Lexical Disambiguation Using Constraint Handling In Prolog (CHIP) *

George C. Demetriou
Centre for Computer Analysis of Language And Speech (CCALAS)
Artificial Intelligence Division, School of Computer Studies, University of Leeds
Leeds, LS2 9JT, United Kingdom

1 Introduction

Automatic sense disambiguation has been recognised by the research community as very important for a number of natural language processing applications like information retrieval, machine translation, or speech recognition. This paper describes experiments with an algorithm for lexical sense disambiguation, that is, predicting which of many possible senses of a word is intended in a given sentence. The definitions of senses of a given word are those used in LDOCE, the Longman Dictionary of Contemporary English [Procter *et al.*, 1978]. The algorithm first assigns a set of meanings or senses drawn from LDOCE to each word in the given sentence, and then chooses the combination of word-senses (one for each word in the sentence), yielding the maximum semantic overlap. The metric of semantic overlap is based on the fact that LDOCE sense definitions are made in terms of the Longman Defining Vocabulary, effectively a (large) set of semantic primitives. Since the problem of finding the word-sense-chain with maximum overlap can be viewed as a specialised example of the class of constraint-based optimisation problems for which Constraint Handling In Prolog (CHIP) was designed, we have chosen to implement our algorithm in CHIP.

2 Background: LDOCE, Word Sense Disambiguation and related work

LDOCE's important feature is that its definitions (and examples) are written in a controlled vocabulary of 2187 words. A definition is therefore always written in simpler terms than the word it describes. These 2187 words effectively constitute semantic primitives, and any particular word-sense is defined by a set of these primitives.

Several researchers have been experimented with lexical disambiguation using MRDs, including [Lesk, 1986; Wilks *et al.*, 1989; McDonald *et al.*, 1990; Veronis and Ide, 1990; Guthrie *et al.*, 1991; Guthrie *et al.*, 1992]. Lesk's technique decides the correct sense of a word by counting the overlap between a dictionary sense definition (of the word to be disambiguated) and the definitions of the nearby words in the phrase. Performance (using brief experimentation) was reported 50-70% and the results were roughly comparable between Webster's 7th Collegiate, Collins English Dictionary and Oxford Advanced Learner's Dictionary of Current English. Methods based on co-occurence statistics have been used by [Wilks *et al.*, 1989; McDonald *et al.*, 1990; Guthrie *et al.*, 1991]. By co-occurence is meant the preference two words appear together in the same context. [Wilks *et al.*, 1989] computed lexical neighbourhoods for all the words of the controlled vocabulary of LDOCE. This neighbourhood information is used for partitioning the words according to the senses they correspond to in order to make a classification of the senses. Their results for using occurences of the word *bank* were about 53% for the classification of each instance into one of the thirteen sense definitions of LDOCE and 85-90% into one of the more general coarse meanings. Neighbourhoods were used by [McDonald *et al.*, 1990] for expanding the word sense definitions. The union of neighbourhoods is then intersected with the local context and the largest overlap gives the most likely sense. A similar technique is used by [Guthrie *et al.*, 1991] except that they define neighbourhoods according to subject categories (i.e engineering, economic etc.) based on the subject code markings of the on-line version of LDOCE.

Closer to the work we describe in this paper is [Guthrie *et al.*, 1992]'s. They try to deal with large-scale text data disambiguation problems. Their method is based on the idea that the correct meaning of a complete phrase should be extracted by concurrent evaluation of sets of senses for the words to be disambiguated. They count the overlap between sense definitions of the words of the sentence as they appear in the on-line version of LDOCE. The problem is that the number of sense combinations increases rapidly if the sentence contains ambiguous words having a considerable number of sense definitions in LDOCE (say that word A has X different senses in LDOCE, B has Y and C has Z, then the number of possible sense combinations of the phrase ABC is X*Y*Z, e.g if X=Y=Z=10 sense definitions for each word then we have 1000 possible sense combinations). Simulated annealing is used by [Guthrie *et al.*, 1992] to reduce the search space and find an optimal (or near-optimal) solution without generating and evaluating all possible solutions, or pruning the search space and testing a well-defined subspace of reasonable candidate solutions. The success of their algorithm is reported 47% at sense level and 72% at homograph level using 50 example sentences

*This work was supported by the Greek Employment Manpower Organisation (OAED), Ministry of Labour, as part of an 1991-93 scholarship scheme.

from LDOCE.

3 CHIP: Constraint Handling In Prolog

We decided it was worthwhile investigating the use of a constraint handling language so that we could exhaustively search the space by applying CHIP's optimisation procedures. A CHIP compiler is available from International Computers Limited (ICL) as part of its DecisionPower prolog-based toolkit [1]. CHIP extends usual Prolog-like logic programming by introducing three new computation domains of finite restricted terms, boolean terms and linear rational terms. Another feature offered by CHIP is the demon constructs used for user-defined constraints to implement the local propagation. For each of them CHIP uses specialised constraint solving techniques: consistency techniques for finite domains, equation solving in Boolean algebra, and a symbolic simplex-like algorithm. CHIP's declarations are used to define the domain of variables or to choose one of the specialised unification algorithms; they can be: (1) finite domains (i.e. variables range over finite domains and terms are constructed from natural numbers, domain variables over natural numbers and operators); (2) boolean declarations or (3) demon declarations (for specifying a data-driven behaviour; they consist of a set of rules which describe how a constraint can be satisfied). In addition, classes of built-in predicates over finite domain variables exist for: (1) arithmetic and symbolic constraints (basic constraints for domain variables), (2) choice predicates (help making choices), (3) higher order predicates (providing optimisation methods for combinatorial problems using depth-first and branch and bound strategies) and (4) extra-logical predicates (for help in debugging processes). Forward checking and looking ahead inference rules are introduced for the control mechanism in the computation of constraints using finite domains. Auxiliary predicates to monitor or control the resolution process in the CHIP environment also exist.

In our case we were particularly interested in transforming the general structure of our algorithm into a form usable by CHIP's choice and higher order built-in predicates. Choice predicates are used for the automatic generation of word-sense combinations and higher order predicates facilitate the process of finding the most likely combination according to the 'score' of overlap. To get an idea of this kind of implementation the main core of the optimisation part of our program looks like this:

```
optimize(Words,Choice,Cost):-
        minimize((makeChoice(Choice),
        findCost(Choice,Cost)), Cost).
```

[1] DecisionPower donated by ICL under the University Funding Council's Knowledge and Constraint Management (KCM) Initiative.

`Minimize` is one of CHIP's optimisation built-in predicates. `Words` represents the list of ambiguous words submitted to the program and `Choice` a list of domain variables for the selection of sense definitions. `Cost` is a domain variable whose domain is constrained to an arithmetic term. For our purposes, `Cost` was `Max-Overlap` where `Max` (a maximum possible score) is large enough so that `Overlap` (score of overlap in a sense definition) can never exceed it. Any answer substitution that causes `(makeChoice(Choice), findCost(Choice,Cost))` to be ground also causes `Cost` to be ground. The search then backtracks to the last choice point and continuous along another branch. The cost of any other solution found in the sub-tree must be necessarily lower (i.e `Overlap` must be higher) than the last one found, because `Cost` is constrained to that bound. This process of backtracking for better solutions and imposing constraints on `Cost` continues until the space has been searched implicitly. At the end, `(makeChoice(Choice), findCost(Choice,Cost)` is bound to the last solution found which is the optimal one.

4 Algorithm

Our method is based on the overlap between sense definitions of the words to be disambiguated. This is similar to [Guthrie *et al.*, 1992] although there are distinct differences on the scoring method and the implementation. To illustrate our method we use the following example and describe each phase:

```
The bank arranged for an overdraft on my
account.
```

4.1 Step 1

All the common function words (particles) belonging to our 'stop list' (a set of 38 very common words) e.g. for our example the set of words (the, for, an, on, my) should be removed. Function words tend to appear very often both in context and in sense definitions for syntactic and style reasons rather than pure semantics. Since our algorithm is intended to maximise overlap the participation of function words in a definition chain could lead to false interpretation for the correct sense combination. Moreover, function words are usually much more ambiguous than content words (for example, there are 21 listed senses of the word *the* and 35 of *for* in LDOCE). Thus, the searching process could be significantly increased without any obvious benefit to the resolution of ambiguity of context words as explained above. Words of the 'stop list' have also been removed from the sense definitions and the remaining words are stemmed so that only their roots appear in the definition. With this way, derived (or inflected) forms of the same word can be matched together. For this reason, the program also uses the primitive or root forms of the input words. After function-word-deletion the program is given the following set of words:

```
    bank arrange overdraft account
```

These are processed according to their stemmed sense definitions in LDOCE, represented as Prolog database structures such as:

```
bank([
      [bank, land, along, side, river,
       lake],
      [bank, earth, heap, field, garden,
       make, border, division],
      [bank, mass, snow, cloud, mud],
      [bank, slope, make, bend, road, race,
       track, safer, car, go, round],
      [bank, sandbank],
      [bank, car, aircraft, move, side,
       higher, other, make, turn],
      [bank, row, oar, ancient, boat, key,
      typewriter],
      [bank, place, money, keep, pay,
       demand, relate, activity, go],
      [bank, place, something, hold, ready,
       use, organic, product, human,
       origin, medical, use],
      [bank, person, keep, supply, money,
       piece, payment, use, game, chance],
      [bank, win, money, game, chance],
      [bank, put, keep, money, bank],
      [keep, money, state, bank]]).
```

The conventions we use are: a) Each word to be disambiguated is the functor of a predicate, containing a list with stemmed sense definitions (in lists). b) We do not put a subject code in each sense definition (as [Guthrie *et al.*, 1992] do). Instead we put the word to be disambiguated as a member of the list of each sense definition. The rationale behind this is that although a word put in its sense definition cannot help with the disambiguation of itself, it can provide help in the disambiguation of the other words if it appears in their sense definitions. c) Compound words of the form 'race-track' were used as two words 'race' and 'track'.

4.2 Step 2

The algorithm generates sense combinations by going through the sense definitions for each word one by one. For example, a sense combination can be called by taking the 8th sense of *bank* (call it *b8*, see above), the first sense of *arrange* (*a1=[arrange, set, good, please, order]*), the definition of *overdraft* (*o1=[overdraft, sum, lend, person, bank, more, money, have, bank]*), and the seventh of *account* (*c7=[account, sum, money, keep, bank, add, take]*).

The scoring process for this sense combination is given by taking the definitions pairwise and counting the overlap of words between them. Before the program proceeds to counting, redundancy of words is eliminated in each sense definition in order to prevent each word from being counted more than once. The algorithm checks for word overlap in advance and in case this constraint is not satisfied, the combination is discarded and a new one generated so that only overlapping combinations are considered. For each combination the total score is the sum of all the overlaps pairwise. This means that for n ambiguous words in the sentence the program counts the overlap for all $n!/(2!(n-2)!)$ pair combinations and add them together. For the above example,

```
score(b8a1o1c7)= overlap(b8a1)
                +overlap(b8o1)
                +overlap(b8c7)
                +overlap(a1o1)
                +overlap(a1c7)
                +overlap(o1c7)
                =0+2+3+0+0+3= 8
```

This scoring method is quite different to the one used by [Lesk, 1986]. Lesk simply counted overlaps by comparing each sense definition of a word with all the sense definitions of the other words. [Guthrie *et al.*, 1992] use a similar method. It is different in that if there is a subject (pragmatic) code for a sense definition they put this subject code as a single word in the definition list. Then they go through each list of the words, put the word in an array and begin a counter at 0. If the word is already in the list they increment the counter. So if, for example, three definitions have the same word they count it 2, while with our method this counts 3 and it seems that our method generally overestimates. Although no evidence of the best scoring scheme can be obtained without results we think that our method may work better in chains where all definitions share a common word (and this overestimation goes higher compared to [Guthrie *et al.*, 1992]) which may indicate a strong preference for that combination.

4.3 Step 3

If a new generated combination has a higher score, it is considered as a better solution. This new (temporary maximum) score acts as a constraint (a lower minimum) to new generated combinations. At the end, the most likely sense combination is the one with the highest score. Implementation in CHIP guarantees to give one and only solution (or no solution if no overlapping combination exists). The way choices are generated is by taking at the beginning the first sense definition for each word in the sentence. This is because the most common or most typical meanings of a word are shown first in LDOCE. Following choices replace the definitions of the words one by one according to the order these words are submitted to the program. An example sentence and its output is illustrated next [Procter *et al.*, 1978]:

```
Sentence: a tight feeling in the chest.
```

```
Total number of sense combinations: 392

Optimal solution found:

tight = [tight, have, produce,
         uncomfortable, feeling, closeness,
         part, body]

feeling = [feeling, consciousness,
           something, feel, mind, body]

chest = [chest, upper, front, part, body,
         enclose, heart, lung]

Its Score is: 5
```

5 Results

Evaluation of a dictionary-based lexical disambiguation routine is difficult since the preselection of the correct senses is in practice very difficult and time-consuming. The most obvious technique would seem to be to start by creating a benchmark of sentences, disambiguating these manually using intuitive linguistic and lexicographical expertise to assign the best sense-number to each word. However, distinctions between senses are often delicate and fine-grained in a dictionary, and it is often hard to fit a particular case into one and only one category. It is typical in work of this kind that researchers use human choices for the words or sentences to disambiguate and the senses they will attempt to recognise [Guthrie, 1993]. In most of the cases [Hearst, 1991; McDonald *et al.*, 1990; Guthrie *et al.*, 1991; Guthrie *et al.*, 1992], the number of test sentences is rather small (less than 50) so that no exact comparison between different methods can be done. Our tests included a set of 20 sentences, from sentences cited in an NLP textbook [Harris, 1985] (used to illustrate non-MRD-based semantic disambiguation techniques) example sentences cited in [Guthrie *et al.*, 1992; Lesk, 1986; Hearst, 1991] (for comparison between different lexical disambiguation routines) and examples taken from LDOCE (to assess the algorithm's performance with example sentences of particular senses in the dictionary-this might also be a way of testing the consistency of the relationship between different senses and their corresponding examples of a word in LDOCE). A sense chosen by our algorithm is compared with the 'intuitive' sense; but if there is not an exact match, we need to look further to judge how 'plausible' the predicted sense remains.

After pruning of function words, length varied from 2 to 6 content words to be disambiguated, with an average of 3.1 ambiguous words per sentence. The number of different sense combinations ranged from 15 to 126000.

Of the 62 ambiguous words, 36 were assigned senses exactly matching our prior intuitions, giving an overall success rate of 58%. Although accuracy of the results is far from 100%, the method confirms the potential contribution of the use of dictionary definitions to the problem of lexical sense disambiguation.

Ambiguous words had between 2 and 44 different senses. Investigating the success at disambiguating a particular word depended on the number of alternative senses given in the dictionary we had the following results:

No. senses per word	No. words per range	Disambiguated correctly	Success %
2-5	23	16	70
6-10	19	11	58
11-15	11	5	45
16-20	3	2	67
21-44	6	2	33

It might be expected that if the algorithm has to choose between a very large number of alternative senses it would be much likelier to fail; but in fact the algorithm held up well against the odds, showing graceful degradation in success rate with increasing ambiguity. Furthermore, success rate showed little variation with increased number of ambiguous words per sentence:

No. amb. words per sentence	No. sentences per range	Success %
2	7	64
3	8	58
4	2	63
5-6	3	50

This presumably indicates a balanced trade-off between competing factors. One might expect that each extra word brings with it more information to help disambiguate other words, improving overall success rate; on the other hand, it also brings with it spurious senses with primitives which may act as 'red herrings' favouring alternative senses for other words.

The average overlap score per sentence for the best analysis rose in line with sentence length, or rather, number of ambiguous words in the sentence:

No. ambiguous words per sentence	Average overlap for best disambiguation
2	2.2
3	3.1
4	5.0
5-6	5.7

We noticed a trend towards choosing longer sense-definitions over shorter ones (i.e senses defined by a larger set of semantic primitives tended to be preferred); 41 out of the 62 solutions given by the program (66%) were longer definitions than average. This is to be expected in an algorithm maximising overlap, as there are more primitives to overlap with in a larger definition. However, this tendency did NOT appear to imply wrong long sense were being preferred to correct short sense leading to a worsening overall success rate: of the 41 cases, 27 were correct, i.e 66% compared to 58% overall. A better interpretation of this result might be that longer definitions are more detailed and accurate, thus making a better 'target'.

Of the 26 'failures', 5 were assigned senses which were in fact incompatible with the syntactic word-class in the given sentence. This indicates that if the algorithm was combined with a word-tagger such as CLAWS [Atwell, 1983; Leech, 1983], and lexical senses were constrained to those allowed by the word-tags predicted by CLAWS, the success rate could rise to 66%. This may also be necessary in cases where LDOCE's definitions are not accurate enough. For example, trying to disambiguate the words *show*, *interest* and *music* in the sentence *'He's showing an interest in music'* [Procter *et al.*, 1978]. the program chose the eighth noun sense of *show* and the second verb sense of *interest*. This was because the occurence of the word *'do'* in both definitions resulted in a maximum overlap for that combination. However, the *'do'*s sense is completely different in each case. For the *show 'do'* was related to *'well done'* and for *interest* to *'do something'*.

Optimisation with CHIP performed well in finding the optimal solution. In all cases no other sense combination had a better score than the one found. This was confirmed by testing our algorithm in a separate implementation without any of CHIP's optimisation procedures but using a conventional method for exploring the search space for the best solution. Optimisation with CHIP was found to be from 120% to 600% faster than the conventional approach.

6 Conclusions and Future Directions

It is difficult to make a straightforward comparison with other methods for lexical disambiguation, particularly [Guthrie *et al.*, 1992]'s and [Lesk, 1986]'s, as there is no standard evaluation benchmark; but this approach seems to work reasonably well for small and medium scale disambiguation problems with a broadly similar success rate. We could try producing a much larger testbed for further comparative evaluations; but it is not clear how large this would have to be to become authoritative as an application-independent metric. Future enhancements to the approach incorporating the automatic use of the on-line subject codes and cross reference and subcategorisation systems of LDOCE can provide better results.

Concerning CHIP, it provides a platform from which we can build in order to deal with large scale disambiguation; this could be used as an alternative to numerical optimisation techniques. The approach will involve the modelling of the problem in a combinatorial form so that constraint satisfaction logic programming [Van Hentenryck, 1989] can apply. For each sense of a word we can specify a set of constraints such as its subject code(s), or part-of-speech information or both. Forward checkable (or lookahead) rules can be introduced to decrease the number of possible senses of other words in advance (say, for example, that the 'economic' sense for the word 'bank' has been chosen, then only the 'economic' or 'neutral' senses of the 'arrange', 'overdraft' and 'account' will be taken into account). This suggests a dramatic reduction on the search space; CHIP offers all the necessary arithmetic and symbolic facilities for the implementation.

Our experiments will be based on the use the machine version of LDOCE to verify the utility of this dictionary for the specific kind of applications we have in mind: the development methods and techniques that can assist large scale speech and handwriting recognition systems using semantic knowledge from already available resources (MRDs and corpora) [Atwell *et al.*, 1992]. But the problem here is somewhat different: semantic constraints must be used for the correct choice between different candidate Ascii interpretations of a spoken or handwritten word.

Acknowledgements

This paper summarizes research reported in [Demetriou, 1992].

I am grateful to Mr Eric Atwell, director of CCALAS and my supervisor, for the motivating influence and background material he provided me for this project.

I would also like to express my appreciation to Dr Gyuri Lajos for the organisational support and advice on CHIP programming and Mr Clive Souter for his useful recommendations.

References

[Atwell, 1983] Eric Atwell. Constituent-Likelihood Grammar. In *ICAME Journal of the International Computer Archive of Modern English no.7*, pages 34-67, 1983.

[Atwell *et al.*, 1992] Eric Atwell, David Hogg and Robert Pocock. Speech-Oriented Probabilistic Parser Project: Progress Reports 1&2. Technical Reports, School of Computer Studies, Leeds University, 1992.

[Demetriou, 1992] George C. Demetriou. Lexical Disambiguation Using Constraint Handling In Prolog (CHIP). MSc Dissertation, School of Computer Studies, University of Leeds, 1992.

[Guthrie *et al.*, 1991] Joe Guthrie, Louise Guthrie, Yorick Wilks and H. Aidinejad. Subject-dependent Co-occurence and Word Sense Disambiguation. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 146–152, 1991.

[Guthrie *et al.*, 1992] Joe Guthrie, Louise Guthrie and Jim Cowie. Lexical Disambiguation Using Simulated Annealing. In *Proceedings of the 14th Conference on Computational Linguistics, COLING-92*, pages 359–364, 1992.

[Guthrie, 1993] Louise Guthrie. A Note on Lexical Disambiguation. In *C. Souter and E.Atwell (eds), Corpus-based Computational Linguistics*, Rodopi Press, Amsterdam, 1993.

[Harris, 1985] Mary D. Harris. *An Introduction to Natural Language Processing.* Reston Publishing Company, 1985.

[Hearst, 1991] Marti Hearst. Toward Noun Homograph Disambiguation Using Local Context in Large Text Corpora. In *Proceedings of the 7th Annual Conference of the UW Centre for the New OED and TEXT Research, Using Corpora*, pages 1–22, 1991.

[Leech, 1983] Geoffrey Leech, Roger Garside and Eric Atwell. The Automatic Grammatical Tagging of the LOB Corpus. In *ICAME Journal of the International Computer Archive of Modern English no.7*, pages 13–33, 1983.

[Lesk, 1986] Michael Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the ACM SIG-DOC Conference*, Ontario, Canada, 1986.

[McDonald *et al.*, 1990] James E. McDonald, Tony Plate and R. Schvaneveldt. Using Pathfinder to Extract Semantic Information from Text. In *Pathfinder associative networks: studies in knowledge organisation*, R. Schvaneveldt (ed), Norwood, NJ:Ablex, 1990.

[Procter *et al.*, 1978] Paul Procter et al. *The Longman Dictionary of Contemporary English.* Longman, 1978.

[Van Hentenryck, 1989] Pascal Van Hentenryck. Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Massachusetts, 1989.

[Veronis and Ide, 1990] Jean Veronis and Nancy M. Ide. Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. In *Proceedings of the 13th Conference on Computational Linguistics, COLING-90*, Helsinki, Finland, 2, pages 389–394, 1990.

[Wilks *et al.*, 1989] Yorick Wilks, Dan Fass, Cheng-Ming Guo, James McDonald, Tony Plate and Brian Slator. A Tractable Machine Dictionary as a Resource for Computational Semantics. In *Computational Lexicography for Natural Language Processing*, B. Boguraev and T. Briscoe (eds), Longman, 1989.