# Disambiguating Noun Compounds with Latent Semantic Indexing

**Alan M. Buckeridge** and **Richard F. E. Sutcliffe**
Department of Computer Science and Information Systems,
University of Limerick, Limerick, Ireland
{Alan.Buckeridge, Richard.Sutcliffe}@ul.ie

## Abstract

Technical terms in text often appear as noun compounds, a frequently occurring yet highly ambiguous construction whose interpretation relies on extra-syntactic information. Several statistical methods for disambiguating compounds have been reported in the literature, often with quite impressive results. However, a striking feature of all these approaches is that they rely on the existence of previously seen unambiguous compounds, meaning they are prone to the problem of sparse data. This difficulty has been overcome somewhat through the use of hand-crafted knowledge resources to collect statistics on "concepts" rather than noun tokens, but domain-independence has been sacrificed by doing so. We report here on work investigating the application of Latent Semantic Indexing to provide a robust domain-independent source of the extra-syntactic knowledge necessary for noun compound disambiguation.

## 1 Introduction

Noun compounds are a frequently encountered construction in natural language processing (NLP), consisting of a sequence of two or more nouns which together function syntactically as a noun. In English, compounds consisting of two nouns are predominantly right-headed. However, compound construction is recursive and both the modifier and the head can themselves be compounds, resulting in structural ambiguities. Consider the following pair of noun compounds:

1. (a) (*cantilever* (*swing wing*))
   (b) ((*information retrieval*) *experiment*)

Both compounds consist of the same parts-of-speech, yet the structures differ: (1a) is right-branching, while (1b) is left-branching.

Phrase structure grammar rules for noun compounds are often similar in form to $\overline{N} \rightarrow \overline{N}\ \overline{N}$ (Lauer, 1995). This rule is applied once to two-word noun compounds, and recursively in the case of longer compounds; therefore the syntax of compounds longer than two words is underconstrained by grammar, resulting in a syntactic ambiguity which grows exponentially with the length of the compound.

Besides causing problems for syntactic parsers, the ambiguities inherent in these structures pose difficulties for NLP systems which attempt to analyse the underlying semantic relationships present in compound technical terms. Often, the first step in such analyses is to decompose terms into nested modifier-head pairs (Barker, 1998). However, such a decomposition is non-trivial for the case of compounds consisting of three or more nouns due to the structural ambiguity of these constructions.

The identification of modifier-head pairs in compounds also has applications within the field of information retrieval (IR). Several studies have shown that extracting modifier-head pairs from text and including these as compound indexing terms can improve recall and precision (Evans and Zhai, 1996; Pohlmann and Kraaij, 1997; Strzalkowski and Vauthey, 1992). Identification of these noun modifier relationships is also important for terminology translation. However, obtaining correct modifier-head pairs is once again hampered by "the notorious ambiguity of nominal compounds" (Strzalkowski and Vauthey, 1992, p.107).

To summarise, the syntactic disambiguation of noun compounds is important for several NLP applications; however, disambiguation is difficult because attachments within compounds are not syntactically governed. Clearly, then, this lack of syntactic constraints forces us to consider the use of extra-syntactic factors in the process of disambiguation. The work reported here describes an approach for automatically deriving a syntactical analysis of noun compounds by adapting Latent Semantic Indexing, a well-established IR technique, to supply this extra-syntactic information.

## 2 Previous Work

The majority of corpus statistical approaches to compound disambiguation use a variation of what Lauer (1995) refers to as the *adjacency algorithm*. This algorithm was originally proposed by Marcus (1980), and essentially operates by comparing the acceptability of immediately adjacent noun pairs. Specifically, given a sequence of three nouns *n1 n2 n3*, if (*n2 n3*) is a more acceptable constituent than (*n1 n2*), then build (*n1* (*n2 n3*)); else build ((*n1 n2*) *n3*).

There remains the question of how "acceptability" is to be determined computationally. Several researchers (e.g., (Barker, 1998; Evans and Zhai, 1996; Pohlmann and Kraaij, 1997; Pustejovsky et al., 1993; Strzalkowski and Vauthey, 1992)) collect statistics on the occurrence frequency of structurally unambiguous two-noun compounds to inform the analysis of the ambiguous compound. For example, given the compound "computer data bases", the structure (*computer* (*data bases*)) would be preferred if (*data bases*) occurred more frequently than (*computer data*) in the corpus. However, by assuming that sufficient examples of subcomponents exist in the training corpus, all the above approaches risk falling foul of the *sparse data problem.* Most noun-noun compounds are rare, and statistics based on such infrequent events may lead to an unreliable estimation of the acceptability of particular modifier-head pairs.

The work of Resnik (1993) goes some way towards alleviating this problem. Rather than collecting statistics on individual words, he instead counts co-occurrences of *concepts* (as represented by WordNet synsets). He uses these statistics to derive a measure, motivated by information theory, called *selectional association* (see Resnik (1993) for full details). "Acceptability" in the adjacency algorithm is then measured in terms of the selectional association between a modifier and head. Selectional associations were calculated by training on approximately 15,000 noun-noun compounds from the *Wall Street Journal* corpus in the Penn Treebank. Of a sample of 156 three-noun compounds drawn from the corpus, Resnik's method achieved 72.6% disambiguation accuracy.

Lauer (1995) similarly generalises from individual nouns to semantic classes or concepts; however, his classes are derived from semantic categories in Roget's Thesaurus. Similar to the approaches discussed above, Lauer extracts a training set of approximately 35,000 unambiguous noun-noun modifier-head compounds to estimate the degree of association between Roget categories. He calls this measure *conceptual association*, and uses this to calculate the acceptability of noun pairs for the disambiguation of three-noun compounds. However, his approach differs from most others in that he does not use the adjacency algorithm, instead using a *dependency algorithm* which operates as follows: Given a three-noun compound *n1 n2 n3*, if (*n1 n3*) is more acceptable than (*n1 n2*), then build (*n1* (*n2 n3*)); else build ((*n1 n2*) *n3*).

Lauer tested both the dependency and adjacency algorithms on a set of 244 three-noun compounds extracted from Grolier's Encyclopedia and found that the dependency algorithm consistently outperformed the adjacency algorithm, achieving a maximum of 81% accuracy on the task. Overall, he found that estimating the parameters of his probabilistic model based on the distribution of concepts rather than that of individual nouns resulted in superior performance, thus providing further evidence of the effectiveness of conceptual association in noun compound disambiguation.

All these approaches rely on a variation of finding subconstituents elsewhere in the corpus and using these to decide how the longer, ambiguous compounds are structured. However, there is always the possibility that these systems might encounter modifier-head pairs in testing which never occurred in training, forcing the system to "back off" to some default strategy. This problem is alleviated somewhat in the work of Resnik and Lauer where statistics are collected on pairs of *concepts* rather than pairs of noun tokens. However, the methods of

Resnik and Lauer both depend on hand-crafted knowledge sources; the applicability of their approaches is therefore limited by the coverage of these resources. Thus their methods would almost certainly perform less well when applied to more technical domains where much of the vocabulary used would not be available in either WordNet or Roget's Thesaurus. Knowledge sources such as these would have to be manually augmented each time the system was ported to a new domain. Therefore, it would be preferable to have a method of measuring conceptual associations which is less domain-dependent and which does not rely on the presence of unambiguous subconstituents in training; we investigated whether Latent Semantic Indexing might satisfy these requirements.

## 3 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a variant of the vector-space approach to information retrieval. It takes as input a collection of documents, from which it constructs an $m \times n$ word-document matrix $A$; cell $a_{ij}$ of the matrix denotes the frequency with which term $i$ occurs in document $j$. At the core of LSI is *singular value decomposition* (SVD), a mathematical technique closely related to eigenvector decomposition and factor analysis. SVD factors the matrix $A$ into the product of three matrices: $A = U\Sigma V^T$. $U$ and $V$ contain the left and right singular vectors of $A$, respectively, while $\Sigma$ is a diagonal matrix containing the singular values of $A$ in descending order. By retaining only the $k$ largest singular values[1] and setting the remaining smaller ones to zero, a new diagonal matrix $\Sigma_k$ is obtained; then the product of $U\Sigma_k V^T$ is the $m \times n$ matrix $A_k$ which is only approximately equal to $A$. This truncated SVD re-represents the word-document relationships in $A$ using only the axes of greatest variation, in effect compressing and smoothing the data in $A$. It is this compression step which is said to capture important regularities in the patterns of word co-occurrences while ignoring smaller variations that may be due to idiosyncrasies in the word usage of individual documents. The result of condensing the matrix in this way is that words which occur in similar documents will be represented by similar vectors, even if these words never actually co-occur in the same document. Thus it is claimed that LSI captures deeper associative relationships than mere word-word co-occurrences. See Berry et al. (1995) and Deerwester et al. (1990) for more thorough discussions of SVD and its application to information retrieval.

[1]The optimal value of $k$ may only be determined empirically, and will depend on the particular application.

Because word vectors are originally based on their distribution of occurrence across documents, each vector can be interpreted as a summary of a word's contextual usage; words are thus similar to the extent that they occur in similar contexts. Of interest for our purposes is the fact that a measure of the similarity or association between pairs of words can be calculated geometrically, typically by computing the cosine of the angle between word vectors. Any two words, which may or may not occur adjacently in text, can be compared in this way; this frees us from the restriction of relying on unambiguous subconstituents in training to inform the analysis of ambiguous compounds in testing.

There is a growing body of literature indicating that distributional information of the kind captured by LSI plays an important role in various aspects of human cognition. For the work reported here, the most interesting aspect of distributional information is its purported ability to model conceptual categorisation. Several studies (Burgess and Lund, 1999; Laham, 1997; Landauer et al., 1998; Levy and Bullinaria, 2001) have shown that similarity between concepts can be measured quite successfully using simple vectors of contextual usage; results show that the performance of such systems correlates well with that of humans on the same tasks. These results are all the more impressive when we consider that such systems use no hand-coded semantic knowledge; the conceptual representations are derived automatically from training corpora.

Noun compound disambiguation appears to be an NLP application for which such measures of conceptual association would be useful. Both the adjacency and dependency algorithms described above in Section 2 rely on some measure of the "acceptability" of pairs of nouns to disambiguate noun compounds. Techniques such as LSI offer a simple, robust, and domain-

| Noun Compound | Branching |
|---|---|
| ((Ami Pro) document) | Left |
| (volunteer (rescue workers)) | Right |
| (tourist (exchange rates)) | Right |
| ((cluster analysis) procedure) | Left |
| ((data base) subcommittee) | Left |
| (Windows (Control Panel)) | Right |

Table 1: Some example noun compounds taken from our test set. Each row shows an example of a manually bracketed compound, along with its branching direction.

independent way in which concepts and the associations between them can be represented. In the next section, we describe an experiment exploring the efficacy of LSI's conceptual representations in disambiguating noun compounds.

## 4 LSI and Noun Compound Disambiguation

### 4.1 Method

#### 4.1.1 Materials

We used four corpora in our study: The *Lotus Ami Pro Word Processor for Windows User's Guide Release 3*, a software manual (AmiPro); document abstracts in library science (CISI); document abstracts on aeronautics (CRAN); and articles from Time magazine (Time). We first ran the LSI software on the corpora to create the word-by-document matrices. The software also subsequently performed singular value decomposition on the resulting matrices. Stopwords were not excluded, as previous experience had shown that doing so degraded performance slightly.

We used Brill's (1994) tagger to identify three-noun sequences in each of the corpora. Tagging was imperfect, and sequences which were not true three-noun compounds were discarded. The remaining noun compounds were bracketed manually and constituted test sets for each corpus; some examples are shown in Table 1. Table 2 summarises the datasets used in our study.

#### 4.1.2 Procedure

Both the adjacency and dependency models were investigated (see Section 2). Recall that the adjacency algorithm operates by comparing the acceptability of the subcomponents (*n1 n2*) and (*n2 n3*), whereas the dependency algorithm compares the acceptability of (*n1 n2*) and (*n1 n3*). "Acceptability" in our approach was measured by calculating the cosine of the angle between each pair of word vectors. The cosine ranges from $-1.0$ to $1.0$; a higher cosine indicated a stronger association between each word in a pair. In the case of a tie, a left branching analysis was preferred, as the literature suggests that this is the more common structure (Lauer, 1995; Resnik, 1993). Thus a default strategy of always guessing a left branching analysis served as the baseline in this study. Each of the corpora contained terms not covered by WordNet or Roget's; thus it was not possible to use the techniques of Resnik (1993) and Lauer (1995) as baselines.

As we could not tell beforehand what the optimal value of $k$ would be (see Section 3 above), we used a range of factor values. The values used ranged from 2 to the total number of documents in each collection. For each factor value, we obtained the percentage accuracy of both the adjacency and dependency models.

### 4.2 Results and Discussion

The results of the experiment are summarised in Table 3 and Figure 1. In most cases the performance rises quickly as the number of SVD factors used increases, and then tends to level off. The best performance was 84% for the AmiPro collection, obtained using the adjacency algorithm and 280 SVD factors. As the task involved choosing the best binary bracketing for a noun compound, we would expect an accuracy of 50% by chance. These results compare favourably with those of Resnik (1993) and Lauer (1995) (73% and 81%, respectively), but as their studies were conducted on different corpora, it would be imprudent to make direct comparisons at this stage. Results for the other collections were less impressive—however, above-baseline performances were obtained in each case.

Substantial differences in the performances of the adjacency and dependency algorithms were only observed for the AmiPro collection, suggesting that the superior performance of the dependency algorithm in Lauer's (1995) study was largely corpus-dependent. This is reinforced by the considerably superior performance of the

| Collection Name | AmiPro | CISI | CRAN | Time |
|---|---|---|---|---|
| Number of Documents | 704 | 1,460 | 1,400 | 425 |
| Number of Tokens | 138,091 | 187,696 | 217,035 | 252,808 |
| Mean Tokens per Type | 46.3 | 18.7 | 26.2 | 11.5 |
| Number of test compounds | 307 | 235 | 223 | 214 |

Table 2: Characteristics of the datasets.

| Name | AmiPro | CISI | CRAN | Time |
|---|---|---|---|---|
| Baseline | 58% | 63% | 74% | 48% |
| Adjacency | 84% (280) | 73% (800) | 75% (700) | 62% (370) |
| Dependency | 70% (200) | 70% (1100) | 75% (600) | 62% (240) |

Table 3: Percentage disambiguation accuracy on each collection. The Baseline row shows the accuracy of always choosing a left-branching analysis. Highest accuracies for the Adjacency and Dependency algorithms are shown, with the corresponding number of SVD factors in parentheses.

adjacency algorithm on the AmiPro data set.

Another interesting finding was that there were more right-branching (52%) than left-branching (48%) compounds in the Time collection. This contrasts with previous studies which discuss the predominance of left-branching compounds, and suggests that the choice for the default branching must be corpus-dependent (see Barker (1998) for similar findings).

There also appears to be a positive relationship between performance and the token-type ratio. The number of tokens per type in the AmiPro collection was 46.3; the worst performance was found for the Time collection, which had only 11.5 tokens per type. There are at least two possible explanations for this relationship between performance and token-type ratio: First, there were more samples of each word type in the AmiPro collection—this may have helped LSI construct vectors which were more representative of each word's contextual usage, thus leading to the superior performance on the AmiPro compounds.

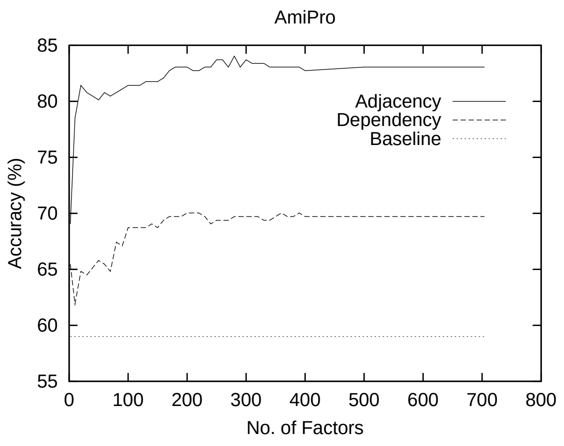
Second, LSI constructs a single vector for each token—if a particular token is polysemous in text then its vector will be a "noisy" amalgamation of its senses, a factor often contributing to poor performance. However, due to the controlled language and vocabulary used in the software manual domain, few if any of the words in the AmiPro collection are used to convey more than one sense; once again, this may have resulted in "cleaner", more accurate vectors leading to the superior disambiguation performance on the AmiPro compounds.

These points lead us to the tentative suggestion that our approach appears most suitable for technical writing such as software manuals. As usual, however, this is a matter for future investigation.

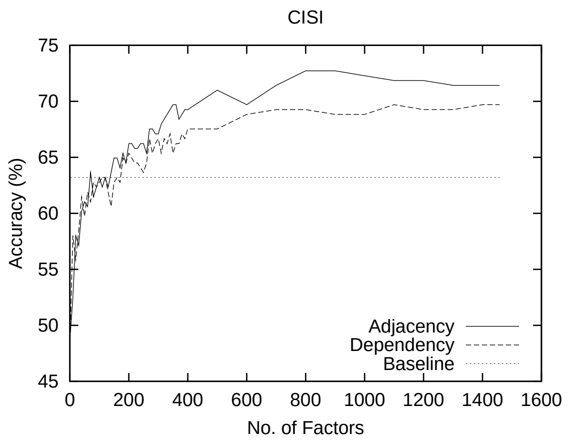
## 5 Conclusions and Future Research

In this study, we extended LSI beyond its usual remit by adopting it as a measure of conceptual association for noun compound disambiguation. The results reported here are encouraging, the highest accuracy of 84% on the AmiPro collection indicating the potential of our approach. However, poorer performance was obtained for the other collections indicating that there is much room for improvement. We therefore intend to pursue our investigation of the utility of applying vector-based measures of semantic similarity to the problem of syntactic disambiguation. An attractive feature of this approach for the processing of terminology is that it requires no manually constructed knowledge sources, meaning that it does not suffer the same coverage limitations as the methods of Lauer (1995) and Resnik (1993). In principle, our approach can be applied to any domain.

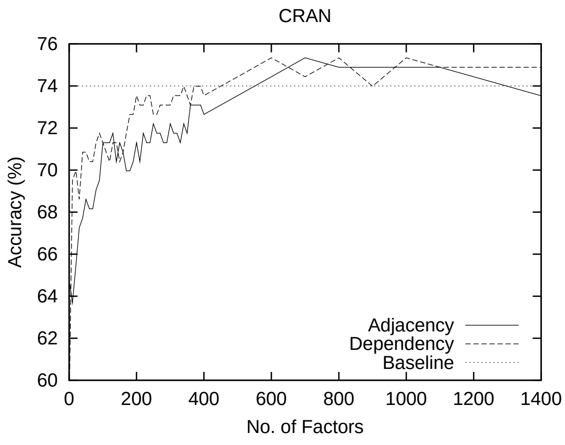
Another attractive feature is that it does not rely on counts of unambiguous subconstituents in training. This means that it can be applied to novel compounds for which no subcompounds exist in training, something which would not be



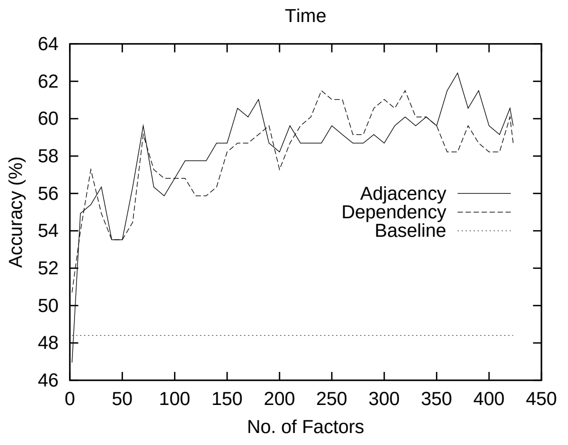
(a) AmiPro



(b) CISI



(c) CRAN



(d) Time

Figure 1: Results of an experiment investigating noun compound disambiguation using LSI. Each figure shows percentage disambiguation accuracy of the *adjacency* and *dependency* models for a range of SVD factors. The percentage of left-branching compounds in each test set, which served as the baseline in our study, is also shown for comparison.

possible for the statistical techniques outlined in Section 2. Our next step will thus be to investigate the efficacy of our approach on novel compounds.

We are currently examining the use of other techniques for deriving vector-based measures of conceptual association; preliminary investigations using a "sliding window" method (Burgess and Lund, 1999; Levy and Bullinaria, 2001) to disambiguate compounds from the AmiPro corpus show results even better than those reported here. Present work involves setting various parameters (e.g., window size, similarity metric, weighting method) to study their effect on performance. We are continuing to test both the adjacency and dependency algorithms on this corpus, and have consistently found better performance using the former.

Future work will involve continuing to test the technique in other domains; we also intend training on larger and more diverse corpora. Furthermore, we plan to investigate other examples of syntactic ambiguity, such as prepositional phrase attachment. Such structures pose many problems for traditional NLP systems, but may prove amenable to the techniques discussed in this paper.

## 6 Acknowledgements

Our thanks go to Pat Hickey of ECE, University of Limerick, for technical assistance; and to two anonymous reviewers for helpful comments.

## References

K. Barker. 1998. A trainable bracketer for noun modifiers. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence*, pages 196–210, Vancouver.

M. W. Berry, S. T. Dumais, and T. A. Letsche. 1995. Computational methods for intelligent information access. In *Proceedings of Supercomputing '95*, San Diego, CA.

E. Brill. 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*.

C. Burgess and K. Lund. 1999. The dynamics of meaning in memory. In E. Dietrich and A. Markman, editors, *Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines*, pages 17–56. Lawrence Erlbaum Associates Inc., Hillsdale, NJ.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

D. A. Evans and C. Zhai. 1996. Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 17–24, Santa-Cruz, CA, June.

D. Laham. 1997. Latent Semantic Analysis approaches to categorization. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, page 979. Erlbaum.

T. K. Landauer, P. W. Foltz, and D. Laham. 1998. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284.

M. Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds.* Ph.D. thesis, Macquarie University, Sydney, Australia.

J. P. Levy and J. A. Bullinaria. 2001. Learning lexical properties from word usage patterns: Which context words should be used? In *Proceedings of the Sixth Neural Computation and Psychology Workshop*, pages 273–282. London: Springer.

M. Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, MA.

R. Pohlmann and W. Kraaij. 1997. The effect of syntactic phrase indexing on retrieval performance for Dutch texts. In *Proceedings of RIAO '97*, pages 176–187, Montréal, June.

J. Pustejovsky, S. Bergler, and P. Anick. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

P. S. Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships.* Ph.D. thesis, University of Pennsylvania.

T. Strzalkowski and B. Vauthey. 1992. Information retrieval using robust natural language processing. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Newark, Delaware, USA.