0. However, it is used as a restriction since it is equivalent to a matched couple by the fact that the cost difference regarding the couple immediately below is 1. Therefore, by definition, a null couple can never be the core of a *lcc*.
3. The matched couple is neither the first nor the last triplet. By this condition, 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. More than one *lcc* may be found for any pair of strings. 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.

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. Therefore, 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*. 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.

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 $lcc \geq 5$ (see table 6).

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.

Merged clusters are represented as a list of binary trees. Column 1 of table 3 shows the merged clusters found after cycle i, column 2 shows the clusters found during cycle i+1, and column 3 shows the new merged clusters formed by merging those of column 2 with those of column 1, at the end of cycle i+1. The lists of new clusters found in a cycle are processed one at a time. Each member of each such list is taken and its stem form is searched in the list of binary trees. Whenever a match is found in a tree (only one match is necessary) all other words in that list are added to this tree, omitting any that are already present. Use of binary trees not only cuts down on search time but also allows clusters to be easily printed out in alphabetical order when required. 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.

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 ff_i with the full forms 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.

Table 3. Merging clusters

Merged clusters after cycle i	New clusters at cycle i+1	Merged clusters after cycle i+1
1: direction inclination	7: sunlight day	1: field limits
2: instrument telescope	8: method system	2: method system
3: swinging turning	9: limits field	3: day sunlight
4: amount intensity rate strength	10: measuring taking ascertaining testing determining	4: direction inclination
5: celestial heavenly	11: amount percentage	5: instrument telescope
6: determining measuring		6: swinging turning
		7: amount intensity percentage rate strength
		8: celestial heavenly
		9: ascertaining determining measuring taking testing

3 EXPERIMENT RESULTS

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. 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.

We used the stemming algorithm of Porter [1980], which removes endings from words. This algorithm, widely used in IR, was chosen because it performs slightly better than other similar algorithms. It should be noted that the Porter algorithm causes some overstemming and understemming. The risk of overstemming is low, since there is a low probability of having two different full forms in the same definition with the same stem form. Understemming is more probable and can cause some words not to match, but the proposed clustering procedure will eventually match them, due to cyclic replacement.

Table 4 presents the bindings for the corpus on measuring instruments after removing both stop words and matched couples with *lcc* < 5. For simplicity here, we have also deleted duplicate bindings and preserved just those with higher *lcc* scores.

Table 4. Bindings with lcc *≥ 5*

ff_i	ff_j	lcc_{ij}	ff_i	ff_j	lcc_{ij}
determining	measuring	9	accurate	precise	5
celestial	heavenly	8	analyse	recording	5
intensity	amount	8	apparatus	instrument	5
swinging	turning	8	concentration	strength	5
inclination	direction	7	concentration	amount	5
instrument	telescope	7	conditions	variations	5
amount	percentage	6	frequency	wavelength	5
determining	ascertaining	6	heights	distances	5
limits	field	6	mass	weight	5
measuring	taking	6	measurement	location	5
measuring	ascertaining	6	radio	hyperbolic	5
method	system	6	recording	measuring	5
sunlight	day	6	specific	set	5
testing	measuring	6	tracing	observing	5

Table 5 presents the cluster results after two cycles of the clustering procedure starting from the Levenshtein distance. The procedure then stops, as no more matched words with *lcc* ≥ 5 have been found for our data.

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.

Table 5. Clusters for measuring instruments after 2 cycles

1. mass weight	11. hyperbolic radio radiofrequency
2. conditions variations	12. observing tracing
3. swinging turning	13. day sunlight
4. direction inclination	14. apparatus instrument telescope
5. accurate precise	15. amount concentration intensity percentage proportion rate salinity strength
6. distances heights	16. celestial heavenly
7. set specific	17. analyse ascertaining determining estimating location measuring recording taking testing
8. method system	
9. field limits	
10. frequency wavelength	

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 6 demonstrates 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 on table 5. Two out of the five new clusters (18 and 22) have far from similar members, but curiously most of the words added to the existing clusters (14 and 17) are correctly used as equivalent words.

Table 6. Clusters after replacing clusters of function words

1. mass weight	15. amount concentration intensity percentage proportion rate salinity strength
2. conditions variations	16. celestial heavenly
3. swinging turning	17. analyse ascertaining *astronomical counting detecting* determining estimating *indicates* location *making* measuring *provides* recording taking testing
4. direction inclination	18. *photometric relative*
5. accurate precise	19. *displaying producing*
6. distances heights	
7. set specific	
8. method system	
9. field limits	
10. frequency wavelength	
11. hyperbolic radio radiofrequency	

12. observing tracing	20. *angle slope*
13. day sunlight	21. *photographic visual*
14. apparatus *device* instrument *meter* telescope	22. *reticle time*

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.

4 FUTURE WORK

We have developed an algorithm to cluster semantically words from definitions based on the Levenshtein distance. However, the algorithm presented is perfectible, and there is always the alternative of exploring new techniques to improve it. Work on alignment techniques is growing. Just recently, for example, the University of Leeds has started working on grammar correction [van Zaanen 1999] and UMIST on machine translation [McTait and Trujillo 1999], based on alignment techniques.

The processes of alignment and determination of matched pairs are currently carried out on raw definition text that has undergone little preprocessing. Future work could lead to an improvement in alignment and in finding suitable matched pairs in several ways:

- by preprocessing the definitions to deal with various punctuation phenomena, particularly where abbreviations are concerned;
- by recognising and handling items such as scientific formula and other mathematical or formal notations;
- by dealing sensibly with lexicographic devices where these are consistently used (e.g. 'see also');
- by interpreting sensibly other formal lexicographic devices where these are consistently used (bold face, small capitals, etc.);
- by working with a notion of sentence boundary, or definitional unit (this relates also to the point concerned with punctuation) - this would require a greater measure of linguistic analysis;

- by attempting to align units of the same part of speech or (more ambitiously) same semantic class - this would require also a greater degree of linguistic analysis via part of speech taggers, consultation of semantic dictionaries for NLP, etc.;
- by attempting to identify compound words and terms that could be aligned and clustered as single units - the identification of compounds is one of the harder tasks in NLP and little progress has been made towards high accuracy of identification, also much human intervention is required to validate candidate compounds.

It might be thought that more work could be successfully carried out already on processing definitions in order to handle e.g. lexicographic metalanguage or the various abbreviations and notations used by lexicographers. However, experience has shown that there is much inconsistency in the writing of definitions by terminologists and lexicographers. Moreover, editing decisions play a part in e.g. altering house style in order to render a word entry more interpretable i.e. less, or indeed more, typographically complex. Thus, general automatic analysis of dictionary definitions, where these have been culled from publishers' machine readable dictionaries, is not at all straightforward, not even for definitions taken from a single source, as inconsistency is a major problem. However, with current moves towards the use of languages such as SGML and the family of mark up languages based on it, it will become increasingly possible to be able to reliably analyse and interpret definition text which is already marked up to some degree. This is however not possible at present with current dictionary resources, on the scale that would be necessary.

5 CONCLUSION

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. Alignment algorithms based on either edit distance or 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 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.

REFERENCES

Adamson, G.W. and Boreham, J. 1974. "The use of an association measure based on character structure to identify semantically related pairs of words and document titles". *Information Storage and Retrieval* 10, 253-260.

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.

Arranz, M.V. 1997. "Lexical Bottleneck in Machine Translation and Natural Language Processing: A Case Study". *Aslib 97*. London.

Chakravarthy, A.S. 1994. “Representing Information Need with Semantic Relations”. Proc. COLING-94. *The 15th International Conference on Computational Linguistics*, Kyoto, 737-741.

[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.

Gordon, A.D. 1981. *Classification*. Cambridge: University Press.

Grefenstette, G. 1996. “Evaluation Techniques for Automatic Semantic Extraction: Comparing Syntactic and Window Based Approaches”. *Corpus Processing for Lexical Acquisition*. B. Boguarev and J. Pustejovsky (eds.). Cambridge: The MIT Press.

Habert, B., Naulleau, E., and Nazarenko, A. 1996. “Symbolic word clustering for medium-size corpora”. Proc. COLING-96. *The 16th International Conference on Computational Linguistics*, Copenhagen, 490-495.

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.

McTait, K. and Trujillo, A. 1999. “A language-neutral sparse-data algorithm for extracting translation patterns.” In *Proc. Theoretical and Methodological Issues in Machine Translation (TMI-99)*, Chester, 98-108.

Miller, G.A., Fellbaum, C., Kegl, J., and Miller, K. 1990. “The Princeton Lexicon Project: A Report on WordNet”. In *BudaLEX’88 Proceedings*. T. Magay and J. Zigány (eds.). 543-558.

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.

Roget, P. 1987. *Roget’s thesaurus of English words and phrases*. Essex: Longman.

van Zaanen, M. 1999. “GAS: generic alignment system”. Unpublished draft paper. University of Leeds.

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.

Gerardo Sierra leads the research on Language Engineering at the Engineering Institute, Universidad Nacional Autónoma de México (UNAM), Ciudad Universitaria, Apdo. Postal 70-472, México 04510, D.F., Mexico. He can be reached at gsm@pumas.iingen.unam.mx.

John McNaught carries out research and teaching in Computational Linguistics at the Department of Language Engineering, UMIST, PO Box 88, Manchester M60 1QD, UK. He specialises in corpus linguistics, information extraction and computational lexicography, and has a keen interest in standards for Language Engineering, being co-Editor of the European Advisory Group for Language Engineering Standards. He can be reached at John.McNaught@umist.ac.uk.