Grammatical Inference and Corpus Linguistics

Andrew Roberts

Submitted in accordance with the requirements for the degree of
Master of Philosophy

The University of Leeds
School of Computing

January, 2009

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

Acknowledgements

I acknowledge the support of members of the Language research group, particularly Latifa Al-Sulaiti and Bayan Abu Shawar. I am grateful for all they have taught me. We had many fun times during our time in Leeds and it's been a pleasure to make such ever-lasting friendships.

I'd also like to extend my thanks to the School of Computing. I've enjoyed my time here tremendously; the staff, postgrads and students are first-class.

This thesis is dedicated to my supervisor, Eric Atwell – a man of unlimited generosity, encouragement and patience. I'm not sure if I would have achieved a fraction of what I have – either academically or professionally – if it wasn't for him.

My thanks also to all my friends and family who supported me during my studies.

Abstract

This thesis looks at ways to bring together two research fields: Grammatical Inference and Corpus Linguistics. Grammatical Inference is research on using Unsupervised Machine Learning to extract grammatical descriptions from a corpus or text dataset, and Corpus Linguistics is all about using corpora in linguistic research; so on the face of it the two fields should be related.

The thesis starts with a review of grammatical Inference research, and an overview of Corpus Linguistics research as represented at the International Conference on Corpus Linguistics. Part 1 presents a review of Unsupervised Grammar Inference Systems for Natural Language. Part 2 presents a snapshot of current research topics in Corpus Linguistics, as represented at the International Conference on Corpus Linguistics, and reviews the extent to which Grammar Inference has been used in Corpus Linguistics. It transpires that GI researchers focus on development of novel algorithms and have little or no interest in standardisation of corpus resources; and Corpus Linguists are largely unaware of GI as a potential tool for their research. To introduce the two research communities to each other, this thesis presents an application of Corpus Linguistics principles to GI research, and an application of GI to a Corpus Linguistics research tool. Part 3 presents the use of corpora for automatic evaluation of Grammar Inference systems. Part 4 presents aConCorde: an open-source, extendable concordancer for Arabic, which includes Grammar Inference. Part 5 presents more details of this technique, Grammatical Inference of word classes using distribution analysis of content words with respect to function words. In conclusion, CL provides principled methods for evaluation of GI, and useful applications for GI; so GI and CL research communities should work together. This thesis should contribute to increasing mutual awareness and cooperation.

Contents

Preface

Leeds University Ordinances and Regulations specify what is required of a MPhil thesis: "... the thesis must contain evidence of originality and independent critical ability and matter suitable for publication, and be of sufficient merit to qualify for the degree of Master of Philosophy." In practice, journal and conference proceedings editors would not publish a paper unless it showed "originality and independent critical ability"; so a thesis containing published papers surely meets all the criteria to qualify for the degree of Master of Philosophy.

So, this thesis is based on three of my publications:

Roberts, Andrew; Atwell, Eric. Unsupervised Grammar Inference Systems for Natural Language. Research Report number 2002.20, School of Computing, University of Leeds. 2002.

Roberts, Andrew; Atwell, Eric. The use of corpora for automatic evaluation of grammar inference systems in: Archer, D, Rayson, P, Wilson, A & McEnery, T (editors) Proceedings of CL2003: International Conference on Corpus Linguistics, pp. 657-661 Lancaster University. 2003.

Roberts, Andrew; Al-Sulaiti, Latifa; Atwell, Eric. aConCorde: Towards an open-source, extendable concordancer for Arabic. Corpora, vol. 1, pp. 39-57. 2006.

During my research period, I also contributed to a number of other published papers; these are not included in the Thesis as they were either peripheral to my research focus, or they report work at an early stage:

Atwell, Eric; Abu Shawar, Bayan; Babych, Bogdan; Elliott, Debbie; Elliott, John; Gent, Paul; Hartley, Anthony; Hu, Xunlei Rose; Medori, Julia; Oba, Toshifumi; Roberts, Andy; Scharoff, Serge; Souter, Clive. Corpus Linguistics, Machine Learning and Evaluation: Views from Leeds. Research Report number 2003.02, School of Computing, University of Leeds. 2003.

van Zaanen, Menno; Roberts, Andrew; Atwell, Eric. A multilingual parallel parsed corpus as gold standard for grammatical inference evaluation in: Proceedings of LREC'04 Workshop on The Amazing Utility of Parallel and Comparable Corpora, pp. 58-61 European Language Resources Association. 2004.

Abu Shawar, Bayan; Atwell, Eric; Roberts, Andrew. FAQchat as an Information Retrieval system in: Vetulani, Z (editors) Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the Second Language and Technology Conference, pp. 274-278. 2005.

Al-Sulaiti, Latifa; Roberts, Andrew; Atwell, Eric. The use of corpora and concordance in the teaching of contemporary Arabic in: Proceedings of EuroCALL 2005.

Roberts, Andrew; Al-Sulaiti, Latifa; Atwell, Eric. aConCorde: towards a proper concordance of Arabic. in: Proceedings of Corpus Linguistics 2005.

Atwell, E; Shawar, B A; Roberts, A; Al-Sulaiti, L. Unsupervised learning of linguistic significance. in: Barber, S, Baxter, P D, Mardia, K V & Walls, R E (editors) Interdisciplinary Statistics and Bioinformatics, pp. 107-108 University of Leeds. 2006.

Atwell, Eric; Roberts, Andrew. Combinatory hybrid elementary analysis of text in: Kurimo, M, Creutz, M & Lagus, K (editors) Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes. 2006.

Atwell, Eric; Roberts, Andy. CHEAT: combinatory hybrid elementary analysis of text in: Proceedings of Corpus Linguistics 2007.

Chapter 1:
Unsupervised Grammar Inference Systems for Natural Language

In recent years there have been significant advances in the field of Unsupervised Grammar Inference (UGI) for Natural Languages such as English or Dutch. This survey presents a broad range of UGI implementations, where we can begin to see how the theory has been put in to practise. Several mature systems are emerging, built using complex models and capable of deriving natural language grammatical phenomena. The range of systems is classified into: models based on Categorial Grammar (GraSp, CLL, EMILE); Memory Based Learning models (FAMBL, RISE); Evolutionary computing models (ILM, LAgts); and string-pattern searches (ABL, GB). An objectively measurable statistical comparison of performance Of the systems reviewed is not yet feasible. However, their merits and shortfalls are discussed, as well as a look at what the future has in store for UGI.

1.1 Introduction

Gold's seminal paper (Gold, 1967) showed that it was theoretically impossible to extract a definitive grammar from examples of a target language, unless selected negative counterexamples were also available. Encouragingly, this theoretical hurdle has not deterred research: the natural language learning (NLL) community has witnessed rapid advances in unsupervised grammar inference.

Interesting developments have arisen from the psychological perspective of this task: research has been driven to devise psychologically plausible models of natural language acquisition.

Grammar inference is not just restricted to its classical domain of syntactic pattern recognition, with many useful functions within other levels of Natural Language Processing, such as speech processing. It has expanded in to other important areas, for example, information retrieval (Freltag, 1997; Hong and Clark, 2001) and gene analysis (Dong and Searls, 1994).

This survey focuses on recent UGI implementations. Some are pitched as UGI systems in their own right, others could be classed as solutions to subtasks that would be useful to language learning systems. It is worth noting that this is not a comprehensive review of every such system, but more of a snapshot of some of the interesting avenues of research being explored. Highly supervised systems, regardless of their performance, have not been included, nor have visualisation techniques that make it easier for human experts to discover grammar structure from text (Belkin and Goldsmith, 2002; Elliott et al, 2001). We also exclude systems which only infer word-classifications without attempting to learn structure, such as (Atwell, 1983; Hughes and Atwell, 1994; Roberts, 2002).

1.2 Categorial Grammar

A categorial grammar is a simply a grammar rather than a learning paradigm. However, CG clearly lends itself to unsupervised learning as it has been adopted as the foundation for many systems. One of the main reasons for this that the lexicon and the grammar are acquired in the same task; the psychology literature (Bates and Goodman, 1997) suggests that the two are not separate mental processes. This is a bonus for those researchers striving to produce a realistic psychological model of human language acquisition, and also for those who wish to implement simpler and more efficient algorithms for what is still a complex task.

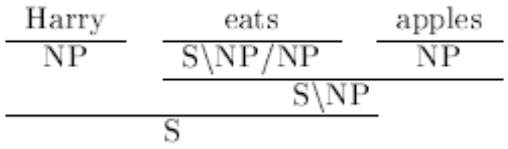

Figure 1.1: A simple sentence parsed in CG

CGs were first proposed by Ajdukiewicz (1935) and have since matured and modified. Steedman's generalised version (Steedman, 1989) serves well for a brief overview. A CG comprises of two components. Firstly, the categorical lexicon is essentially a dictionary which associates each word within the lexicon a syntactic and semantic category. Secondly, the combinatory rules provide the functional application of the grammar, and allow more complex categories to be created from the simpler ones.

$$S/b \longrightarrow a$$
$$S\backslash a \longrightarrow b$$

These operators provide the freedom to transform rules, allowing you to isolate and manipulate parts that would otherwise be inaccessible (Adriaans, 1999). Fig. 1.1, taken from Steedman (1989), illustrates how a simple sentence is parsed.

1.2.1 GraSp

GraSp (Henrichsen, 2002) is a learning algorithm designed specifically for inducing grammars from large, unlabelled corpora. Its long term goal is to provide insight to the innateness debate. In this instance, the hypothesis is that there is no such linguistic innateness.

Henrichsen used a variant of the Gentzen-Lambek categorial grammar, which was enhanced with non-classical rules for isolating a residue of uninterpretable sequent elements. Empty categories are also permitted in this version which are not normally allowed in CG due to the principle of adjacency: combinatory rules may only apply to entities which are linguistically realised and adjacent (Steedman, 1989).

The learning algorithm begins by assigning each word type with its own unique category. The learning process applies changes in the lexicon by adding, removing and manipulating the basic categories using the CG operators (/, n or *). The changes are guided by a measure of disorder. Dis(§) returns the number of uninterpretable atoms in the sequent §. The update process is iterative. GraSp monitors the measure of disorder before applying each update, and the process will halt as when the update no longer improves the disorder of the lexicon.

No quantifiable measurements of GraSp's accuracy were published. Therefore, commenting on its performance is obviously difficult. Whilst rigorous evaluation may not have taken place, GraSp clearly has many merits in that it does succeed in learning linguistic features from unlabelled corpora. Henrichsen describes the output being rich in “microparadigms and microstructure", which inter-connect to form a complex grammar.

1.2.2 CLL

CLL (Watkinson and Manandhar, 2001) is not only concerned with developing a computationally feasible language learner, but one that is also psychologically plausible too. Therefore, the algorithm used by CLL was designed to also make way for a model of human language learning facilities, as well as being a computational learning tool.

CLL is trying to emulate a child with respect to its acquisition of its first language. The psychology influence in the research refers mainly to the environment in which the learner learns. This deals with the type of language a child is likely to encounter and the effect of language teaching. The conclusion reached: “Hence, we have a learner that is unsupervised, positive only and does not have a teacher." Unfortunately, the algorithm was arguably built with too much `innateness', which reduces its credibility as an unsupervised process. The provision of a complete set of lexical categories was quite justly acknowledged by the authors as being “too strong a bias to be psychologically plausible". Additionally, the algorithm is given a set of closed-class words (with categories) at the start of the learning process. Two different sizes of the initial lexicon were tried, 31 and 348.

The learning algorithm functions by taking an example sentence from a corpus, which is then parsed using a n-best probabilistic chart parser (developed from a standard stochastic CKY algorithm). This can result in a number of possible parses, of which it is then up to the parse selector to decide which one would benefit the lexicon the most. The metric which decides the `goodness' of a parse is based on which creates the most compressive lexicon. To do this, it must also evaluate the effect of the newly modified lexicon by reparsing any examples that may be affected. Whilst it appears to be a costly approach, it does ensure the most compressive lexicon.

Watkinson and Manandbar created a relatively robust approach to evaluating their results (Watkinson and Manandhar, 2001). As the Penn Treebank corpus was the source of the text to learn, its annotation was translated into CG annotation, so that it could be compared with the output of the learning algorithm. Whilst the newly annotated corpus was considered a gold standard, it was converted automatically, and therefore liable to error. The best performance attained by CLL was 51.9%

accuracy (this is with an initial 348 word lexicon). While this performance is still relatively low, considering the difficulty of the problem, and using a complex corpus, to perform above 50% is still a recognisable achievement.

1.2.3 EMILE

EMILE (Adriaans, 1992, 1999) has been around for some time now. It will continue to be with us because it is a well executed algorithm and performs well and efficiently. EMILE has been updated through the years, and is currently at version 4.1, although this latest version has been implemented by Vervoort (2000).

For EMILE to be in this section, it clearly relies on a categorial grammar. A given input set of example sentences is converted into a CG of basic categories. After applying first order explosion, each sentence is examined to discover how it can be broken up into subexpressions (using the standard CG operators). The resulting set of subexpressions is passed to an oracle. The reason for this is because EMILE uses a teacher/child metaphor. Therefore, the system can ask the oracle which ones are valid. Any subexpressions that can be substituted into the same contexts, and still be valid are said to be of the same type. Therefore, the next step employed is to cluster to rules passed by the oracle and cluster them into types. The final phase is rule induction. The clustering has resulted in a variety of basic and complex rules, however, they tend to relate to specific types. Thus the rule induction step generalises them to general types, with the outcome being a shallow context-free grammar.

EMILE tends to produce accurate results due to the fact that it waits for enough evidence to be found before constructing grammar rules. However, according to Adriaans' calculations, in order for his system to acquire a language with 50,000 words, it would need learn from a sample of 5 million sentences. Assuming an average of 15 words per sentence, then a 75 million word corpus is required. My initial thoughts was that figure was too large. However, it is quite reasonable considering EMILE is generating a grammar to cope with 50,000 words considering the complexity of the task. It does mean that EMILE is a slow learner (compared to some other systems, such as ABL), as was concluded in van Zaanen and Adriaans (2001). Experiments conducted on the ATIS corpus produced precision of 51.6%.

1.3 Memory Based Learning (MBL)

The MBL paradigm attempts to discover ways in which you can abstract information from a given data set, whilst maintaining accuracy. The hope is to develop methods that perform at least equal to pure MBL (where all information is retained).

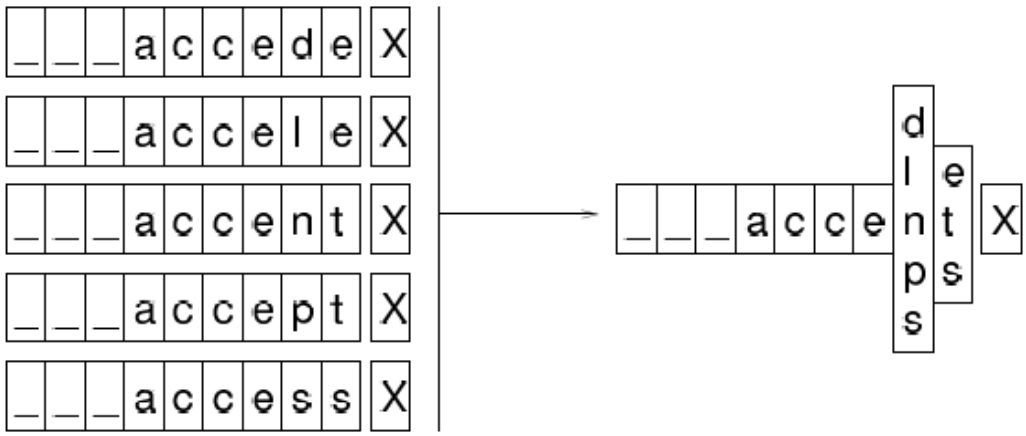

Figure 1.2: Example of family creation in FAMBL taken from Van den Bosch (1999). This shows instances of grapheme-phoneme occurrences being merged into a single family expression.

Pure MBL tends to give the best performance in analysis of unseen data, due to the very fact that it never forgets any training examples. An advantage of computerised systems is that it is often feasible to keep every instance - memory limits are seemingly infinite. So, why bother with MBL? Abstraction should result is smaller and more efficient learning models. After all, humans do not have a capacity for pure MBL, therefore, MBL should offer a more psychologically plausible approach too.

1.3.1 FAMBL (Family Based Learning)

Classification systems that are equipped with a forgetting facility, use it to not only perform more efficiently, but to avoid over-fitting, which can be detrimental to accuracy. However, default parameters for this process often generalise too much for learning tasks which produces poor performance (Van den Bosch, 1999).

Therefore, the key is to use careful (or weak) abstraction, whereby instances can be abstracted without doing harm (e.g., forgetting exceptions would be careless (Daelemans et al, 1999)). The way in which FAMBL achieves this is to transform the instance base into instance families. A family is a cluster that has been classified using k-NN (Nearest Neighbour). The instances surround a given instance are its family. Fig. 1.2 gives an example of how the instances are then merged into hyper-rectangles that define a family expression.

The FAMBL algorithm randomly selects instances individually, that are not a member of a family. For each one found, its family is determined and a family expression is created from those instances. It continues to generate families until the instance base doesn't contain any instances that do not belong to a family.

FAMBL has not yet been used as a full-scale grammar induction system. It has, however, been applied successfully to a variety of relevant tasks in language learning, including morphological segmentation, base-NP chunking, PP attachment, and POS tagging. For example, the experimentation with POS resulted in an accuracy of 97.87%, and family-abstraction yielded a reduction of 75% memory compared to pure-MBL.

1.3.2 RISE (Rule Induction from a Set of Exemplars)

RISE (Domingos, 1995) is a multi-strategy approach, which comprises of MBL and rule induction. Rules begin as being instance specific. It then begins to generalise by looking at each rule and searching for other instances that fall within the same class. Any instances that satisfy this are merged and their rules are generalised.

In order to ensure the rules deduced are productive, RISE estimates the `goodness' by computing its apparent accuracy using its class prediction strength with Laplace correction. Performance is constantly monitored by the algorithm and generalisation ceases if the apparent accuracy worsens.

1.4 Artificial Life: Evolutionary Optimisation

Nature has allowed humans to acquire the abilities to learn, understand and communicate using language by process of evolution. With that precedent, it should therefore by possible for us to apply similar techniques to create Language Acquisition Devices: LADs. Of course, its feasibility is the matter of debate.

LADs are already complicated, but adding an extra discipline of modelling evolution brings a new dimension of difficulty. The payoff is that once the system is set up, natural selection will take over and allow optimal language learning conditions to emerge.

1.4.1 Iterated Learning Model (ILM)

The idea behind Kirby's work is to take away the emphasis of biological evolution and he believes too much importance has been placed upon it. The alternative is to treat languages as adaptively evolving systems (Kirby, 2002).

If language is like an organism in its own right, then you begin to see that it has its own set of selected pressures. Humans are its host, and its method of transmission is via human communication. A successful language is one that can be learnt, understood and used, for the benefit of its hosts.

Therefore, Kirby and Hurford (2002) suggests that language is not only subject to biological natural selection, but is the result of three complex adaptive systems, as illustrated in fig. 1.3. There clearly are interactions: “for example, biological evolution provides the platform on which learning takes place, what can be learnt influences the languages that can persist through cultural evolution, and the structure of the language of a community will influence the selection pressures on the evolving language users." (ibid)

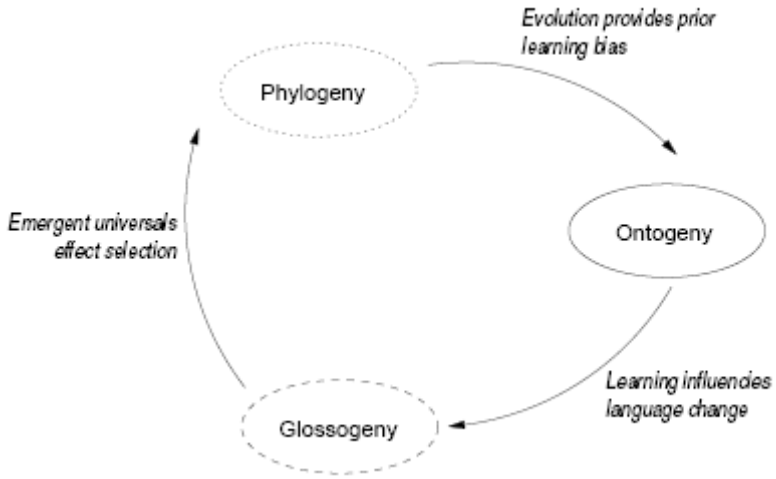

Figure 1.3: Interactions of the three adaptive systems.

ILM is composed of four elements:

1. A meaning space
2. A signal space
3. One or more language-learning agents
4. One or more language-using adult agents

The adult agent as a set of meanings for which is must convert into signals to then be transmitted to the learner (both agents have been initialised with random parameters). The speaker makes an assumption that the hearers signal-to-meaning mapping will be approximately that of the speakers. Therefore, the principle behind transmission is the generate signals that maximise the confidence of the given meaning. Once the speaker has finished, it is then redundant, and the learner is promoted. A new learner is added and the cycle continues.

This is a somewhat simplified version of events, but the overall result is after a few hundred generations (after much randomness) meaningful structure emerges.

Kirby's research on the whole is looking at linguistic evolution rather than creating a tool for grammar inference (although one does lead to the other). And as such, there are no large scale runs of using the ILM to acquire grammar from a large corpus, and to evaluate it in any way. However, its inclusion is still relevant because

it is an interesting approach - one that could be adapted to evolve grammar inference devices.

1.4.2 Language Agents (LAgts)

Language agents (LAgts) are born from the research carried out by Briscoe (2000). An equally intriguing approach to grammar induction through the application of alife, populations of LAgts are simulated. LAgts are language learners, generators and parsers.

Each LAgt adopts a Categorial Grammar as its underlying framework, albeit an extended version known as Generalized Categorial Grammar (GCG), (Wood, 1993). They are used to provide a relationship between the LAgt's universal grammar and the specification of a given grammar; Briscoe is clear which side of the 'innateness' fence he stands on, which is why LAgts possess a universal grammar. They are embedded in a default inheritance network, and are represented as a sequence of p-settings. Each setting can be encoded as True, False or ? (which indicates that it is yet to be specified). They are partially ordered too, which means atomic categories can be in an arbitrary position, although more complex types, where directionality is significant, then ordering must be preserved within that type. There is also a distinction between Absolute, Default and Unset parameters.

Simulations are run to model the evolving population of LAgts. A successful interaction is generally one where a random agent generates a sentence based on its current grammar, which can be parsed by another randomly selected agent. Such a pair are said to have compatible p-settings. Each LAgt has a set lifespan of 10 interaction cycles. Between the ages of 4-10, an agent can reproduce new agents, and from aged 5 onwards, an agent stops learning and its grammar is therefore fixed. Agents older than 10 are removed from the simulation. During their learning phase, it is possible for LAgts to alter any of their parameters that were given Default of Unset attributes from their conception. There is a cost associated with an update, which is why successful agents tend to changed by one point of their initial p-settings. This results in the classical strategy of inheritance where parents pass on their genes and not any acquired characteristics.

Once again, performance related information about the LAgts' actual abilities to acquire grammar accurately is difficult to extract. Much of Briscoe's work has

been to experiment with the seemingly unlimited number of factors that affect evolving systems (coevolution, migration, acquisition effectiveness etc). However, languages did emerge from the learners that were described as `full language': that is, `close to an attested language'. Ideally, Briscoe would have elaborated as to just how `close' this is. There were seven full languages available for experimentation. A language is given to one or more adult LAgts (depending on the simulation) who then communicate with the learners, and so on.

1.5 String Pattern Searches

Some systems do not fit conveniently into the above groups, and illustrate the greater breadth of approaches to UGI.

1.5.1 ABL (Alignment Based Learning)

ABL (van Zaanen, 2002) is a learning paradigm in its own right. It is based on the principle of substitutability, whereby two constituents are of the same type, then they could be substituted. Of course, the system is unsupervised, and therefore, does not know types. Thus, the principle is reversed so that if two constituents can be substituted, they are of the same type. Fig. 1.4 shows an example from van Zaanen and Adriaans (2001) of two segments from two sentences are declared as being of the same type.

$$\frac{\begin{array}{l}\underline{\textit{What is}}\ \textit{a family fare} \\ \underline{\textit{What is}}\ \textit{the payload of an African Swallow}\end{array}}{\begin{array}{l}\textit{What is (a family fare)}x \\ \textit{What is (the payload of an African Swallow)}x\end{array}}$$

Figure 1.4: Example of bootstrapping structure in ABL

Much complexity is added due to alignment learning phase finding constituents that overlap each other. This problem is overcome using a selection learning phase. This is where the ABL algorithm determines the correct (or at least the best fit) constituent using probabilistic methods. Selection is decided upon calculating the probability of the words in the overlapping constituent and its type. A Viterbi-style algorithm is also used to search through all possible combinations of overlapping constituents and select the best one. ABL performs well. The computational

demands of its algorithm mean that it is not suitable for large corpora (>100K sentences). However, its greedy nature means it will learn quickly. An accuracy of 62% was recorded when ABL was given the OVIS corpus to process. Versions of the ABL system have also been tested with the Penn Treebank and ATIS corpora.

1.5.2 GB (Grammatical Bigrams)

GB or Grammatical Bigrams were proposed by Paskin (Paskin, 2001), in the hope of creating a simple language learning model, and therefore, making the actual learning process tractable. Independence assumptions are introduced to reduce complexity, although at the same time it increases the model's bias.

The Grammatical Bigram model uses the Dependency Grammar formalism to describe the relationship between pairs of words. One word in this link is the head, and the other is its dependent. A dependency parse is a directed graph, consisting of a set of such relationships. No word can be dependent on more than one head (the root of a sentence has no dependency). Also, a word cannot be dependent on itself, making the links acyclic. If the dependents of a head word are completely independent of each other, and their order, then this independence assumption results in a much simpler model of grammar and so the parser is spared of that complexity.

The parser is used to learn grammar from labelled corpora. However, for unsupervised learning, the EM algorithm is used to learn the optimal parameters (probabilities of dependency for a given head). Other statistics are computed using an adapted version of the Inside-Outside algorithm that works in O(n3) time.

Unfortunately, Grammatical Bigrams are suited more to the generalisation of labelled data than to unsupervised induction. When given an unlabelled corpus of Wall Street Journal articles, and its output evaluated against the annotated Wall Street Journal section of the Penn Treebank, GB yielded an accuracy of only 39.7%. Clearly, the compromise in making the model computationally efficient results in a grammar model that is still too approximate to represent the sorts of structures it sees in the input corpus.

1.6 Other approaches

1.6.1 PCFG-BCL

Tu and Honavar (Tu, and Honavar, 2008) present an approach called PCFG-BCL which roughly stands for Probabilistic Context-Free Grammar [using iterative] Biclustering. Context-free grammars with attached probabilities are not described the traditional Chomsky Normal Form, but in equivalent AND-OR terms instead. The notation is similar and is easily demonstrated in Figure 1.6.

CNF	**AND-OR Form**
$S \rightarrow a\ (0.4) \mid AB\ (0.6)$	$OR_S \rightarrow a\ (0.4) \mid AND_{AB}\ (0.6)$
$A \rightarrow a\ (1.0)$	$AND_{AB} \rightarrow OR_A OR_B$
$B \rightarrow b_1\ (0.2) \mid b_2\ (0.5) \mid b_3\ (0.3)$	$OR_A \rightarrow a\ (1.0)$
	$OR_B \rightarrow b_1\ (0.2) \mid b_2\ (0.5) \mid b_3\ (0.3)$

Figure 1.6: Comparison of CNF and AND-OR Form.

The advantage of this notation for the authors is that it allows them to focus on the AND-OR groups, that are the rules that describe the higher-level rules, and they are always in the form of AND $\rightarrow OR_A OR_B$, where an AND joins two ORs.

Like most grammar inference algorithms, the process begins with a grammar consisting entirely of terminals as given in the input corpus. The system works in an iterative fashion to find abstractions within the input, and thus find the rules. New symbols are found, which are then compressed by searching for possible ORs to bind to ANDs, and so on until no new rules are found.

The method employed to acquire the new groups is called “biclustering” (Madeira and Oliveira, 2004). If the corpus is represented as an NxN matrix where the rows and columns represent the terminals, then a bicluster is simply a sub-matrix within. The matrix cells are populated with straight-forward bigram counts. When biclusters are found to be multiplicatively coherent, they are inferred as AND-OR groups and therefore added to the grammar.

After each group is extracted, a metric is applied to determine the benefit to the grammar with the newly acquired group. A fairly complex calculation to find the Liklihood Posterior Gain (LPG) is made and if an improvement is shown then the rule is formalised. It is this calculation that steers the iterative nature of the approach.

Experiments were performed on fairly small sample corpora and compared against rival systems EMILE and ADIOS. Performance was evaluated by comparing output against the reference grammar of the corpus, and on average performed 1.5% better than its rivals, with f-scores ranging from 77%-100% depending on the input.

1.6.2 ADIOS

The Automatic Distillation of Structures system (Solan, 2006; Solan et al, 2005) relies on two processes to acquire a context-free grammar: a specialised pattern extraction algorithm and a generalisation process.

Sentences from a corpus are modelled as paths in a graph structure, where vertices are the constituent words. After each sentence is loaded the system looks for "significant patterns" which will then be abstracted as terminals. The patterns are determined following principles influenced by Harris's substitutions (1954) and van Zaanen's alignment-based learning (2000), as illustrated in Fig 1.7. From this patterns and equivalence classes can be deduced and a CFG constructed.

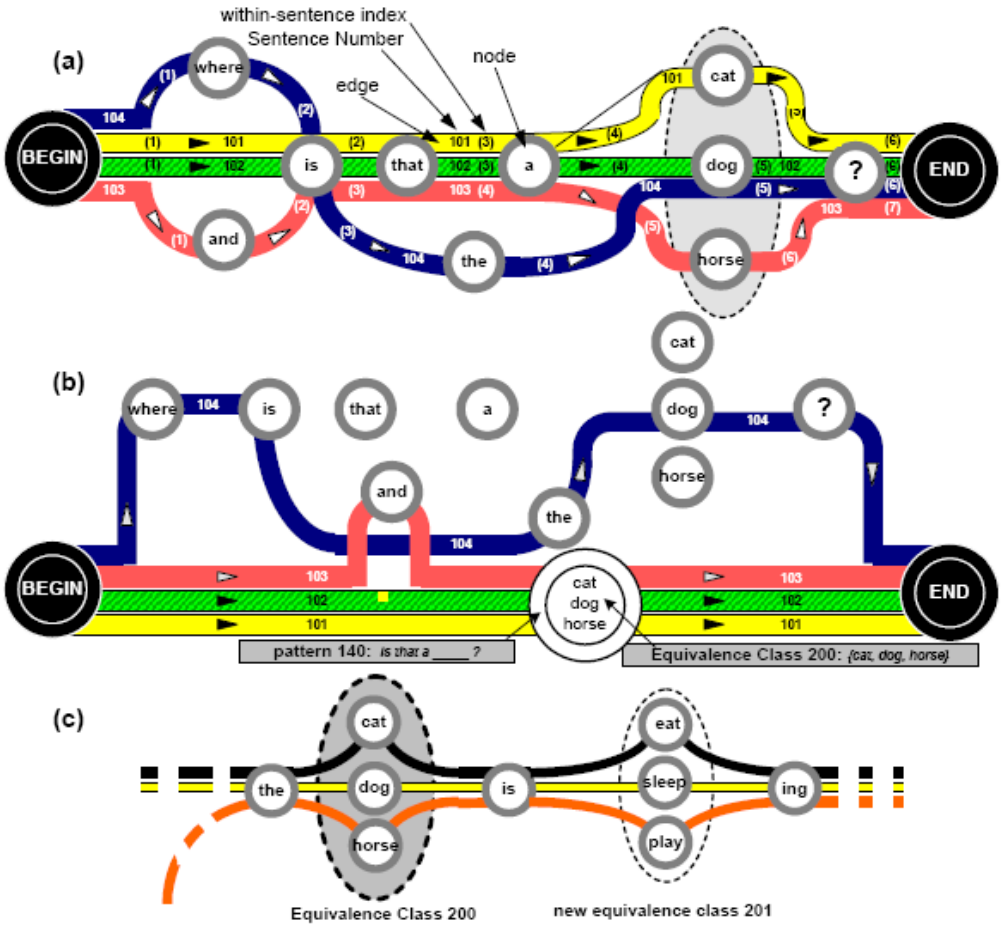

Figure 1.7: example of ADIOS pattern extraction and equivalence classes being found. (Horn et al, 2004)

When evaluating against an artificial corpus of 6400 tokens it achieved a precision of 95% and a recall of 86%. Interestingly this result outperformed smaller corpora that were also tested, which is indicative of the need to work on larger, more substantial collections of sentences in order to generate more reliable grammars.

The algorithm was also tested on 1.3million word sample from the CHILDES corpus (spoken utterances by, or directed to, youngsters). On this much larger, more realistic (and therefore more susceptible to “noise”) performance was significantly lower, yet still reached levels of 60% accuracy.

ADIOS was also tested on a parallel corpus of the Bible and a comparison study of the resulting grammars was made. Intuitive generalisations were confirmed by the output, as Horn et al (2004) noted:

"There are two observations that are very obvious in these results. First, clearly Chinese is very different from the five European languages. Second, the histograms peak at collocation structures of two and three words, TT and TTT. Closer scrutiny of similarities between the grammars of different languages, as described by inner products of the relevant vectors of ADIOS *spectra, leads to the conclusion that Spanish is closest to French and Danish is closest to Swedish. All these observations may come as no surprise, but it is satisfying to see them emerging from this system."*

1.6.3 Hierarchical Dirichlet Processes

Antoniak (1974) describes Dirichlet process mixture models, which are essentially an extension to Bayesian finite mixture models, where the finite limit is lifted. Liang et al (2007) use this, whilst employing a "stick-breaking" representation (Sethuraman, 1994) which a very robust method for determining the number of terminals that the grammar should be broken into.

Their algorithm generates probabilistic context-free grammars from an input corpus. Experiments were run with an artificial corpus and also on a section of the Penn Treebank corpus. Performance results were not given as the authors were focussing on evaluating grammar size and were anxious to steer their system away from over-fitting.

1.6.4 Unsupervised DOP

Data Orientated Parsing (DOP) has been researched by Bod since 1998 – DOP parses new sentences by taking all subtrees, regardless of size, of an annotated training corpus. In recent years, thanks to advances in unsupervised learning techniques, DOP has been extended to incorporate such techniques into an approach labelled Unsupervised DOP, or U-DOP (Bod, 2006). The system is pragmatic in its approach as there are many computational limitations and reasonable compromises have been sought to ensure a tractable solution.

U-DOP begins by taking sentences and representing them as binary trees. There are many potential permutations, and therefore to keep computing requirements realistic, only a subset, albeit relatively large one, is considered, and a

Viterbi *n*-best parsing measurement is applied to each one to calculate the most likely binary tree.

U-DOP was extended further into UML-DOP where probabilities are evaluated with a maximum likelihood re-estimation technique based on the expectation-maximization algorithm (Dempster et al, 1977). The reason being was that the relative probabilities were not statistically consistent, where as the ML estimation was, and has been found to increase accuracy, as well as capturing grammars with fewer rules than its U-DOP counterpart.

Due to DOP's all-subtrees approach, it permits a rich language model that can capture both contiguous and non-contiguous lexical dependencies that are essential when inferring grammar in real-world corpora. And the results are indeed very promising. Experiments on a range corpora versus a number of competing approaches were conducted. The corpora were filtered to only include sentences of up to 10 words, and included the Wall Street Journal corpus, the German NEGRA corpus, and the Chinese CTB corpus. UML-DOP had f-scores of 82.9, 67.0 and 47.2 respectively. An additional test was done on another WSJ sample consisting of sentences up to 40 words, and this saw f-score performance drop to 66.4.

Interestingly Bod acknowledges the complexity of evaluating inferred grammars and notes that the Penn Treebank – from which the WSJ corpus is a component – is very flat and therefore penalises unsupervised systems that are inclined to find deeper parse trees.

1.6.5 LARS

Eyraud et al (2007) discuss the issues facing those trying to automatically acquiring more complex grammars as one moves up the Chomsky hierarchy. The context-free class is theoretically not learnable. The authors therefore try to reformulate the problem in a different way which is to take a representation known as String-Rewriting Systems (SRS), and like many grammars allow symbols to be composed of terminals. For example, the string 'abaabbab' can be rewritten using the substring 'ab':

abaabbab → abaabb → ababb → abab → ab → λ

Of course, this raises issues because it could potentially substitute in undesirable places and therefore a enhancement to SRS is proposed called Delimited

SRS that augments the rules to allow rewrites to be specified at the beginning, end or middle.

With the language representation understood, the induction process can commence. LARS stands for Learning Algorithm for Rewriting Systems. The LARS system is a simple greedy algorithm that seeks out repetitions that it classes as substring.

Initially Lars computes all the substrings from the input. Left and right-hand sides of the rules will be chosen in this set Then Lars enumerates the elements of this set to build the candidate rules. Several checks are performed to validate any candidate rule. Firstly is verification that the candidate left-hand side is a substring of at least one string in the current inferred grammar. This is a precaution to discard rules that cannot be used to rewrite at least one string in the system, and therefore reduces the chances of introducing false rewrites. This is followed with a check that the new rule is also consistent with the definitions of DSRS. Finally a data consistency check is done to ensure that the rule generate true strings as defined in the grammar.

Results in terms of typical Treebank comparison f-scores were not given as authors preferred to prioritise on special small-scale samples. That said, trials were done on the Omphalos grammar inference competition training sets but unfortunately results were very poor due to the greedy nature of the core algorithm. The authors sum up:

"Essentially Lars is provable: it makes its decisions (to create rules) on the basis of the fact that there is no reason not to create the rule. A more pragmatic view is to base such a decision on the fact that it seems a good idea to do so. But convergence, in the second setting, needs a statistical decision, and changes quite drastically the type of results one can achieve."

1.7 Grammatical Inference of Word Classes

Kiss (1972), in the context of understanding the psychology of learning, studied the order of acquisition of words in children. He showed that it was statistically possible to distinguish different word classes based on their

distributional properties. His approach took 30 words from language found in children's literature and clustered them based on their similarity to their nearest neighbour bigram counts. An *n*-gram is a sequence of *n* successive objects; in this instance, the objects are words, and *bi* represents $n = 2$. He was restricted to only being able to classify few words.

Baker (1975, 1979) published a model which originally intended as a procedure for automatic training of speech recognition systems. However, it happened that it could be also used for the purpose of tagging parts-of-speech. Assuming that a language could be generated by a Markov process, then he proposed a technique to automatically calculate the parameters of a Markov model which was compatible with the data. Unfortunately, Baker's recursive formulae for estimating the parameters of a Markov model was computationally expensive which prevented any practical application (Atwell 1987).

Atwell (1987) attempted to apply the theory put forward by Baker (1975, 1979). He had to place considerable constraints in order to make the computational demands more reasonable; for example, the assumption that a word can only belong to a single word class. Also, he only considered words located immediately before and after each designated word (although this was performed separately because processing was still very slow). Despite some promising outcomes, Atwell acknowledged that his sample was too small to provide conclusive results (at the time, using a sample of 200,000 words, the program took several weeks on a mainframe computer to complete the task), and that the constraint on single word class participation would need to be removed for the system to be wholly successful.

Hughes (1994), (Hughes and Atwell 1994) performed many experiments in order to find which factors give the best accuracy when automatically determining parts-of-speech. The variables were:

1. The contextual pattern, e.g., sentence position distribution; various sized bigrams.

2. The metric used to calculate the distance between words in vector space.

3. The clustering method, e.g., Single linkage; Centre of gravity; Ward's method etc.

4. The size of the comparison set.

With a 35 million word corpus available, generated from extracting USENET articles, the 200 most frequent words were used in the experiments, which were adequate to evaluate the most effective factors. Once the best combination had been established, he clustered the top 2000 occurring words from his corpus with a high degree of success: 87% accuracy if the words are classified into 100 clusters.

Hughes found that the distributional redundancy of word types was revealed using bigrams rather than absolute sentence position. And that the context of absolute sentence position distribution was contained within the context of bigram counts.

Redington *et al*. (1998) undertook research into automatically acquiring syntactic categories in the psychological context of attempting to understand the processes of how children learn language showed good success. A particularly relevant experiment performed was clustering a set of words with the function words removed. The reasoning for the test was that child speech is sparse in function words in comparison the proportion of content words they use. If they are effectively only concentrating on content words, then a child's mechanism for acquiring word classes does not require function words. That specific experiment did show that word classification was in fact possible when function words were excluded from the

input stream. However, they did acknowledge that removal of the function words did have a “considerable impact” on the informativeness of the results.

An alternative to Grammatical Inference of word classes based on word-bigram patterns is automatic acquisition of word classification using distribution analysis of content words with respect to function words; in chapters 4 and 5, I apply this approach to English, Spanish and Arabic corpora.

1.8 Discussion

This review has attempted to analyse the range of underlying algorithms used by various approaches:

1. GraSp (Henrichsen, 2002): minimising “disorder" in CG lexicon;

2. CLL (Watkinson and Manandhar, 2001) : stochastic CKY parser optimisation from a given lexicon;

3. EMILE (Adriaans, 1999): rule induction from candidates, guided by an “oracle";

4. FAMBL (Daelemans et al, 1999): weak abstraction of common subexpressions into “families";

5. RISE (Domingos, 1995): Memory Based Learning or patterns, with rule induction;

6. ILM (Kirby and Hurford, 2002): evolutionary optimisation of a language-space;

7. LAgts (Briscoe, 2000): evolutionary optimisation of language-acquisition software agents;

8. ABL (van Zaanen, 2002): rule induction from substitutable subexpressions;

9. GB (Paskin, 2001): stochastic optimisation of dependency-pair sets;

10. PCFG-BCL (Tu and Honavar, 2008): probabilistic context-free grammar induction using iterative biclustering;

11. ADIOS (Solan, 2006): automatic distillation of structures by graph-based pattern extraction and generalisation;

12. Dirichlet process models (Liang et al, 2007): hierarchical extended Bayesian finite mixture models;

13. U-DOP (Bod, 2006): unsupervised learning using data-oriented parsing metrics for subtree evaluation;

14. LARS (Eyraud et al, 2007): learning algorithm for delimited string rewriting systems;

15. Grammatical Inference of word-classes by clustering of word-bigrams (Kiss, 1972), (Baker, 1979), (Atwell, 1987), (Hughes and Atwell, 1994), (Reddington et al, 1998).

A review ought to include a comparative evaluation of the alternative approaches and systems; but we could find few objective metrics or evaluative features reported across the source literature. Most researchers appeal to a linguist's “looks good to me" evaluation (Hughes and Atwell, 1994), (Jurafsky and Martin, 2000): they demonstrate that their systems can infer some examples of grammatical constructs which seem similar those found in linguists' grammars. Unfortunately,

this is a subjective qualitative assessment, and does not yield a percentage score which can be compared.

Early research on unsupervised word-class inference, clustering words into classes e.g. (Atwell, 1987; Atwell and Drakos, 1987; Finch, 1993) also appealed to “looks good to me" evaluation; but more recent research has tried to measure inferred word-classes against a human-tagged corpus (Hughes and Atwell, 1994; Roberts, 2002). Other areas of Language Engineering also try to evaluate rival systems against a “Gold Standard" human-annotated corpus; so why not Unsupervised Grammar Inference? Only four of the projects surveyed report accuracy measured against a human-parsed corpus:

1. CCL (Watkinson and Manandhar, 2001): 51.9% on Penn Treebank;

2. Grammatical Bigrams (Paskin, 2001): 39.7% on Penn Treebank;

3. EMILE (Adriaans, 1992; Vervoort, 2000): 51.6% on ATIS Corpus;

4. ABL (van Zaanen, 2002): 62% on OVIS corpus (lower with Penn and ATIS).

Even these percentage scores cannot be compared meaningfully as they are based on different alignment-measures, and used different corpora and human parsing schemes. Parsing schemes used in human-annotated treebanks can capture a variety of grammatical information, including some or all of the following (Leech et al, 1996): (a) Bracketing of segments; (b) Labelling of segments; (c) Showing dependency relations; (d) Indicating functional labels; (e) Marking sub-classification of syntactic segments; (f) Deep or `logical' information; (g) Information about the rank of a syntactic unit; (h) Special syntactic characteristics of spoken language. In their review of corpus parsing schemes, Atwell et al (2000) conclude:

“unlike the tagging schemes, it does not make sense to make an application-independent comparative evaluation. No single standard can be applied to all parsing projects. Even the presumed lowest common denominator, bracketing, is rejected by

some corpus linguists and dependency grammarians. The guiding factor in what is included in a parsing scheme appears to be the author's theoretical persuasion or the application they have in mind."

So, an objective measure of alignment against a human-parsed “gold standard" Treebank may not be feasible or even desirable. In fact, one intriguing potential of Unsupervised Grammar Inference is that it may yield analyses which fit the data better than traditional grammarians' parse-categories; but if we measure against an established Treebank, any such innovation will be penalised. Unsupervised Grammar Inference has potential applications with unknown languages e.g. (Atwell and Elliott, 2001), for which a high score in learning English grammar may be inappropriate.

An alternative possible metric which suggests itself is “how much has been learnt": some measure of the difference between size or scale of the initial assumption or “learning bias" (Jurafsky and Martin, 2000) and the final grammar which has been inferred. For example, Watkinson and Manandhar (2001) contrasted CCL experiments with initial lexicons of 31 and 348 words, and found that the latter yielded a larger grammar. Unfortunately, other sources did not report comparable “starting assumption" and “final grammar" metrics.

1.9 Future of Grammar Inference

The next big step within the UGI community is to design and develop a robust evaluation procedure. Many of the systems featured in this survey did not publish performance results of their experiments, favouring a `looks good to me' approach. This is certainly not an attempt to say expert linguistic evaluation is somehow inferior to an automatic, computerised approach. However, it is rather subjective, and if carried out by the author of the UGI system, likely to be partial to some bias.

An automatic evaluation tool - if designed correctly - would allow consistent comparison between rival systems. To be able to quantify performance would allow GI designers and developers to ensure that future updates actually provide greater accuracy, and quickly, so that research is not led down a dead end if the results of a system looked good, but performance was in fact degrading. However, it is clearly fraught with difficulties, which is the likely reason why many have steered clear.

With respect to UGI itself, we can look forward to great advances in the long term. The computational complexity of the algorithms will become less of a burden with optimisation and increased computational resources. We also look forward to wider applications on different datasets, as Unsupervised Grammar Inference is more widely recognised as a powerful data-mining technique.

Chapter 2:
Current research topics in Corpus Linguistics

This thesis looks at ways to bring together the two research fields of Grammatical Inference and Corpus Linguistics; so this chapter investigates the question "To what extent has Grammatical Inference been used in Corpus Linguistics, and in what areas should it be applied?" A good place to start to answer this question is to review current research topics covered in the International Conference on Corpus Linguistics 2003, where I presented my own research reported in Chapter 3 (see figure 2.1). After the initial success of the first Corpus Linguistics conference in 2001 in honour of Geoffrey Leech's 65th birthday (Rayson et al 2001), Tony McEnery *et al*. decided to make it a more permanent fixture in the linguistics calendar, and stage the conference every two years. Hence, it was time for the sequel to commence, and as a result saw about two hundred corpus linguists (representing 30 countries) flocking to Lancaster.

Around this time two years ago, Geoffrey Sampson wrote an article in ELSNews regarding the future role of ICAME (in particular, their conferences). He remarked that ICAME's focus on English research, coupled with its restrictions on conference sizes to ensure a *friendly* atmosphere, means that not only does it miss out on the ever increasing (and exciting) shift towards non-English language research, but also the next generation of researchers, with energy and fresh ideas, but who are not yet established enough to join the ICAME clique. He went on to comment about the success of CL2001 that had taken place shortly before, and predicted success for its future. ICAME's weaknesses are CL's strengths, which is why Geoffrey's prediction was correct.

The conference kicked off with a day of workshops covering development of learner corpora and multilingual corpora, corpus-based approaches to figurative

language, and shallow processing of large corpora. The SProLaC workshop issued its own proceedings (Simov and Osenova 2003); other papers appeared in the main CL2003 proceedings (Archer et al 2003). The following four days saw papers presented in three parallel sessions. With approximately 95 papers and 30 posters on offer, it would be impractical to go into any real depth about them. Needless to say, the coverage of the field was well represented. New resources and research were being developed for European minority languages and South-East Asian languages. The usual suspects, such as tagging, parsing, disambiguation, grammars and information extraction were well covered. Unsurprisingly, there was also a strong focus on corpus development, annotation and tools. Even if translation studies, exploiting corpora and semantics are added to the list, it is still not exhaustive – which simply illustrates the breadth of the conference. Of course, the fact that each piece of research presented by definition is linked to corpus, means that there was this common thread throughout, and so as varied as the topics were, they never felt disjointed.

Invited speakers were Michael Hoey, Nancy Ide, Susan Hunston, Geoffrey Sampson and Nicoletta Calzolari. All are well known within the field and offered fascinating and well received talks. Probably the most infamous presentation of the conference was (McEnery and Xiao 2003), “Fuck revisited”, covering studies of the f-word within the BNC. It was even heard that Eric Atwell, a speaker in a parallel session, was actually recommending to his audience to go and see Tony’s instead as it was “more interesting!”

The conference covered a broad, representative range of Corpus Linguistics research topics; so it was striking that very few papers applied or even mentioned Grammatical Inference. Apart from my own presentation, the only papers involving Grammatical Inference were (Atwell 2003) on a new Machine Learning algorithm for neoposy: coining new Parts of Speech; and (Gronqvist and Gunnarsson 2003) on a method for finding word clusters in spoken language. However both of these papers focussed on Machine learning algorithms, rather than applications of GI in

Corpus Linguistics research. It seems that Corpus Linguistics researchers are unaware of Grammatical Inference as a potential research tool.

Further evidence for this conclusion comes from other research publication sources in the Corpus Linguistics field: proceedings of TALC, Teaching and Language Corpora; and ICAME, the International Computer Archive of Modern and medieval English. I reviewed the online Proceedings for TALC 1994, 1996, 2000, 2004, 2006, 2008; but I found no papers on Grammatical Inference applied to Corpus Linguistics research. I also reviewed the online ICAME Journal, and the ICAME bibliography of publications relating to English computer corpora; the only papers I found were on research by Atwell and his Leeds co-researchers on GI algorithms (see chapter 1), and these were on GI theory rather than on applications in other Corpus Linguistics research.

The research reported in this Thesis has attracted some interest from corpus linguistics researchers. Some initial evidence for this comes from citations of the papers which form the basis of chapters 3 and 4, for example: (Wiechmann and Fuhs 2006) on concordancing software for corpus linguistics research, (Arlund 2006) on a corpus linguistics approach to the acoustic, historical and developmental analysis of Sarikol Tajik diphthongs, (Al-Sulaiti and Atwell 2006) on the design of a corpus of contemporary Arabic, (Ludeling and Kyto 2008)'s international handbook on corpus linguistics, (Smith et al 2008) on corpus tools and methods for incorporating linguists' manual annotations. However, other citations are in research and development of algorithms and tools, rather than uses in corpus linguistics research; for example: (Worboys 2007) on integration of JBootCat and aConCorde for web mining, (Drayson 2007) on open-source applications for web-trawler and knowledge discovery, (Fussiger 2006) on integration of NLP tools with search-engine APIs.

In summary, Grammar Inference has not been applied much, if at all, in Corpus Linguistics research; but there are potential applications in corpus linguistics for language teaching and learning, and in corpus linguistics research on under-resourced languages.

Figure 2.1. Corpus Linguistics researchers from Leeds University, alongside Geoffrey Leech, original raison d'etre for the Corpus Linguistics conference

Chapter 3:
The use of corpora for automatic evaluation of Grammar Inference systems

The evaluation of grammar inference systems is clearly a non-trivial task, as it is possible to have more than one correct grammar for a given language. The 'looks good to me' approach, carried out by computational linguists analysing their own grammar inference system results, has prevailed for many years. This study explores why this method has been so popular, in terms of its strengths, and also why it is no longer adequate as a reliable means to measuring performance. Corpus based methods, that can be performed automatically, are investigated to see how they can meet the needs of this difficult problem.

3.1 Introduction

In the past few years, the Natural Language Learning community have produced systems targeted at the complex task of Grammar Inference (GI): automatically inferring or learning grammatical descriptions of a language from a corpus of language examples. A low-level GI task is to group or cluster words into tentative categories according to their distributional behaviour, e.g., Atwell and Drakos (1987), Hughes and Atwell (1994) and Roberts (2002). A more ambitious aim of GI is to propose grammatical phrase structure; a number of mature solutions have emerged, for example, GraSp (Henrichsen 2002), CLL (Watkinson and Manandhar 2001a), ABL (van Zaanen 2001) and EMILE (Adriaans 1992). All systems mentioned aim to be unsupervised, and thus can only rely on raw (unlabelled) text for their learning process. Results from these systems are promising – a variety of linguistic phenomena are induced.

Despite the advances in this field, the issue of thorough evaluation has been poorly address. Evaluation within Natural Language Processing tasks in general is problematic, not least because there is no obvious single correct output to measure against. For example, for PoS-tagging and parsing, different linguists advocate different tagsets and parsing schemes, making it difficult to compare accuracy metrics (Atwell 1996; Atwell *et al*. 2000). Ambiguity poses a similar threat in GI:

the basic problem is given a training corpus, there is no single correct grammar that represents it. In the majority of systems, a 'looks good to me' approach has been used: success is illustrated by presenting some grammar classifications or structures proposed by the system which appeal to the linguist's intuition.

There is a need to develop methods for consistent evaluation of GI systems. Without a reliable way of determining the performance of such systems, it will become increasingly difficult to assess how competently they are doing the task they are supposed to do. Nor will it be trivial to compare two or more systems, which would be very valuable in deciding which techniques and algorithms work best for Natural Language Learning.

This study investigates the feasibility of harnessing corpora that can be used to develop a standard mechanism (which aims to be reliable and accurate) for evaluating GI systems in the future. The next section explores the 'looks good to me' approach mentioned earlier to understand why it is so widely used, its merits and any shortcomings. Section three introduces various approaches for evaluating GI. A discussion takes place in section four to highlight important issues from the approaches reviewed in this study. Section five provides brings the study to a close with a conclusion.

3.2 Looks good to me

The success of this approach is essentially its apparent simplicity. A system performs its GI procedures on a piece of unstructured text, and its resulting grammar can be analysed by a computational linguist, generally the computational linguist who built the Grammar Inference system under scrutiny. A qualitative evaluation takes place based on the linguistic intuitions of the evaluator, by highlighting features or structures in the learner output which look "good", or reminiscent of structures in a recognised linguistic theory. Of course, the apparent simplicity is due to the fact that a linguist possesses the required skills and experience to easily deduce whether the grammar in question contains a plausible structure. Researchers who came into the field of GI from a computing/AI/machine learning background are less likely to be as proficient at evaluating grammars using this approach.

'Looks good to me' is an important method of evaluation, and certainly should not be discredited just due to its lack of automation. Verification of a system carried out independently by one or more linguists would provide – in most cases – a more reliable measure of performance. It is also arguably the most resource efficient, since it can be evaluated on different languages without the need of structured corpora (van Zaanen 2001). Examples of this method in use can be found in. (Finch and Chater 1992; Losee 1996; Vervoort 2000; Henrichsen 2002).

However, the method has many disadvantages. Evaluation of this nature is mainly conducted by the developer of the system rather than someone independent to the work. Thus, there is a high chance of bias whereby the systems successes are highlighted and shortcomings are glossed over, making it almost impossible to gain an accurate picture of system performance. Even without any bias, the process is time consuming and would rely on the subjectivity of the expert(s), and may also be prone to unknown external factors that can affect humans. And finally, it does not offer the facility for comparing different systems which would be of great benefit.

3.3 Automatic evaluation

Ideally, the process of evaluation should be performed automatically which will save on time and on the necessity of an expert linguist in the chosen language. This section focuses on methods that harness corpora for this purpose.

3.3.1 'Gold standard' treebank

This method is currently the most common to be adopted. It works by extracting the original natural language sentences from an existing treebank (the 'gold standard'), and using it as input to a given GI system. The structured sentences produced by the GI system are then compared to the structure found in the original treebank. Common metrics include recall (measures the completeness of the learned grammar) and precision (measures the correctness of the learned grammar) that can be calculated to give an objective measure of system performance. This method has been used in (Brill 1993; Déjean 2000; van Zaanen 2001)

Although on the surface, this method appears sound, and simple to implement, there are in fact many issues that cause it to be insufficient for it to be reliable and

accurate. Treebanks are expensive (both in time and money) to create, and so they are not in plentiful supply. The material within also tends to be exclusive to a particular domain, e.g., the ATIS treebank consists of sentences on questions and imperatives on air traffic, or the Penn Treebank is taken mostly from articles in the Wall Street Journal. If a GI system is not designed to learn from raw text of the same genre or subject,, then it is not an ideal candidate for a 'gold standard'. The way in which the treebank has been structured poses a problem. Just because it has been labelled the 'gold standard' does not necessarily mean that anything that differs is wrong – it may simply be different, but equally correct. However, such differences will clearly affect the measure of the systems' performance. It does also beg the question about whether certain treebanks are credible enough to be used for the purposes of evaluation. One must be confident that the quality and validity of an annotated is sufficiently high, otherwise, it is pointless evaluating systems with corpora that contain errors or are not well structured.

On a slightly more technical level, a large step to overcome is the likely discrepancy between the actual annotation schemes of the GI system to be evaluated and the 'gold standard'. For example, an increasingly common grammar being employed within GI is the *categorial grammar* (Ajdukiewicz 1935). However, there is no corresponding treebank whose annotation formalism uses CG, which means the only useful measurements that can be acquired from this mismatch are metrics such as *crossing-brackets*, but so much information is wasted. Therefore, a system of translation is required to either convert the original treebank to the annotation scheme used by the GI system, or the other way around, in order for a more precise comparison to commence.. This was the tactic taken by Watkinson and Manandhar (2001b) in their evaluation of CLL, whereby they set about establishing a 'gold standard' by translating the Penn Treebank annotation scheme into one using categorial grammar markup. Others have also advocated systems for mapping between parsing schemes to enable evaluation of parsers trained on treebanks with different encoding schemes; for example, (Forst 2003), (Daum et al 2004), (Kubler et al 2008).

3.3.2 Multi-annotated corpora

To overcome some of the main disadvantages of the 'gold standard' approach, the next logical step is to develop a multi-annotated corpus, as exemplified in Atwell *et al*. (2000). The premise is that the same corpus is parsed by a variety of systems, rather than merely one as all common treebanks are. Providing all parses are trusted

to be correct, then upon evaluation of a GI system, it is less likely to be penalised just because it produces different but equally correct parses, because it is being compared to a variety of such parses.

This step should make the evaluation fairer, however, it does add a new set of complexities. Not least due to the aforementioned annotation translation problem, which multiplies for each unique scheme used by the various parsers. Ideally, the same annotation would be used for all parses within the treebank – the 'gold standard' annotation scheme. But this too is fraught with difficulties, especially as it would be much simpler to accomplish a multi-annotated corpus by using off-the-shelf parsers.

Another issue is how to compare a given parsed sentence to be evaluated with a set of known correct parses. The best case is that a perfect match will be found. The worse case is no match. However, for many cases, it will be somewhere in between, where a number of partial matches are likely to exist. A best match approach seems logical, i.e., of the candidate parses, the one that is the most similar is the one that is then considered for the actual metric calculations.

3.3.3 Multi-corpora

In a similar vein to the above approach, the idea for using more than one corpus is primarily to avoid the domain specific issue that was addressed earlier (see section 3.1). Unlike other corpora, such as the Brown or LOB, which span many subjects in different contexts (newspaper articles, novels etc), well known treebanks fail to compare in size and variety. Therefore, this method aims to take advantage of smaller individual treebanks and combine into a single, large one. This would certainly make it fairer for general purpose learning systems, although perhaps there is less of a need for more focused systems.

Having said that, an alternative tactic is to treat each of the contributing corpora as individual and each as with the normal 'gold standard' method. You would then have a more detailed type of benchmark style scoring system, where not only is there an overall performance measure against all the corpora on test, it can also be seen on which type of corpora the GI system works best (or worse) on.

Of course, the most thorough approach would be if the smaller treebanks were also multi-annotated as described in the previous section. A 'multi-multi-annotated corpus' – if you like – would be a magnificent resource. This subtly shifts the aim of evaluation: the result or output is not a single overall "accuracy score", but an exploration of a range of parsing schemes and genres to highlight similarities and differences between GI output and a variety of accepted linguistic analysis schemes. Ideally such an analysis could be facilitated by a parsed-corpus exploration toolkit: a tool to compare parses in GI output and linguist-annotated multi-multi-corpus, and automatically highlight differences.

3.4 Discussion

There does not seem to be a perfect solution to the problem of reliable evaluation, whether it be manual or automatic. GI and parsing is very subjective, and it is unlikely that a method will be created that will satisfy everyone. However, what should be unanimous is that a system of evaluation needs to exist. It may not be perfect, but at least a fair and consistent scheme can be created.

Unfortunately, it is clear that the most thorough automated evaluation approaches could be such a burden to developers to implement that they will opt for a simpler approach. Which is why it may be worthwhile to outsource the task of creating the ultimate 'gold standard' into a research project within its own right. The creation of a collection of multi-annotated corpora as the central resource, as well as a variety of translation interfaces for commonly used annotation schemes, that allow easy comparison between output of a GI system and the 'gold standard' corpus. It could be packaged like an *evaluation toolkit* – a black box that is publicly available, easily accessible, and simple to add on as the final phase within the pipeline of the GI system.

Alternatively, a multi-multi-annotated corpus could be achieved by getting the GI community – or at least a group of GI researchers – to work together, each contributing their preferered "target" analysis schemes. Sutcliffe et al (1996) describes an analogous evaluation exercise for a range of (non-ML) parsers: each researcher was asked to "bring along" their parser analyses of an agreed set of test sentences to a joint workshop, and then highlight and discuss successes (and failings) of their systems.

Perhaps in a future GI workshop, GI researchers could be invited to bring along their preferred evaluation treebanks to challenge each other, and to each present both "looks good to me" and quantitative evaluations of systems for comparison.

3.5 Conclusion

The 'looks good to me' approach, despite the critical slant within this study, is not an inferior one. However, it will not meet the demands for robust NLL evaluation. Which is why corpus based approaches have been presented as the most feasible and reliable way of essentially trying to emulate 'looks good to me' whilst eliminating bias and providing consistency.

There is a great need for researchers within the field to begin addressing accurate evaluation techniques. The current 'gold standard' method (as described in section 3.1) is too basic and will simply favour GI systems that behave similar to the way the 'gold standard' treebank was parsed. Greater flexibility is required, and a method like the multi-multi-annotated corpus (see section 3.3) will be a beneficial innovation to do just that. Instead of replacing "looks good to me", the two approaches should be combined, so that evaluation is no longer a simple quest for a single "score", but instead an exploration of strengths (and weaknesses) of GI systems.

Chapter 4:
aConCorde: an open-source, extendable concordancer for Arabic

There is, currently, a surge of activity surrounding Arabic corpus linguistics. As the number of available Arabic corpora continues to grow, there is an increasing need for robust tools that can process this data, whether for research or teaching. One such tool that is useful for both of these purposes is the concordancer – a simple tool for displaying a specified target word in its context. However, obtaining one that can reliably cope with the Arabic language had proved difficult. Also, there was a desire to add some novel features to the standard concordancer to enhance its usefulness within the classroom – easy-to-use root- and stem-based concordance and integration to corpus clustering algorithms are two examples. Therefore, *aConCorde* was created to provide such a tool to the community.

4.1. Introduction

'If a corpus is to be useful, we obviously need to be able to search it quickly and automatically to find examples of a particular linguistic phenomenon (say, a word), to sort the set of examples as required, and to present the resulting list to the user. The kind of program which performs these tasks is a concordancer, and the output it produces is known as a concordance. A concordance is a list of all the examples of the target item (the linguistic phenomenon being searched for), normally accompanied by enough context to enable a human being to study the item's occurrence in detail.' (Leech and Fligelstone, 1992)

In effect, a concordancer is a tool for summarising the contents of corpora based on words of interest to the user. Lexicographers can use the evidence of a concordance to establish whether a word has multiple senses, and also to help define its meaning (Sinclair, 1987; Zernik, 1991). Concordance tools can be beneficial for data-driven language learning (Johns, 1990).

A number of studies have demonstrated that providing access to corpora and concordancers benefits students learning a second language. Just as lexicographers and linguists can find insights into grammatical structure by studying concordance

output, so can language learners (Dodd, 1997). Cobb *et al*. (2001) detailed the improved rates of vocabulary acquisition when using a software tool of which a concordancer was a major component.

The simplicity and usefulness of concordancers is such that they should be a valuable tool for any linguist. Unfortunately, as is the general trend within corpus linguistics, research priorities have focused on European languages (although there has been a recent shift towards developing tools in non-European languages such as Chinese, Japanese and Korean). This has meant that there are fewer tools for other languages such as Arabic. When this project began, to the best of our knowledge, there were no readily available concordancers that function correctly with the Arabic language. Arabic is an international language rivalling English in the number of mother-tongue speakers (al-Sulaiti and Atwell, 2006). Graddol (1997) predicts that the number of Arabic mother-tongue speakers will rise from 200 million in 1995 to about 480 million by 2050, whereas English mother-tongue speakers will rise from 360 million to about 500 million within the same period. Considering the growth of interest in Arabic, we felt that Arabic linguists deserved a useful, usable concordancer tool. To this end, the *aConCorde* project was established.

4.2. Why is Arabic Concordancing Hard?

It is fair to say that for non-Arabic developers, writing tools to process Arabic is more difficult. This perhaps explains why many tools developed for Western languages do not extend well to Arabic. The two most commonly perceived difficulties with Arabic language processing are:

1. The cursive nature of Arabic script means that letters are represented differently depending on whether they occur at the beginning, middle or end of a word. Characters within words are always joined, and never written individually. Vowels are “second-class” characters, in that they can be omitted, and are left for the human reader to infer from context.

2. Arabic is written from right-to-left, as opposed to the majority of other languages that are, of course, written left-to-right.

The process of transliteration is a common approach to resolving these issues. The Buckwalter system (amongst others) will convert Arabic script to an equivalent based on the Roman alphabet (Buckwalter, 2002). It means that the text orientation is also converted, too: the transliterated text reads left-to-right. However, this alone does not fill in missing vowels from the source text. From a computational perspective, transliteration has historically been a necessary step: computers used to be restricted to ASCII (or similar) character sets (i.e., Roman alphabets). However, thanks to the popularity of the Unicode standards since the early 1990s, most modern operating systems and products now support multi-lingual text encodings, and the fonts to display Unicode characters are increasingly common. Implementation of bi-directional text components are also part of most modern operating systems and platform-independent programming languages like Java.

Due to these developments, there is little reason why existing concordance tools have not been adapted to take advantage of the above technologies in order to render the *aConCorde* project redundant even before it had begun. One implementation that sought to provide multi-lingual concordancing, was *xconcord* (Ogden and Bernick, 1996; Boualem *et al*., 1999), developed at the Computing Research Laboratory (CRL), New Mexico University. The CRL produced their own graphical components that were able to display Unicode text and they worked successfully at displaying Arabic text. Even though *xconcord* was ahead of its time when it was first developed in 1996, it did not become a mainstream application. It would still be a useful tool today, but it is hindered because it was written for the Sun Solaris platform. This is not widely used – especially by the average linguist – since this type of system is typically used on expensive high-powered workstations and servers. Unfortunately, therefore, it was not possible to experiment with it as resources required were not available. Development ceased many years ago, but the program can be downloaded from http://crl.nmsu.edu/software/

The real difficulty in Arabic concordancing lies not in the displaying of results, but in the inadequacy of searching for Arabic word stems. Commonly, when performing concordance searches, the user wants to see the usage of words all from the same stem. Concordancers tend to offer *wildcard* searches, for example, *perform** (where the asterisk means any character zero or more times) would match *perform*, *performs*, *performer*, *performers*, *performing*, etc. However, Arabic morphology is substantially more complex than English. Arabic words are derived from a root of typically three consonants, and each root has a set of patterns which can add additional characters as an affix, a prefix or even an infix. An example

would be the transliterated Arabic root *ktb* (write). Patterns associated with that root can produce words semantically linked, including verbs like *kataba* (he wrote) and *naktubu* (we write), and many nouns like *kitAb* (book), *maktuub* (letter) and *maktaba* (library). However, performing a wildcard search like **k*t*b** would return too many matches, many of which are not associated with the *ktb* root, simply because they happen to contain these consonants. Many sources cover Arabic grammar, however, issues about Arabic and computing are well introduced by Khoja (2003) and de Roeck (2002).

With the exception of the Qur'an, the lack of vowels in modern written text provides difficulties too. This leaves a great deal of ambiguity that is only resolved when looking at the context of a given word. An English example may be *fr* that could be *for*, *fir*, *fur*, *far*, *four*, *fear*, *fair*, *fire*, *afar*, *afore*, etc. Yet, if you can only search for *fr* even though you are only interested in *far*, it would clearly be frustrating to have to read through irrelevant results. It is unlikely that these specific issues will be resolved by generic concordancers.

4.3. Arabic Concordance with *MonoConc*, *Wordsmith*, *Xaira* and *aConCorde*

For the task of concordance, the market leaders are *MonoConc* (Lawler, 2000) and *WordSmith* (Scott, 2004). Both are commercial products and are well-established tools for text analysis. They are only available on the Microsoft Windows platform. *Xaira* (Burnard, 2004) is a relative newcomer but stems from the well-known *SARA* toolkit for users of the British National Corpus. This section primarily seeks to compare the ability of retrieving concordance output from Arabic corpora using these established tools.

The corpus used in the comparisons is the Corpus of Contemporary Arabic, or CCA (al-Sulati, 2004), consisting of 850,000 words of various text-types in over 400 files. This corpus is annotated according to the TEI standards using XML markup, and all files are encoded in the 8-bit Unicode standard, UTF-8. A key aim of the CCA was that it should be for use within a pedagogical context to supplement the teaching of Arabic as a foreign language using corpus-based approaches.

4.3.1 MonoConc

Developed by a linguist, Michael Barlow, for language teachers and researchers, *MonoConc* has proved to be very popular for language analysis because it comes with many additional features, such as collocation discovery and tag-sensitive searches. The *MonoConc* website http://www.monoconc.com states, 'Some users have been using *MonoConc/Pro* with Arabic. I don't have any details of their procedures.' Unfortunately, despite initial optimism, using Arabic texts with *MonoConc* was not totally successful. In Figure 4.1, discovered collocates are underlined, although they should not be there. For unknown reasons the text in the top pane did not display using Arabic fonts. Issues that arose during experiments were as follows:

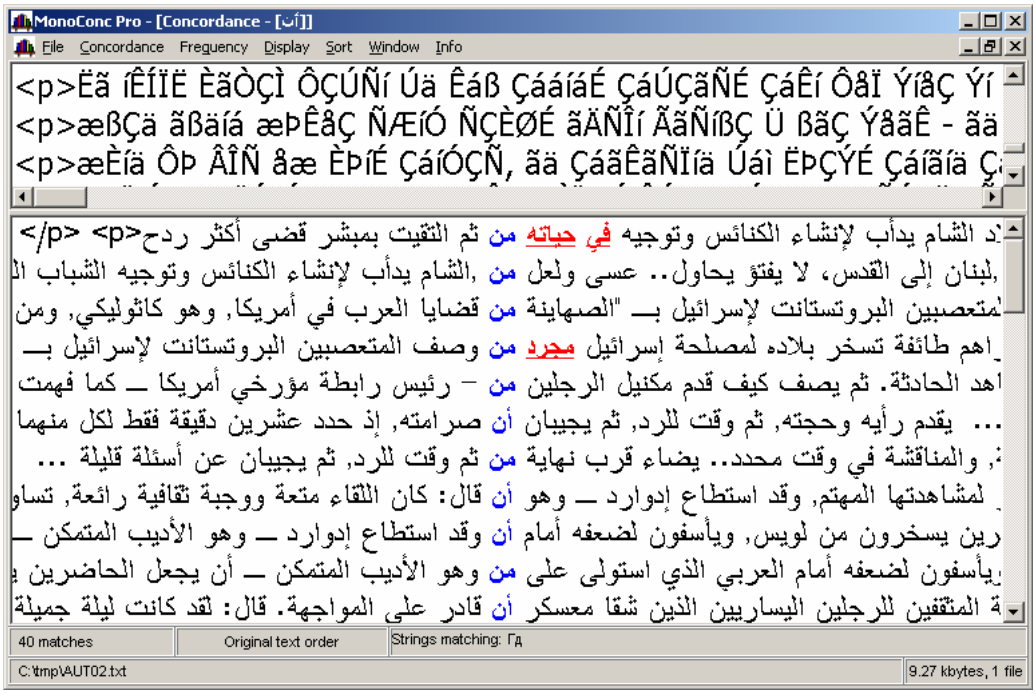

Figure 4.1: Arabic concordance using *MonoConc*

Unicode support. It appeared that *MonoConc* does not support texts encoded in UTF-8. It was therefore necessary to convert texts to the Windows Arabic Codepage-1256 before the program would display the Arabic text correctly.

Incorrect concordance order. Perhaps the most significant problem for performing Arabic concordance is that the output is in the wrong order (See Figure 4.1). *MonoConc* does not take the right-to-left nature of Arabic into account when displaying results. For an English reader, it would be equivalent to seeing the output:

“on the mat. [sat] The cat”, rather than “The cat [sat] on the mat.” (where the bracketed word is the target word). This was observed by Hoogland (2003) during the Nijmegen Dutch-Arabic Dictionary Project, ‘This seemed a serious shortcoming in the beginning, but soon we experienced that we got used to this very quickly.’ Hoogland *had* to get used to it because it was better to have a broken concordance than none at all. It is clearly far from ideal – it would be a problem for teaching purposes in particular as it would confuse the learner. (N.B. When the concordance results are saved to a file, and this file is viewed with a text editor, the ordering issue disappears.)

‘Noisy collocates’. A strange artefact appeared in the concordance output whenever a match included a discovered collocate. It was default behaviour for the software to highlight found collocates. However, the only problem was that these collocates were actually inserted into the wrong position within the concordance context, and as a result caused a certain amount of interference. This was remedied by turning off the option to highlight collocates, after which the highlighted words would then vanish from the context.

Addition of unwanted terms. Despite a simple search term being submitted to *MonoConc*, sometimes it included matches within the results that are not equal to the target word. This behaviour appears to be random as it only occurs with some words and not others. This is illustrated, too, in Figure 1 where the central column contains tokens that were different from the search term.

4.3.2 WordSmith

WordSmith was first released in 1996 and is still developed by a linguist, Mike Scott. *WordSmith* is currently at version 4 and sports three tools: Wordlist, Keyword and Concord. The final tool is of interest in this report, but it should be obvious what the others do. The documentation notes that *WordSmith* supports Unicode, which obviously lends itself to the analysis of the CCA. A demo version is available, which was used for this report. It retains all the functionality of the full version, but significantly limits the number of results returned when performing queries.

A degree of success was initially achieved when using the tool during September 2005, in that it was possible to get the tool to display Arabic script – which means its Unicode support was working. Unfortunately, like *MonoConc* it

displayed the prior and posterior contexts in the opposite order required for a right-to-left language. The matches are displayed in two columns, the left column must be read from right-to-left first, and then the reader can continue with the second column, which begins with the target word, followed by the rest of the context (see Figure 4.2).

We were only recently made aware that *WordSmith* had in fact been updated to cope better with Arabic concordance. We had initially overlooked this because the version number for the latest issue of *WordSmith* had not been incremented to indicate any change. We are pleased to report that the tool does in fact work properly with Arabic text, using the correct ordering. That said, it did require a little more configuring, such as enabling right-to-left support at the level of the Windows operating system. You also need to spend some time in the *WordSmith* preferences dialogs to add Arabic to its supported language set and then select it to be used within the current session. Other Arabic colleagues who we asked to experiment with *WordSmith* had great difficulty achieving a result. The feeling was that whilst the support was clearly there, it was not necessarily as accessible as they would have liked it.

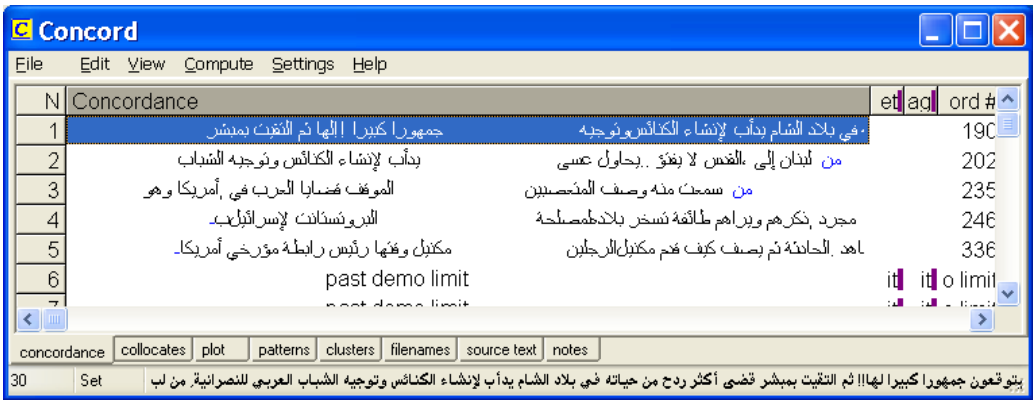

Figure 4.2: "Incorrect" Arabic concordance as displayed by *WordSmith* (now corrected in the latest version)

4.3.3 *Xaira*

Xaira is a new tool designed to supersede *SARA* (Dodd, 1997; Aston and Burnard, 1998) – an application for text analysis on the British National Corpus. *Xaira* is no longer tied to the one corpus, instead opting for a more generalised application. It takes advantage of Unicode and XML technologies to achieve this (Burnard and Dodd, 2003). At the time of writing, its first version is still in beta

testing. It can perform complex searches as it can filter results according to the XML annotation.

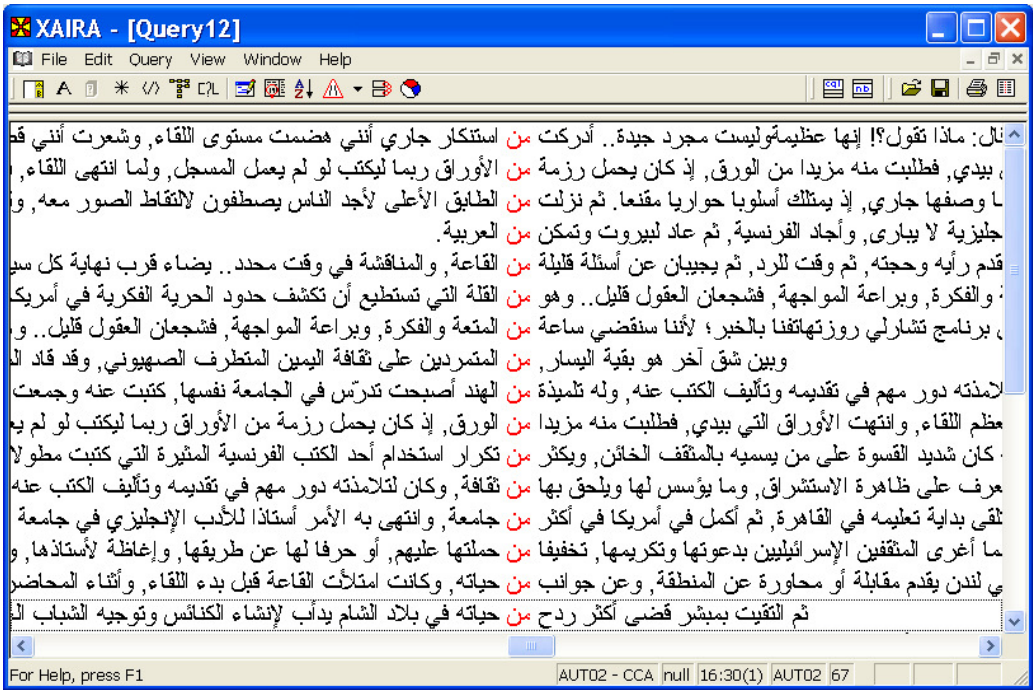

Figure 4.3: Example of *Xaira* producing correct concordance output

For example, a spoken corpus is commonly annotated with information such as the gender of the current speaker. Therefore, *Xaira* is able to perform searches based on male language usage and compare it to female usage.

Xaira does not suffer any problems in that it displays the entire concordance in the correct order (see Figure 4.3). However, in order to get to the point of looking at results, the user must go through a relatively long procedure, using an additional tool to prepare the source texts into the format expected by the software. This need not be a problem for dedicated computational linguists, but may deter more casual users, such as linguists or language teachers wanting to use corpus and concordance evidence only occasionally.

4.3.4 aConCorde

aConCorde (Roberts and Atwell, 2005) is neither a commercial product nor a research project. In terms of features, it is relatively basic when compared to the systems already discussed. The design aim was to ensure it was as multi-lingual as

possible, with extra emphasis on the Arabic language. It will be no surprise, therefore, that it works well for right-to-left languages (see Figure 4.4). No additional configuration of both the operating system or the application is required. *aConCorde* cannot detect the language of selected texts, but it can detect easily whether the language is left-to-right or right-to-left. Therefore, it will adjust its concordance display automatically depending on the texts currently being analysed.

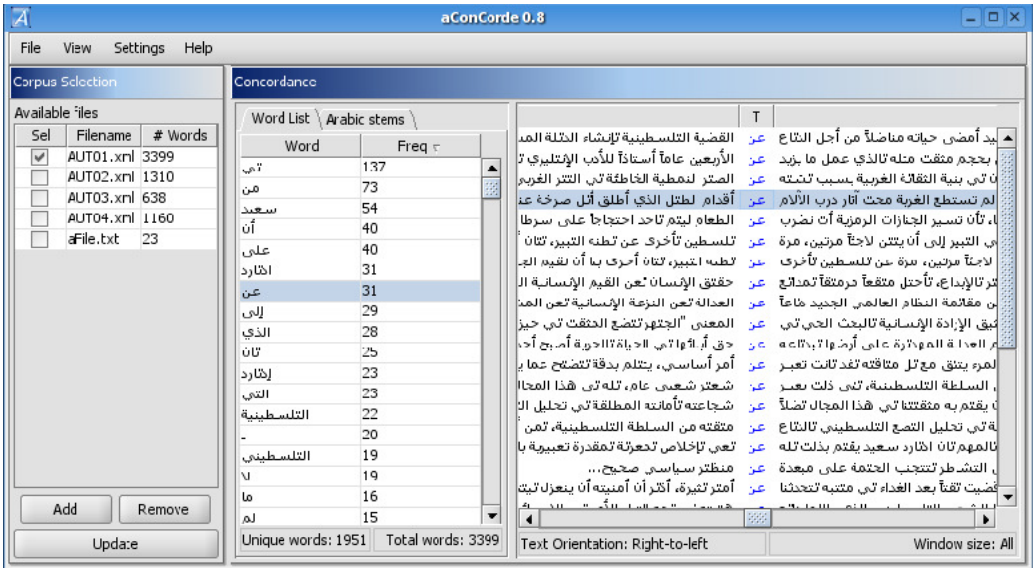

Figure 4.4: Example of *aConCorde* displaying Arabic concordance correctly

An additional feature that is useful for Arabic users is the provision of an Arabic interface. Not only does this provide Arabic translations for all the menus, buttons *etc*., but even switches the entire application layout to right-to-left. In theory it is relatively easy to add additional language support, although it can be difficult to find willing translators.

4.3.5 Discussion

The *aConCorde* project was started solely because we could not find a concordancer to work correctly with Arabic. The *Xaira* software has been demonstrated to work with Arabic too, so it could be argued that there was no need for *aConCorde*. Unfortunately, *Xaira* was only discovered *after* development had already begun. (And the same goes for the recent discovery of *WordSmith*'s recent update.) Initially, it might seem that *Xaira* and *WordSmith* have now rendered *aConCorde* redundant, but there is a market for both. Packages like *Xaira* are very sophisticated, but with increased power and flexibility comes additional complexity.

It certainly lends itself to research purposes requiring complex queries for fine-grained language analysis. However, it is less appropriate in the language teaching environment. Teachers and students may be put off by the complex interface and the amount of effort needed to get a concordance output. By contrast, *aConCorde* is relatively simple to get up and running – literally select a corpus to use, then either type in your query or select *word* from the word frequency panel and you will be presented with your results. This approach could well be more suited to this audience. Also, *aConCorde* does provide special searches that are specific to Arabic analysis in the form of root/stem-based concordance, and this is not found in any other package. Finally, the *aConCorde* software is free to use by anyone (as is the source code).

4.4. The *aConCorde* System

The *aConCorde* project was originally conceived as a prototype to illustrate the application of new developments in software engineering, specifically Java programming language support for language-independent (and direction-independent) processing of Unicode character sets, rapid development of graphical user interfaces, and ease of integration with and into other open-source code and tools. This makes it much easier to write tools to support not just the Arabic language, but practically any language. Within one afternoon, a very simple multi-lingual concordance engine was programmed, and during the following afternoon, a basic graphical user interface was added. The user is able to switch between a native English or Arabic interface. All Arabic text is displayed correctly, all concordance is output as expected by an Arabic reader, and there is no transliteration required for the input or output. This prototype was very crude and would not have been adequate for proper use in the real world – it was simply a quick experiment to prove that the display of Arabic concordance was in fact fairly trivial. Of course, developing a more robust, feature-packed and user-friendly tool is infinitely more difficult and time consuming.

4.4.1 *aConCorde* features

Subsequent improvements and extensions have allowed *aConCorde* to mature into a more reliable and useful tool. The *aConCorde* system contains a number of features that make it to stand out from competing products, including:

• full Arabic support (no need to transliterate to Roman alphabet before concordance)

• English and Arabic native interfaces

• multi-platform functionality – can run on most major operating systems. It supports an extensive range of character encodings, including Unicode (UTF-16 and UTF-8), Windows Arabic (CP1256), IBM Arabic (CP420), MacArabic, ISO Latin/Arabic (ISO 8859-6) and ASCII encoding

• multi-format support that allows you to load text files, XML files, HTML files, RTF files and MS-Word files directly

• can save results either as a plain text file, or HTML file that can keep the alignment of the concordance

• word frequency analysis

• concordance that can be sorted on left or right contexts

• a range of query options: key-word, phrase, proximity, boolean, wildcard and Arabic root/stem queries, and

• *aConCorde* is freeware and open-source (released under the General Public License http://www.gnu.org/copyleft/gpl.html).

The *aConCorde* system is built using the Java programming language. Java allows the programmer to produce software for the most popular operating systems without any extra effort, which is why *aConCorde* will run correctly on Microsoft Windows, Linux or Mac OS (and others) providing the user has the Java Runtime Environment installed. It is important to note that many earlier Arabic tools were reliant on Arabic Windows (a localised version of Microsoft Windows with support

for Arabic script and right-to-left interfaces). *aConCorde* will of course run on standard Windows or Arabic Windows.

The Java platform is also one of the reasons why producing a multi-lingual capable application was straightforward. Internationalisation features were an important aspect of the design of Java. For instance, Java's internal mechanism to store text uses the Unicode standard, and all visual components have in-built support for left-to-right or right-to-left languages (some components can even cope with Japanese top-to-bottom text orientation).

4.4.2 Root- and stem-based concordance

As discussed in Section 4.2 the wildcard searches that make it reasonably simple for stem-based concordance in Western languages do not scale universally. To recap, an Arabic root is typically a sequence of three consonants (although two, four and five consonant roots exist) which forms the basis on which words are constructed. For example, the root *ktb* (write) has stems derived such as *katab* (write) and *iktatab* (register). From the stem you can derive the various morphological forms using affixes, infixes and suffixes, e.g., *Taktub* (she writes), *yaktub* (he writes), *aktub* (I write), and so on.

4.4.2.1 Buckwalter's morphological analyser

Buckwalter's morphological analyser is distributed by the LDC and has been used by many projects including the LDC's Arabic Treebank project (Maamouri *et al.*, 2004). It consists of three lexicons (affixes, suffixes and stems) and three sets of rules that specify how these lexicons can be intersected to derive Arabic word tokens. The analyser reads in a file and processes each word in isolation. By utilising the lexicons and rules, it can break down the input token and produce all possible valid morphological solutions. In cases where there is more than one solution, it does not offer any hints as to which is the most probable analysis.

The databases and the analyser work with Buckwalter's transliteration system. Until recently, direct input/output of Arabic characters in computer systems has not been trivial. Buckwalter's system is a one-to-one mapping of the Roman alphabet and the Arabic alphabet. Conveniently, there is also a one-to-one mapping to the Unicode character encoding standard.

There are other transliteration systems in use although there is no single official transliteration standard. Buckwalter's transliteration is not recognised within the Arabic linguistic community at large (Beesley, 2003), since it is relatively modern, and aimed at the computational processing of Arabic. Its use of Latin punctuation symbols for Arabic letters is clearly less intuitive to a human reader than some of the more phonetic-based transliteration systems. For example, Buckwalter uses '$' for Sheen, whereas others, such as Qalam (Heddaya, 1985), use 'sh'.

4.4.2.2 Buckwalter and *aConCorde*

If a user wished to produce a concordance for all words derived from the stem *katab*, a wildcard search approach is, clearly, not feasible. The alternative is to search for a manually hand-crafted list of derived words. However, with *aConCorde*, this burden is removed. Using the databases of Buckwalter's morphological analyser, *aConCorde* can provide the roots and stems *a priori* to the user. They can then select to search for one or more stems, or even everything within a given root. Buckwalter's databases were converted from Buckwalter's transliteration alphabet to Unicode so that the root/stems are displayed in native Arabic (see Figure 4.5).

There is a limitation here in that the user is always presented with the full database, even if many of the terms are not present within the corpus being currently analysed. When loading an Arabic corpus, it might be preferable to analyse each word and determine its stem and root, and then only have these entries visible. However, real-time stemming of a corpus in Arabic is difficult due to the complex morphology. For example, it is quite easy to remove accidentally those affixes that were in fact part of the word. That said, there are stemmers that exist, such as Khoja's (2003) stemmer used in her APT tagger.

4.4.3 Similar terms and word clusters

A novel feature that is specific to *aConCorde* is the use of similarity functions. These are borrowed from the information retrieval (IR) domain. They are typically used to find similar documents based on the terms contained within. In *aConCorde* its low-level model, from an IR perspective, regards sentences as "documents". The concordancer has a set of term-frequency vectors and these can be plotted into a vector space. During the concordance, the user can specify their search term, select a sample line from the concordance and its term-vector will be compared to the others using cosine similarity (Jurafsky and Martin, 2000: 650) and then be presented with similar sentences from within the corpus.

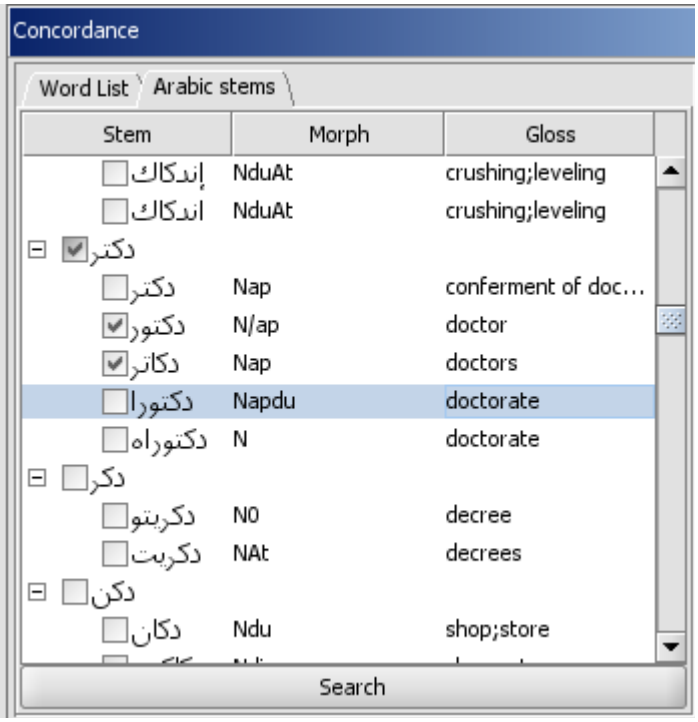

Figure 4.5: Example of root/stem selection within *aConCorde* using Buckwalter's stem database

4.4.3.1 Clustering and *aConCorde*

aConCorde provides a means for the user to interface external clustering code and display the output within the concordancer itself (see Figure 4.6). Section 4.5 illustrates the advantage of Java open-source development: more potential features could be readily added to concordance software to enable alternative perspectives and analyses of a corpus.

The term similarity and clustering features were implemented primarily for the use of concordance within the learning environment. Concordance has already proved successful within the classroom, but this concordancer can provide additional information to the user to assist in building vocabularies and grammatical patterns.

4.5. Clustering

The principle of clustering algorithms is to divide a set of objects into clusters. By using an appropriate measure of similarity, a good clustering algorithm will place

objects that are similar into the same cluster, whereas dissimilar ones are clustered into different groups. It has the advantage of being truly unsupervised, i.e., it requires no prior knowledge about the data being clustered. Clustering is a broad subject and has been widely applied. Within computational linguistics, it has been used successfully in syntactic (Atwell and Drakos, 1987; Finch and Chater, 1992; Hughes, 1994) and semantic (Ibrahimov *et al*., 2001) classification and information retrieval systems.

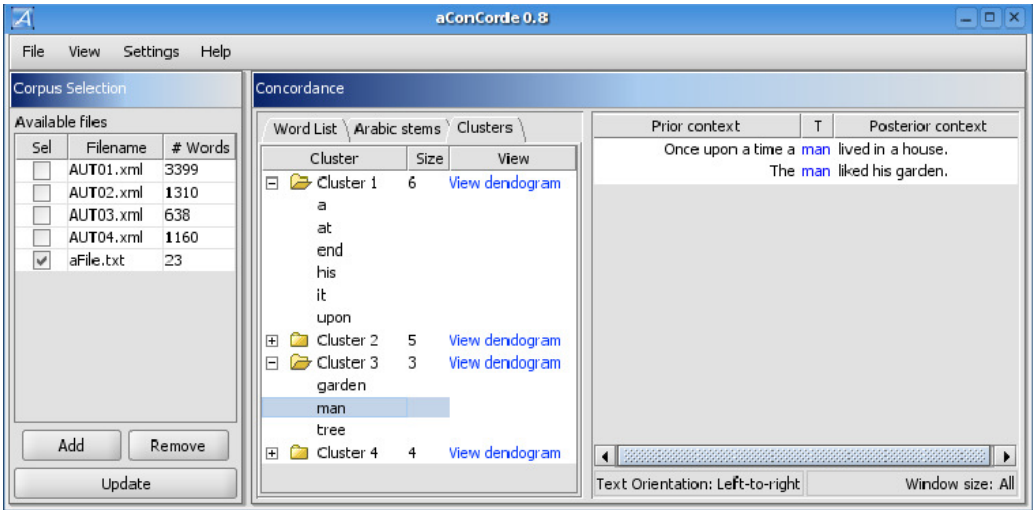

Figure 4.6: Cluster view within *aConCorde*

Clustering is a mathematical procedure that merges points in a vector space that are close to each other. To cluster words, a method is required to represent a given word into a vector to be plotted. The approach used here records collocation frequencies of a given content word relative to function words, for a specified window size. This data can be converted simply to a vector ready to be clustered. Chapter 5 gives more details of my experiments with this algorithm; the following outline introduces the method and its application to English and Arabic corpus text.

4.5.1 Roberts' Method

Function words have an important role in language. The words themselves contain little meaning but instead specify relationships between other (content) words. Function words tend to be used frequently. The rationale behind Roberts' method is that if function words are so important and frequent, all content words within a corpus should co-occur with them a number of times, even within a small window. An hypothesis follows from this: words that co-occur similarly with the

same function words could be considered grammatically similar, in that they share the same word class.

For a given corpus, *T*, you can construct a set of function words, *F* (see Section 4.5.2), and a set of content words, *C* (i.e., a word list of all the words in the corpus). A window size, *w*, must also be selected; a window size of *n* means *n* words before and *n* words after a target word, thus spanning 2*n*+1 words in total, including the target word.. Select a content word, *c*, from *C* and then select a function word, *f*, from *F*. Scan the corpus for every instance of *c*. For each one, check if *f* co-occurs within the window, *w*. If it does, record the relative position from the target content word. Figure 4.7 illustrates such an example, showing how the content word 'first' co-occurs with the function word 'the' in a window size of four.

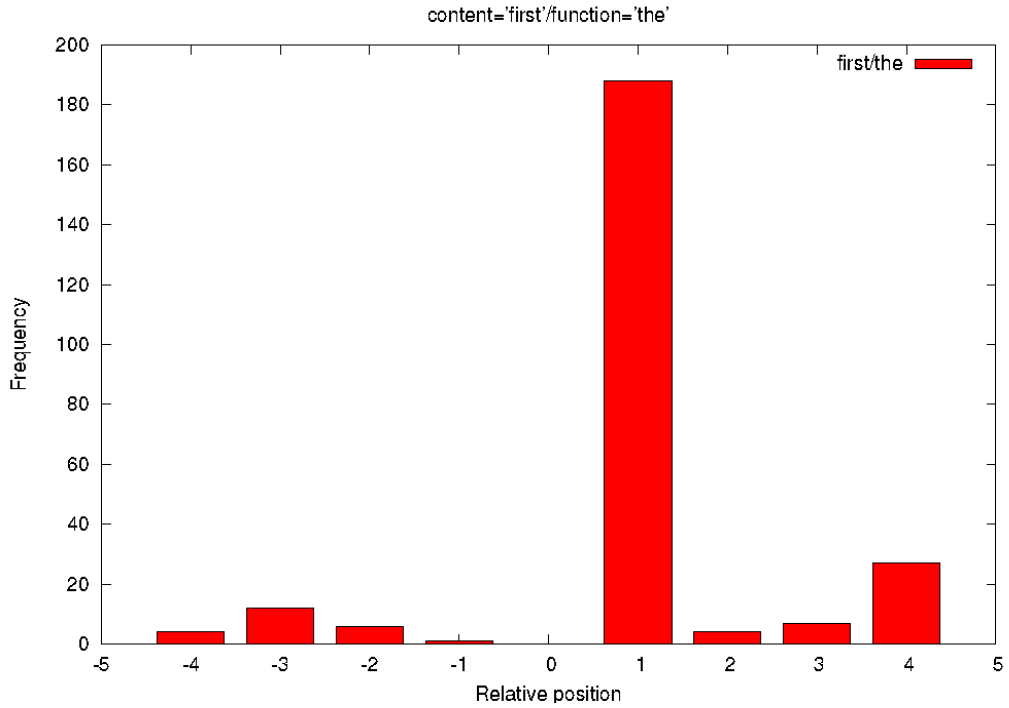

Figure 4.7: Graph showing how *first* is positioned relative to *the*

By treating each possible position of the target content word as a separate dimension, it is possible to represent the graph as a vector storing the frequency of occurrence for each dimension.

$$\begin{bmatrix} 4 \\ 12 \\ 6 \\ 1 \\ 0 \\ 188 \\ 4 \\ 7 \\ 27 \end{bmatrix}$$

An equivalent vector is needed for every function word in *F*, and the vectors concatenated to create a single vector that represents a profile of the content word, *w*, with respect to the set of function words, *F*. Once this has been applied to all of the words in *C*, a full set of vectors has been acquired, and these represent points in a high-dimensional vector space ready for the clustering phase.

4.5.2 Acquiring Function Words

The simplest way to acquire function words is to find an expert of the chosen language who can provide you with a list of them. This should be a finite, closed-class list. In practice, the list does not have to be perfect or complete, as clustering is an imperfect approximation. To acquire in a more automatic fashion, selecting the fifty most frequent words of a corpus will yield a reasonable list with adequate precision and recall. This is likely to return a few frequent content words, too.

Some function words are not that frequent, but are used consistently throughout a corpus. A fairer automatic method is to take a corpus and split it into a set of subcorpora. For each subcorpus, generate a word list, then the words that appear in all of the subcorpora are classed as function words. The exact proportions for the subcorpus size can depend on the corpus itself. For a million words, multi-genre corpus, having each subcorpus representing one percent of the overall corpus gives reasonable results.

4.5.3 Clustering Algorithms

There are a range of clustering algorithms available. Hierarchical agglomerative algorithms are well-suited to word clustering. Each word in the vector space starts off as an individual one-object cluster. The algorithm evaluates each pair of points to find the closest, which are deemed the most similar, and merges them to

form a single cluster. The way in which algorithms compute the distances between clusters is typically the differentiating factor (Everitt, 1993).

Figure 4.8 shows an example of the Complete Linkage algorithm. This is also known as the *furthest neighbour* method since its measure of distance between two clusters is to find the furthest two points within. Some of the more sophisticated, albeit, less simple to visualise, include Group Average and Ward's Method, which are good performers in this task (Zupan, 1982; Hughes and Atwell, 1994). The results return as a set of clusters, the size of which can vary. A nice feature of hierarchical methods is that you can profile the clustering process and generate a dendrogram that reveals how the cluster was formed, e.g., which terms were the most similar (see Figure 4.9).

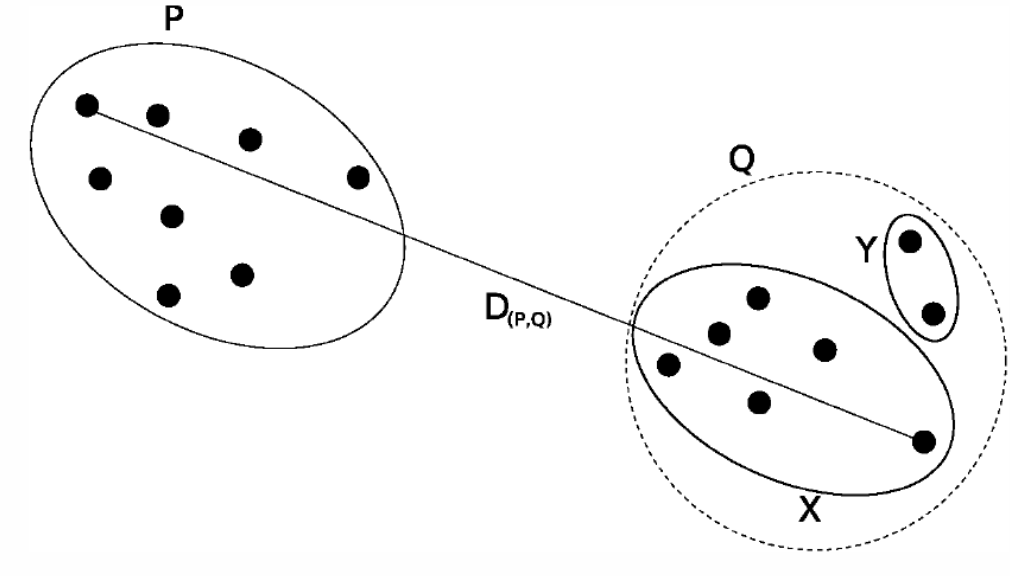

Figure 4.8: Diagram illustrating 'furthest neighbour' clustering algorithm

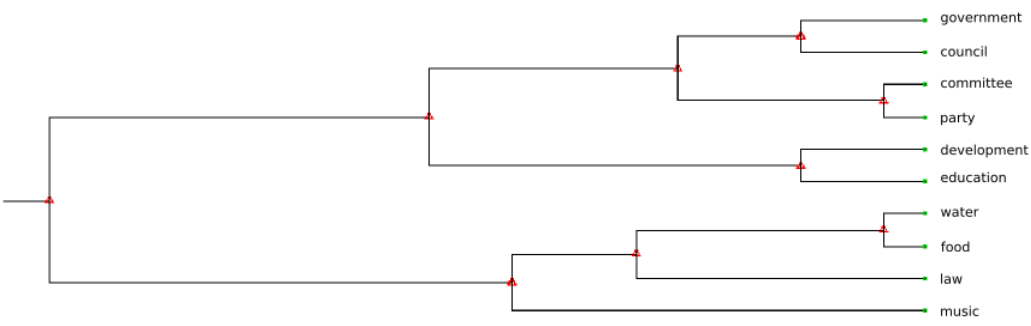

Figure 4.9: An example of a cluster visualised as a dendrogram

Even though clustering is an unsupervised approach, such algorithms have reported accuracies of 85+ percent in correctly grouping words into course-grained parts-of-speech. Figures 4.10 and 4.11 show a few samples of the clustering output. A common drawback with clustering is that it can be prone to merging two well-defined clusters, (a set of nouns with a set of adjectives, for example), because at a given stage, they happened to be the 'closest'. The clustering algorithm has to know, as it were, when to stop, and this can be difficult for the program to judge automatically. If the algorithm stops too soon, the result is a large number of clusters each containing only a few words, which is unlikely to be very useful; and if the algorithm stops too late, there will be a small set of clusters each containing many words with not much (apparently) in common.

Cluster 61		Cluster 82		Cluster 49	
dr		her		committee	
miss		his		council	
mr		my		government	
sir		your		party	
mrs		herself		development	
NOUN	**100%**	him		education	
		them		food	
		himself		water	
		PRON	**100%**	law	
				music	
				NOUN	**100%**

Figure 4.10: Examples of clusters obtained from the LOB corpus

Cluster 14		Cluster 25		Cluster 30	
أشار	*He pointed out*	الإذاعة	*The broadcasting*	الجزائر	*Algeria*
أضافت	*She added*	الصحيفة	*The newspaper*	القاهرة	*Cairo*
أعلن	*He announced*	البيان	*Al-Bayan (newspaper)*	القدس	*Al-Quds*
أكد	*He confirmed*	المصدر	*The source*	الرياض	*Riyadh*
قالت	*She said*	المقرر	*The fixed…. (date)*	عمان	*Oman/Amman*
تابع	*He kept on doing*	المسؤول	*The official*	طهران	*Tehran*
VPERF	**100%**	المتحدث	*The spokesman*	باريس	*Paris*
		الوزير	*The minister*	بغداد	*Baghdad*
		صحيفة	*A newspaper*	بيروت	*Beirut*
		مصادر	*Sources*	دمشق	*Damascus*
		مصدر	*A source*	فرنسا	*France*
		مسؤول	*An official*	غزة	*Gaza*
		NOUN	**100%**	لندن	*London*
				موسكو	*Moscow*
				واشنطن	*Washington*
				NOUNP	**100%**

Figure 4.11: Examples of clusters obtained from the Arabic Treebank corpus

4.6. Conclusion

This study has summarised many of the issues regarding the difficulty of Arabic concordance. The *aConCorde* project attempts to resolve some of the core issues, for example, interactive searching that displays the Arabic text correctly and stem-based searching. However, at the time of writing, *aConCorde* is still limited in terms of its features compared to the more established products on the market, these include:

• Mark-up: *aConCorde* is generally ignorant of mark-up annotation within a corpus. Whilst it can successfully parse XML and HTML files, it essentially works by filtering out the tags to isolate the content. If a corpus was annotated with part-of-speech tags *aConCorde* would not recognise them and they would be ignored (even if they were encoded in XML), or treated as "words", as if part of the text itself. Heavily annotated corpora could therefore benefit from some pre-processing to strip away such mark-up.

• Full context: *aConCorde* does not have the functionality to allow the user to see the full context of a selected concordance item. The current display is to see the word within the sentence it was found, and that is the largest scope currently implemented.

• Arabic root/stem database is always exhaustive rather than reflecting only the tokens available in the corpus that is currently loaded. This could be a disadvantage in some circumstances as the user may have to browse through many stems that have no coverage.

• Clustering integration is at present crude and takes a long time to gather and display the data.

• Computational resources: *aConCorde* can be resource intensive with large datasets. Whilst the software utilises extremely scalable indexing technologies (courtesy of the Lucene indexing library: http://lucene.apache.org/) similar to those found in modern databases (e.g., binary inverted indexes), loading corpora (10,000

words per seconds), or gathering the concordance for highly frequent words, takes time (retrieving 3,000 results could take a few seconds). However, once corpora are loaded, those files do not need to be reloaded and will be available immediately for all subsequent instantiations of the concordance software.

Naturally, the development of *aConCorde* will continue, and the limitations mentioned here are high priority issues that will be addressed in future issues. However, *aConCorde* has already made significant progress. The development of a simple user-interface with novel exploration features mean that it is well suited to a learning environment. With the foundation firmly laid, we can focus on other tasks in future, for example, improving and creating fresh approaches to intuitive morphological searching, or improving word similarity exploration through machine learning methods such as clustering. We also hope that the established concordance tools will realise the demand for products that cope with Arabic script, and adapt their software accordingly.

Chapter 5:
Grammatical Inference of word classes using distribution analysis of content words with respect to function words

The clustering approaches outlined in 4.5 have been used to perform extended experiments to determine the effectiveness of clustering techniques for the task for automatic part-of-speech acquisition. This chapter outlines the steps taken and the justification for various decisions relating to corpus choice, experiment design, as well as findings from the experiments.

5.1 Obtaining a corpus

Choosing a corpus should never be a trivial task as they can influence greatly the performance of a system. For example, if a speech recognition system were being developed, then it would benefit from a corpus of spoken language to train the system, rather than any other types. It does not mean that a corpus of written language is not valid, however, it merely does not reflect the most appropriate usage.

Another important feature of a corpus is its size. The major reason for this is that it combats the data sparseness problem. A common approach for researchers in the past few years has been to compile massive corpora by extracting text from USENET newsgroups. Examples include:

- Hughes (1994) collected 35 million words, of which 30 million were taken from USENET.
- Finch and Chater (1992) built a 40 million word USENET corpus.
- Lund and Burgess (1996) experimented with a 160 million word USENET corpus.
- Levy and Bullinaria (2001) managed to accumulate a vast 168 million word corpus from USENET.

Despite the availability of such vast bodies of written text, it may be unwise to rely on them, and their ever increasing size. If a machine learning system was developed in order to penetrate the syntax of an unknown language, it may fail if its success depends on analysing massive samples of that language. In fact, to collect 168 million words in any language other than English would be not be a trivial task.

Corpora of other *known* languages are not nearly as plentiful, or as large. USENET cannot be relied upon due to the dominant language being American-English.

For experimentation with the English language, two corpora were used, both chosen to be deliberately small (compared to modern day corpora). The first corpus was taken from a single source; an English translation of *Don Quixote de la Mancha* by Miguel de Cervantes (Ormsby 1885). An electronic version was obtained freely from Project Gutenberg. This body of text contains 426,700 words, consisting of 16,000 unique tokens. The size of this corpus, though relatively small, is still adequate to perform the distributional analysis on, and cluster effectively. The added benefit is that execution time for the whole clustering process is greatly reduced due to not handling a massive number of words.

The second corpus used was the LOB corpus. It totals one million words, of which there are approximately 50,000 unique tokens. At over twice the size of the *Don Quixote* corpus, it should provide an interesting glimpse at how the size of corpus can affect the end performance of the clustering process.

The clustering process will not attempt to cluster all of the words in the entire corpus. For the experiments performed in this work, a limit of the most frequent 500 content words was set. Such a limit may sound small relative to the number of distinct words in the corpora, however, Zipf's Law (Zipf, 1949) states that the vast majority of the words in a body of text can be accounted for from a small percentage of the highest ranked words.

For experimentation with an alternative language, Spanish was selected. Acquiring a relatively small Spanish corpus proved to be straightforward, as an electronic copy of the original Spanish version of *Don Quixote* was again obtained courtesy of Project Gutenberg. An interesting observation with the Spanish *Don Quixote* is that it is only 383,200 words in size, with a vocabulary of 24,000 words. This is roughly a difference of 43,000 words compared to the English translation. Yet, the vocabulary is 8,000 richer than the English equivalent. The variation in the word count can largely be attributed to the translator's preface which totals to over 17,000 words. The rest of the difference illustrates that there is never a direct one-to-one relationship between words from different languages. In this example, there are clearly words and phrases in Spanish that cannot be expressed in the same number of words for their English equivalent.

5.2 Factors

The following factors can be experimented with:

- Type of corpus
- Size of the corpus
- Number of function words
- Number of content words to be clustered
- Size of the window
- Metric used
- Clustering algorithm used
- Number of resulting clusters

There is clearly plenty of scope for research into how the above factors affect the overall performance of the clustering. However, due to time restrictions, only a selection of the above factors will be investigated. The English corpora for this project are the 427,000 word *Don Quixote* text and the one million word LOB corpus. The first two factors will remain static with these two corpora and their respective sizes. The number of content words will stay fixed on 500.

5.3 How to evaluate clusters

It is first necessary to describe the method of evaluation in order to understand how the performance of the clustering was measured. Without a reliable approach, it would not be possible to quantify the effects of a given factor. Hughes (1994) outlined an excellent approach to automatically evaluating the clusters his algorithms produced. He used a pre-tagged LOB corpus, where all words had been assigned an appropriate part-of-speech. However, the LOB tagset comprises of over 130 separate tags, which was too detailed, therefore, he devised the *reduced LOB tagset*. This shrinks the tagset to only 23 parts-of-speech.

Reduced Tag	**Replaced Tags**	**Type of Item**
ADJ	J*	Adjective
ADV	R*	Adverb
ART	A*	Article
CCON	CC*	Coordinating conjunction
CARD	CD*	Cardinal numeral
DET	DT* PP$*	Determiner

EX	EX	Existential *there*
EXPL	U*	Interjection
LET	Z*	Letter of the alphabet
MD	MD	Modal auxiliary verb
NEG	XNOT	Negator
NOUN	N*	Noun
ORD	O*	Ordinal numeral
OTH	&*	Foreign words, formulas
PAST	BED DEBZ BEN DOD HVD HVN VBD VBN	Past tense verb
PREP	I*	Preposition
PRES	BE BEG BEM BER BEZ DO DOZ HV HVG HVZ VB VBG VBZ	Present tense verb
PRON	P* (not PP$*)	Pronoun
PUNC	! ( ) , . … ; : ? *..	Punctuation
QUAL	Q*	Qualifier
SCON	CS*	Subordinating conjunction
TO	TO	Infinitive marker
WH	W*	WH-word

Table 5.1 – The reduced LOB tagset. Note * is a wildcard character, e.g., J* means any tag beginning with the letter J and *any* letter after.

The automatic approach to evaluate a given cluster works as follows:

1. For each word, lookup in the LOB corpus and find any tags that it has assigned to it (from the reduced tagset).
2. Determine which is the most common tag for that cluster.
3. Calculate the ratio of the number of words in the cluster that possess the most common tag, to the total number of words in the cluster. Return as a percentage.

The *Don Quixote* corpus contains words that do not appear in the tagged LOB corpus. For any word that does not appear in the LOB corpus, then it is simply assigned "*UNK*" to represent that its part-of-speech is unknown.

A powerful motive for this methodology is its automatic nature. It allows to calculate a score one an batch of clusters are generated. However, that is not sufficient alone to justify this approach. The described clustering process generates a set of clusters. The theory is that each cluster is a significant group, and therefore each cluster needs to be assessed on its own merit. Each cluster *should* be grouping words of a similar word class. Therefore, the first step is to find a way to automatically determine the appropriate overall class for the cluster. This is achieved by comparing each word against a dictionary to find the most common tag in that cluster. Once a tag is attributed to a given cluster, a simple accuracy measure is calculated to see how many of those words match the cluster tag. For example, imagine a cluster of ten words, nine were nouns and one was a verb. The cluster tag is NOUN, which results in a cluster accuracy of 90%. This accuracy measure is a fair way of determining how well the clustering algorithms group individual clusters.

The final step is to generate a score for the experiment as a whole, and this is done by taking a mean average of all the cluster scores. Again, this is a fairly standard way of expressing the overall effect of a set of constituent results.

There are some limitations to this method. For example, when clustering is forced to merge some clusters to reach the target number of groups, you may get a cluster that is clearly a product of two distinct clusters, e.g., a cluster of ten words with five closely related adverbs and five closely adjectives. This cluster will only get an accuracy of 50%. Also, the vast majority of the words processed are nouns, and so by virtue of this, many clusters of nouns could be potentially formed without any intuition by the applied algorithm. Of course, many part of speech classification systems can fall victim to (or benefit from!) this language property.

Although not consistent with similar parsing evaluation, this is because this task is different and there is a need to capture the effectiveness of the resulting clusters. Not only is this approach automatic and scalable – a huge benefit when performing a large set of experiments on the many permutations available – it's also portable to other languages. One needs to justify the tagset, which will be different per language, naturally, yet that is a perfectly valid way of determining and validating the clustering potential of this machine learning technique.

5.4 Effect of clustering algorithm

It was anticipated that the clustering algorithm would have a profound effect on the eventual clustering accuracy. Each algorithm behaves differently, therefore it was a reasonable assumption to make, and was backed up by this experiment: number of clusters = 100; number of function words = 25; window size = 12.

Corpus	**Metric**	**Algorithm**			
		Complete linkage	Group average	Weighted group average	Ward's method
LOB	Euclidian	82.2%	86.5%	83.6%	77.2%
	Manhattan	80.9%	83.5%	85.65	77.1%
Don Quixote	Euclidian	74.9%	78.2%	77.2%	77.2%
	Manhattan	73.9%	84.5%	82.7%	68.7%

Table 5.2 – The effect of the clustering algorithm on the clustering accuracy.

Perhaps more interesting than the varying nature than the algorithms themselves, is the role that the metric plays. In Fig. 5.1, the effect of the metric is illustrated clearly, and can alter the performance of the subsequent clustering considerably. There is a difference of 8.5% between the two metrics when used with Ward's method. It is not entirely obvious why this is the case. For the larger LOB corpus, Ward's method performs almost equally with either metric. In fact, differences between the metrics are much less pronounced in the LOB corpus, which suggests that the size of the corpus is influencing the overall performance.

Regardless of the metric, overall, Group Average performs the best of the four algorithms used. Closely followed by Weighted Group Average, Complete linkage and finally Ward's method. This order nearly always remains for any given cluster size, number of function words and window size. There are a few exceptions to the rule, where for example weighted group average may perform slightly better than group average.

5.5 Effect of number of function words

Due to much of the project revolving around the function words, it makes sense to investigate the role they play. Fifty function words were selected; the *n* highest ranking function words were used, and the distribution of the target content words were profiled against those *n* words. Table 5.3 shows the results from an

experiment where: number of clusters = 100; clustering algorithm = Group Average; metric = Manhattan.

Num. of function words	**DQ**			**LOB**		
	Window size			**Window size**		
	4	*8*	*12*	*4*	*8*	*12*
5	72.0%	73.9%	78.3%	77.0%	79.9%	80.6%
10	72.5%	76.5%	81.7%	80.7%	83.3%	84.7%
15	74.8%	75.5%	81.1%	81.7%	84.4%	86.35
20	75.8%	77.7%	82.7%	84.1%	85.5%	87.0%
25	75.4%	78.2%	84.5%	83.0%	85.6%	83.%
30	77.3%	78.1%	83.6%	85.1%	85.5%	86.7%

Table 5.3 – The clustering accuracy for number of function words against the window size.

The results show hat in general, for any window size, the greater the number of function words used, the better the accuracy. Initially, the performance increased substantially for each increment of 5 function words, however, it slows down by 20 function words. Performance in fact dips when the number of function words is at 25, albeit a mere one percent.

Finally, investigating the effect of the number of function words for different numbers of clusters. Unfortunately, only a weak link was found. As Fig. 5.1 shows (number of clusters = 100; window size = 12; clustering algorithm = Group Average; metric = Manhattan; corpus = *Don Quixote*), performance was rather irregular (even more so with the LOB corpus), thus a reliable conclusion as to how these two factors behave together cannot yet be obtained. It does *seem* to increase as the number of function words increases. The difference between the performance for five and thirty function words is considerable. However, with so many peaks and troughs in between, it suggests that the number of resulting clusters has a greater influence on the overall performance which is why this experiment returns vague results.

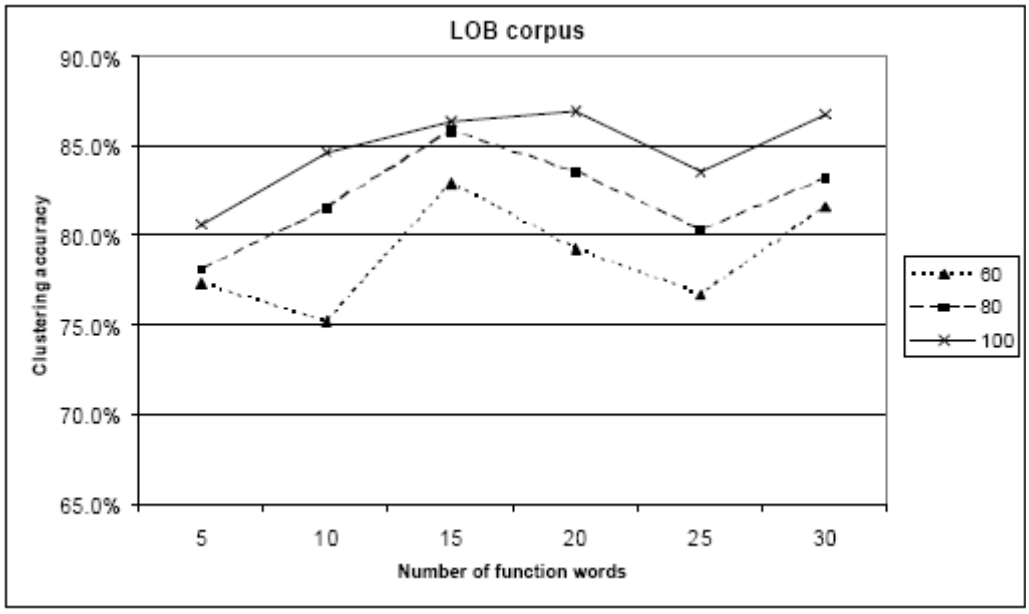

Fig. 5.1 – Graphs plotting the relationship between the number of function words and number of resulting clusters.

5.6 Effect of window size

The effect of varying the window size can be demonstrated from the following experiment, as shown in Table 5.4, where: Number of clusters = 100; number of function words = 25. The table is quite large as it shows the different window sizes for each clustering algorithm, with each metric for both corpora. Looking along each row, it is easy to see that generally, as the window size increases, the clustering performance improves.

Corpus	**Metric**	**Algorithm**	**Window size**					
			2	4	6	8	10	12
Don Quixote	Euclidian	Complete linkage	71.3	72.2	71.8	73.6	72.2	74.9
		Group average	74.1	76.0	77.5	78.4	79.2	78.2
		Weighted group av	73.5	75.3	74.7	75.6	76.5	77.2
		Ward's method	69.8	69.8	70.2	71.6	70.6	71.3
	Manhattan	Complete linkage	72.3	73.9	74.7	71.4	69.9	73.9
		Group average	74.7	75.4	75.2	78.2	79.2	79.1
		Weighted group	75.2	75.0	72.7	76.2	78.8	82.7

		av						
		Ward's method	70.4	71.0	72.4	70.4	70.5	68.7
LOB	Euclidian	Complete linkage	75.4	77.6	78.4	79.3	80.1	82.2
		Group average	81.0	81.4	84.2	85.3	87.3	86.5
		Weighted group av	80.3	81.1	83.6	84.1	82.8	83.6
		Ward's method	74.8	72.7	77.5	78.4	78.3	77.2
	Manhattan	Complete linkage	80.3	81.1	83.6	84.1	82.8	83.6
		Group average	83.6	83.0	83.9	85.6	86.7	83.5
		Weighted group av	82.7	83.1	84.9	85.2	82.6	85.6
		Ward's method	76.1	76.6	78.0	79.7	79.6	77.1

Table 5.4 – The clustering accuracy for different window sizes.

As with all the factors investigated so far, it is not a steady, linear increase. The change in accuracy can be uneven.

5.7 Effect of the number of clusters

This factor was the easiest to predict. At risk of sounding like a scratched record, it was expected that the greater the number of clusters, the better the clustering accuracy. If the number of clusters was equal to the number of content words to be clustered, then the performance of the clustering would be 100% accurate.

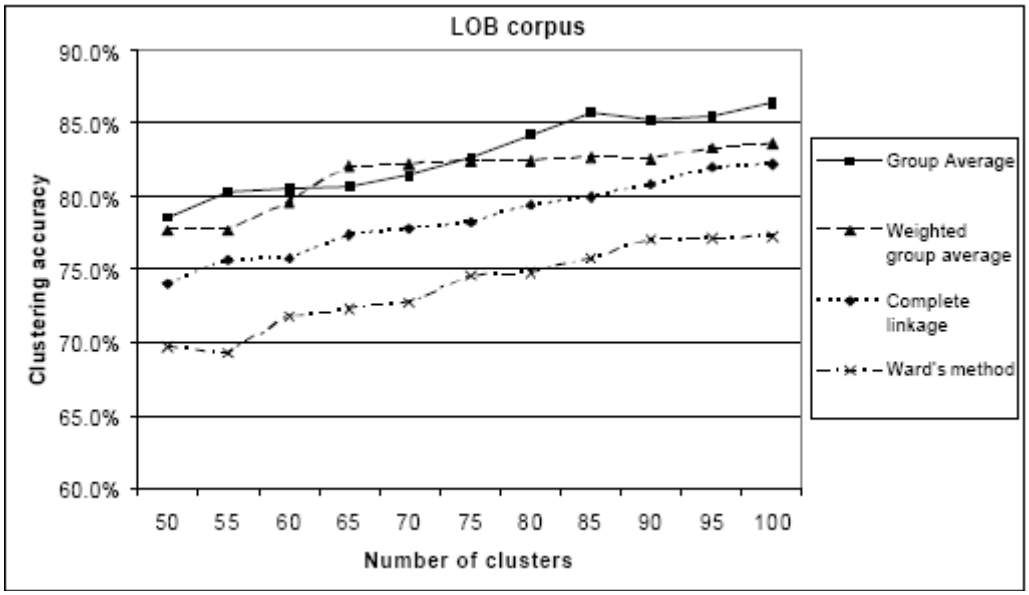

Fig. 5.2 – The relationship between number of clusters and clustering algorithm.

That is because each cluster is simply an individual word and so the method to automatically evaluate a cluster will always return 100%. Conversely, if the number of resulting clusters was simply a single cluster, then the accuracy would be very poor. The one cluster contains all the content words to be clustered, and cannot all be satisfied with a single part-of-speech that will be assigned to it by the automatic evaluation.

The prediction was confirmed with the following experiment, where: number of function words = 25; window size = 12; metric = Euclidian. Fig. 5.7 reveals the influence the resulting number of clusters has, for each of the clustering algorithms used. There is a fairly linear rise in clustering accuracy as the number of clusters increases.

5.8 Alternative language

Experimentation with the Spanish corpus was not as comprehensive as with the English corpora. Emphasis was purposely placed on understanding how the clustering process worked with English. The motivation for clustering an alternative language is to investigate whether the described method is non-language specific.

The style of experimentation remained as close to the ones performed on the English corpora. The first notable difference was the function words – Spanish equivalents were selected, although the new list was slightly longer due to many words being assigned to a gender.

The most significant difference is the method for evaluating the resulting clusters. A large tagged Spanish corpus comparable in size and detail (in terms of its

tagset) to the LOB corpus was not readily available. A tagged Cuban-Spanish corpus was found though it only contained approximately 16,000 words, with 3,000 distinct word tokens. Although only 500 words from the *Don Quixote* corpus were clustered, there were many words that didn't feature in the tagged corpus. Therefore, the tagged corpus was manually supplemented with those missing words, and their parts-of-speech were obtained from *Collins Spanish Dictionary* (Sinclair 1998). The tagset used for the corpus was small, and unfortunately did not differentiate between different types or tenses of verbs.

Unsurprisingly, the same properties identified in the English experiments were present in the Spanish experiments. That is, the window size, number of function words and number of resulting clusters improves the clustering accuracy. Group Average once again the highest performing algorithm, and the Manhattan metric consistently outperformed the Euclidian metric for the experiments carried out. These observations are summarised in Table 5.6.

Metric	Clustering algorithm	Number of clusters				
		80	85	90	95	100
Euclidian	Complete linkage	74.45	75.05	75.70	76.70	77.60
	Group average	81.15	81.55	81.93	82.00	82.33
	Weighted group average	79.86	79.72	80.91	80.62	81.06
	Ward's method	73.07	74.45	74.46	74.88	75.63
Manhattan	Complete linkage	74.96	75.73	76.32	76.41	76.23
	Group average	84.77	85.38	85.38	85.30	85.82
	Weighted group average	80.10	80.37	81.52	82.45	83.31
	Ward's method	73.75	75.16	75.12	76.56	76.87

Table 5.6 – Summary of results showing the clustering accuracy for the Spanish *Don Quixote* corpus.

5.9 Review

The aim of the project was to investigate an approach to automatically acquire word classification. After reviewing many methods for tagging, a clustering

approach was adopted due to its appropriateness for tasks that need to be *unsupervised*. The context measurement was decided to be the relative distribution of a target content word against the most frequent function words. An algorithm was implemented that could extract this distributional information from an inputted corpus. This data was converted in a vector that represented the distribution of the content word and placed into a high-dimensional vector space. A variety of metrics and clustering algorithms were implemented which could made use of the populated vector space, created a distance matrix which then established which words were similar.

A tool was developed to automatically evaluate the clusters that were produced by the clustering process. Such a tool proved to be essential due to the number of experiments that were performed. For the English language:

Factor	**Number**	**Values**
Number of corpora	2	LOB and Don Quixote
Number of cluster sizes	11	55 – 100 in increments of 5
Number of function words used		
6	5 – 30 in increments of 5	
Number of window sizes	6	2 – 12 in increments of 2
Number of metrics	2	Euclidian and Manhattan
Number of clustering algorithms	8	Single linkage, Complete linkage, Group average, Weighted group average, Median, Centoid, Centre of gravity, Ward's Method

The total number of experiments = *2* x *11* x *6* x *6* x *2* x *8* = *12,672*.

(The number of Spanish experiments only totalled to 480)

5.10 Experimental findings

With regards to investigation with English corpora:

- Clustering accuracy was higher for the LOB corpus by a mean average of 6.5%. This was expected due to the LOB corpus being over twice the size as

Don Quixote. Of course, the two corpora are quite different in style, so a direct comparison is not appropriate.

- The best clustering algorithm was found to be Group Average from the automatic evaluation for both corpora.
- The metric had varying success – its effect was dependent on other factors, e.g., the algorithm used, and the corpus too. Overall, Manhattan performed better with the smaller *Don Quixote* corpus, and Euclidian worked better with the LOB corpus. Lund and Burgess (1996) and Hughes (1994) both found Manhattan produced the better results, which fits Shepard's (1980) theory that lower values for *m* in Minkowski's general distance equation, are more suitable for extracting semantic information.
- The clustering performance improved as the number of function words used was increased.
- Clustering accuracy was generally enhanced as the window size increased. This is likely to be due to the fact that the maximum function word separation is eight words, as observed by Elliott (2000). This means that for any content word, there will always be a function word no greater than ±4 words (equivalent to a window size of eight).
- The number of resulting clusters had a great influence on the effectiveness of the clustering. The greater the number of clusters, the better the performance. As discussed however, it is important to find the right balance between the number of content words being clustered and the resulting number of clusters. There is little information to be gained when the number of clusters is close to either extreme (i.e., very small or close to the number of content words).

Even with the small *Don Quixote* corpus, the highest accuracy of the clusters managed to reach 84.7%. With the LOB corpus, an accuracy of 87.8%.· Good evidence of semantic clustering was found for both corpora. Even words such as 'a' and 'an' which could potentially cause trouble (since the words that follow immediately after are mutually exclusive) were grouped together within the same cluster.

The clustering techniques also showed promising results when the Spanish *Don Quixote* corpus was used. Although the tagged corpus was relatively small, as too was the tagset, it was still adequate for the preliminary studies into the effectiveness of clustering an alternative language. An accuracy of 85.8% was

achieved and semantic groups were frequent in the resulting clusters. This shows that the function word profiles were an effective context measure.

The experimental findings indicate that the method proposed by this report is a feasible way for automatically acquiring word classification. A high degree of success was accomplished even with comparatively small corpora. Admittedly, stronger semantic clustering was found in the larger LOB corpus, and can be found in other experiments involving vast corpora.

Chapter 6:
Conclusions

This thesis primarily looks at the complex task of unsupervised grammar inference. It also introduced a working tool that could be used in a linguistic teaching environment that implements some of the advanced techniques, albeit not full grammar learning, but on word class acquisition and automatic synonym retrieval. In doing so, the burden of traditional corpus linguistics resourcing can be substantially reduced, and will therefore show, and invigorate, the need to bring together the two research communities to combine their knowledge to create valuable tools and language resources: Unsupervised Machine Learning to extract grammatical descriptions from a corpus or text dataset, and Corpus Linguistics is all about using corpora in linguistic research; so on the face of it the two fields should be related.

The thesis starts with a review of grammatical Inference research, and an overview of Corpus Linguistics research as represented at the International Conference on Corpus Linguistics. Part 1 presents a review of Unsupervised Grammar Inference Systems for Natural Language. Part 2 presents a snapshot of current research topics in Corpus Linguistics, as represented at the International Conference on Corpus Linguistics. It transpires that GI researchers focus on development of novel algorithms and have little or no interest in standardisation of corpus resources; and Corpus Linguists are unaware of GI as a potential tool for their research. To introduce the two research communities to each other, this thesis presents an application of Corpus Linguistics principles to GI research, and an application of GI to a Corpus Linguisrtics research tool.

Part 3 presents the use of corpora for automatic evaluation of Grammar Inference systems. The ‘looks good to me’ approach, despite the critical slant within this study, is not an inferior one. However, it will not meet the demands for robust NLL evaluation. Which is why corpus based approaches have been presented as the most feasible and reliable way of essentially trying to emulate ‘looks good to me’ whilst eliminating bias and providing consistency. There is a great need for researchers within the field to begin addressing accurate evaluation techniques. The current ‘gold standard’ method (as described in section 3.1) is too basic and will

simply favour GI systems that behave similar to the way the 'gold standard' treebank was parsed. Greater flexibility is required, and a method like the multi-multi-annotated corpus (see section 3.3) will be a beneficial innovation to do just that. Instead of replacing "looks good to me", the two approaches should be combined, so that evaluation is no longer a simple quest for a single "score", but instead an exploration of strengths (and weaknesses) of GI systems.

Part 4 presents aConCorde: an open-source, extendable concordancer for Arabic. This study has summarised many of the issues regarding the difficulty of Arabic concordance. The *aConCorde* project attempts to resolve some of the core issues, for example, interactive searching that displays the Arabic text correctly and stem-based searching. However, at the time of writing, *aConCorde* is still limited in terms of its features compared to the more established products on the market, these include its handling of: Mark-up; Full context; Arabic root/stem database; Clustering integration; Computational resources. Naturally, the development of *aConCorde* will continue, and the limitations mentioned here are high priority issues that will be addressed in future issues. However, *aConCorde* has already made significant progress. The development of a simple user-interface with novel exploration features mean that it is well suited to a learning environment. With the foundation firmly laid, we can focus on other tasks in future, for example, improving and creating fresh approaches to intuitive morphological searching, or improving word similarity exploration through machine learning methods such as clustering. We also hope that the established concordance tools will realise the demand for products that cope with Arabic script, and adapt their software accordingly.

Part 5 looked more thoroughly at clustering introduced in part 4. This time various algorithms and parameters were experimented with to test whether clustering was sufficient in the task of automatically segmenting a lexicon in to groups which not only share a similar high-level part of speech, but also share a semantic similarity to the point which would be of interest in an automatic synonym inference task. The results were very encouraging, and the enhanced vector modelling appeared to pay off compared to the simpler bigram counts of earlier approaches by others in the field.

In conclusion, CL provides principled methods for evaluation of GI, and useful applications for GI; so GI and CL research communities should work together. This thesis should contribute to increasing mutual awareness and cooperation.

References

Adriaans, P.W. 1992. Language Learning from a Categorial Perspective. Ph.D. thesis, Unversiteit van Amsterdam.

Adriaans, P.W. 1999. Learning shallow context-free languages under simple distributions. ILLC Report PP-1999-13, Institute for Logic, Language and Computation, Amsterdam.

Ajdukiewicz K. 1935. Die syntaktische Konnexiät. *Studia Philosophica* 1:1-27.

Al-Sulaiti, L. 2004. Designing and Developing a Corpus of Contemporary Arabic. Masters thesis, School of Computing, University of Leeds, UK.

Al-Sulaiti, L. and Atwell, E. 2006. 'The design of a Corpus of Contemporary Arabic', International Journal of Corpus Linguistics, 11 (2), pp. 135-171.

Archer, Dawn, , Paul Rayson, Andrew Wilson and Tony McEnery (eds.) 2003. Proceedings of the Corpus Linguistics 2003 conference. UCREL technical paper number 16. UCREL, Lancaster University

Arlund, Pamela. 2006. An acoustic, historical and developmental analysis of Sarikol Tajik diphthongs. PhD thesis, University of Texas at Arlington.

Aston, G. and Burnard, L. 1998. The BNC Handbook: Exploring the British National Corpus with SARA. Edinburgh: Edinburgh University Press.

Atwell, Eric. 1983.. Constituent Likelihood Grammar. ICAME Journal, 7, 983, pp. 34-65.

Atwell, Eric. 1987. A parsing expert system which learns from corpus analysis. in Meijs, W, (editor), Corpus Linguistics and Beyond: Proceedings of the ICAME 7th International Conference on English Language Research on Computerised Corpora, pp227-235, Amsterdam, Rodopi.

Atwell, Eric, and Drakos, Nicos. 1987. Pattern recognition applied to the acquisition of a grammatical classification system from unrestricted English text. In Maegaard, B. (Ed.), Proceedings of the Third Conference of European Chapter of the Association for Computational Linguistics , pp. 56-63.

Atwell, Eric. 1996. Comparative evaluation of grammatical annotation models. In Sutcliffe R, Koch H, McElligott (eds), *Industrial parsing of software manuals*. Amsterdam: Rodopi, pp 25-46.

Atwell E, Demetriou G, Hughes J, Schiffrin A, Souter C, Wilcock S. 2000. A comparative evaluation of modern English corpus grammatical schemes. *ICAME Journal*, 24:7-23.

Atwell, Eric, and Elliott, John. 2001. A corpus for interstellar communication. In Rayson, P, Wilson, A, McEnery, T, Hardie, A, and Khoja, S. (Eds.), Proceedings of CL2001: International Conference on Corpus Linguistics. UCREL Technical Paper 13, Lancaster University, 2001, pp. 31-39.

Atwell, Eric. 2003. A New Machine Learning Algorithm for Neoposy: coining new Parts of Speech. in: Archer, D, Rayson, P, Wilson, A and McEnery, T (editors) Proceedings of CL2003: International Conference on Corpus Linguistics, pp.43-47.

Baker, J.K. 1975. Stochastic Modelling for Automatic Speech Understanding. In D.R. Reddy (Ed) - Speech Recognition. Academic Press. New York.

Baker, J. K. 1979. Trainable grammars for speech recognition. In D.H. Klatt and J.J. Wolf (Eds) - Speech communication papers for the 97th meeting of acoustical society of America, pp547-550.

Bates, E. and Goodman, J.C. 1997. On the Inseparability of Grammar and the Lexicon: Evidence from Acquisition, Aphasia, and Real-time Processing. Language and Cognitive Processes, 12, 1997, pp. 507-584.

Belkin, M. and Goldsmith, J. 2002. Using eigenvectors of the bigram graph to infer morpheme identity. Proceedings of the Morphology/Phonology Learning Workshop of ACL-02. Association for Computational Linguistics.

Beesley, K. 2003. Xerox Arabic Morphological Analyzer Surface-Language (Unicode) Documentation. Xerox Research Centre Europe.

Bod, R. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st international Conference on Computational Linguistics and the 44th Annual Meeting of the Association For Computational Linguistics* (Sydney, Australia, July 17 - 18, 2006). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 865-872.

Boualem, M., Leisher, M. and Ogden, B. 1999. 'Concordancer for Arabic' in Arabic Translation and Localisation Symposium, Tunis, URL: http://crl.nmsu.edu/~mleisher/concord.pdf

Brill E. 1993. Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of the 31st annual meeting of the Association for Computational Linguistics (ACL)*, Columbus, Ohio, USA, pp 259-265.

Briscoe, E.J. 2000. Grammatical Acquisition: Inductive Bias and Coevolution of Language and the Language Acquisition Device. Language, 76(2).. pp. 245-296.

Buckwalter, T. 2002. Buckwalter Arabic transliteration. URL: http://www.qamus.org/transliteration.htm

Burnard, L. 2004. 'BNC-Baby and Xaira' in Proceedings of the Sixth Teaching and Language Corpora conference, p. 84, Granada.

Burnard, L. and Dodd, T. 2003. 'Xara: an XML aware tool for corpus searching' in D. Archer, P. Rayson, A. Wilson and T. McEnery (eds.) Proceedings of the Corpus Linguistics 2003 Conference, pp. 142–44. University of Lancaster: UCREL.

Clark, A., Eyraud, R., and Habrard, A. 2008. A Polynomial Algorithm for the Inference of Context Free Languages. In *Proceedings of the 9th international Colloquium on Grammatical inference: Algorithms and Applications* (Saint-Malo, France, September 22 - 24, 2008). A. Clark, F. Coste, and L. Miclet, Eds. Lecture Notes In Artificial Intelligence, vol. 5278. Springer-Verlag, Berlin, Heidelberg, 29-42.

Cobb, T., Greaves, C. and Horst, M. 2001. 'Can the rate of lexical acquisition from reading be increased? An experiment in reading French with a suite of online resources' in P. Raymond and C. Cornaire (eds.) Regards sur la didactique des langues secondes. Montréal: Éditions Logique.

Daelemans, W, Van den Bosch, A and Zavrel, J. 1999. Forgetting exceptions is harmful in language learning. Machine Learning, 11. pp. 11-43.

Daum, Michael, Kilian Foth, and Wolfgang Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC2004), Lisbon, Portugal.

Déjean H. 2000. ALLiS: a symbolic learning system for natural language learning. In Cardie C, Daelemans W, Nédellec C, Tjong Kim Sang E (eds),

Proceedings of the fourth conference on computational natural language learning and of the second learning language in logic workshop. Lisbon, Portugal, pp 95-98.

Dempster, A., N. Laird and D. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of the Royal Statistical Society, vol 39, 1-38.

de Roeck, Anne. 2002. 'Arabic for the absolute beginner', ELRA Newsletter no.7

Dodd, B. 1997. 'Exploiting a corpus of written German for advanced language learning' in A. Wichmann, S. Fligelstone, G. Knowles and T. McEnery (eds.) Teaching and Language Corpora, pp. 131–45. London: Longman.

Dodd, T. 1997. SARA: Technical Manual.

URL: http://www.natcorp.ox.ac.uk/sara/TechMan/

Domingos, P. 1995. The RISE 2.0 system: A case study in multistrategy learning. Technical Report 95-2, Department of Information and Computer Science, University of California.

Dong, S and Searls, D.B. 1994. Gene Structure Prediction by Linguistic Methods. Genomics.

Drayson, Michael. 2007. Open-source application for web-trawler and knowledge discovery. BSc Final Year Project dissertation, University of Leeds.

Elliott, J., E. Atwell and W. Whyte. 2000. *Increasing Our Ignorance of Language: Identifying language Structure In An Unknown Signal*. In*: Proceedings of 4th International Conference on Computational Natural Language Learning*, pp 25-30. Association of Computational Linguistics. New Jersey.

Elliott, J, Atwell, E and Whyte, W. 2001. Visualisation of Long Distance Grammatical Collocation Patterns in Language. In IV2001: 5th International Conference on Information Visualisation, London, UK.

Everitt, B. 1993. Cluster Analysis (third edition). London: Edward Arnold.

Eyraud, R., Higuera, C., and Janodet, J. 2007. LARS: A learning algorithm for rewriting systems. Machine Learning. 66, 1, pp7-31.

Finch, S. 1993. Finding structure in language. PhD thesis, Edinburgh University.

Finch S, Chater N 1992. Bootstrapping syntactic categories using statistical methods. In Daelemans W, Powers D (eds), *Backgrounds and experiments in machine learning and natural language: Proceedings first SHOE workshop.* Institute for Language Technology and AI, Tilburg University, pp 230-235

Finch, S. and Chater, N. 1992. 'Bootstrapping syntactic categories' in Proceedings of the 14th Annual Meeting of the Cognitive Science Society, pp. 820–25. Hillsdale, New Jersey.

Forst, Martin.. 2003. Treebank conversion - creating an f-structure bank from the TIGER Corpus. In Proceedings of the International Lexical Functional Grammar Conference (LFG 2003), Saratoga Springs, NY.

Freltag, D. 1997. Using Grammatical Inference to Improve Precision in Information Extraction. In Working Papers of the ICML-97 Workshop on Automata Induction, Grammatical Inference and Language Acquisition.

Fussiger, Leandro. 2007. NLPSEARCH Framework para integracao de sistemas de PLN e API's de mecanismos de busca. Research Report, Ciencia da Computacao, Centro Universitario Feevale, Brazil.

Gold, E.M. 1967. Language Identification in the Limit. Information and Control. 10. pp. 447-474.

Graddol, D. 1997. The Future of English? London: British Council.

Grönqvist, Leif, and Magnus Gunnarsson. 2003. A method for finding word clusters in spoken language. in: Archer, D, Rayson, P, Wilson, A and McEnery, T (editors) Proceedings of CL2003: International Conference on Corpus Linguistics, pp.265-273.

Heddaya, A. 1985. Qalam: A convention for morphological Arabic-Latin-Arabic transliteration. URL: http://eserver.org/langs/qalam.txt Henrichsen, P.J. GraSp: Grammar Learning from unlabelled speech corpora. In: Roth, D and Van den Bosch, A. (Eds), Proceedings of CoNLL-2002, Taipei, Taiwan, 2002, pp. 22-28.

Henrichsen, P J. 2002. GraSp: Grammar learning from unlabelled speech corpora. In Roth D, van den Bosch A (eds), *Proceedings of CoNLL-2002*. Taipei, Taiwan, pp 22-28.

Hong, T.W and Clark, K.L. 2001. Using Grammatical Inference to Automate Extraction from the Web. In Principles of Data Mining and Knowledge Discovery. pp. 216-227.

Hoogland, J. 2003. The Nijmegen Arabic/Dutch dictionary project using the concordance program.

URL: http://www.let.kun.nl/wba/Content2/1.4.6_Concordancing.htm

Horn, D., Z. Solan, E. Ruppin and S. Edelman. 2004. Unsupervised language acquisition: syntax from plain corpus, presented at the Newcastle Workshop on Human Language.

Hughes, J. 1994. Automatically Acquiring a Classification of Words. Ph.D. thesis, School of Computing, University of Leeds.

Hughes, J and Atwell, E. 1994. The automated evaluation of inferred word classifications. In Cohn, A. (Ed), Proceedings of ECAI'94: 11th European Conference on Artificial Intelligence. John Wiley, pp535-539.

Ibrahimov, O., Sethi, I. and Dimitrova, N. 2001. 'Clustering of imperfect transcripts using a novel similarity measure' in Proceedings of the SIGIR'01 Workshop on Information Retrieval Techniques for Speech Applications.

Johns, T. 1990. 'From printout to handout: Grammar and vocabulary teaching in the context of data-driven learning', Computer Assisted Language Learning, 10, pp. 14–34.

Jurafsky, D. and Martin, J. 2000. Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Upper Saddle River NJ: Prentice-Hall Pearson.

Khoja, S. 2003. APT: An Automatic Arabic Part-of-Speech Tagger. Ph.D. thesis, Computing Department, Lancaster University, UK.

Kirby, S. 2002. Natural Language from Artificial Life. Artificial Life, 8(2). pp.185-215.

Kirby, S and Hurford, J. 2002. The emergence of linguistic structure: an overview of the iterated learning model. In: Cangelosi, A and Parisi, D (Eds.) Simulating the Evolution of Language. pp. 121-148.

Kiss, G. R. Grammatical Word Classes: A Learning Process and its Simulation. Psychology of Learning and Motivation. 7. pp1-41. 1972

Kubler, Sandra, Wolfgang Maier, Ines Rehbein, Yannick Versley. 2008. How to Compare Treebanks. Proceedings of LREC'2008.

Lawler, J. 2000. 'Review of MonoConc Pro 2.0 concordancing software', Linguist, 11 (1411).

Leech, G, Barnett, R and Kahrel, P. 1996. EAGLES Final Report and guidelines for the syntactic annotation of corpora. European Expert Advisory Group on Language Engineering Standards (EAGLES) Report EAG-TCWG-SASG/1.5.

Leech, G. and Fligelstone, S. 1992. 'Computers and corpus analysis' in C. Butler (ed.) Computers and Written Texts, pp. 115–40. Oxford: Blackwell.

Levy, J.P. and J.A. Bullinaria. 2001. *Learning Lexical Properties from Word Usage Patterns: Which Context Words Should be Used?* In: R.F. French & J.P. Sougne (Eds) - *Connectionist Models of Learning, Development and Evolution: Proceedings of the Sixth Neural Computation and Psychology Workshop,* 273-282. London: Springer.

Liang, P., S. Petrov, M.I. Jordan and D. Klein. 2007. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL).

Losee R M. 1996. Learning syntactic rules and tags with genetic algorithm for information retrieval and filtering: An empirical basis for grammatical rules. Information processing & management, 32(2):185-197.

Ludeling, Anke, and Merja Kyto. 2008. Corpus Linguistics: an International Handbook. Volume 1. 776 pages. De Gruyter, Berlin.

Lund, K. and C. Burgess. 1996. *Producing high-dimensional semantic spaces from lexical cooccurrence*, Behavior Research Method, Instruments, & Computers 28(2), pp 203-208.

Maamouri, M., Bies, A., Buckwalter, T. and Mekki, W. 2004. 'The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus' in NEMLAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

Madeira, S.C. and Oliveira, A.L. 2004. Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. on Comp. Biol. and Bioinformatics (1).

McEnery, Tony and Zhonghua Xiao. 2003. Fuck revisited. in: Archer, D, Rayson, P, Wilson, A and McEnery, T (editors) Proceedings of CL2003: International Conference on Corpus Linguistics, pp.504-512.

Ogden, B. and Bernick, P. 1996. 'Oleda: User-centered Tipster technology for language instruction' in Proceedings of the Tipster Pharse II 24 Month Workshop, VA, USA. *aConCorde: A concordancer for Arabic* 57

Miguel De Cervantes. 1885. *Don Quixote de la Mancha*. 1665. Trans. John Ormsby.

Paskin, M.A. Grammatical Bigrams. In Dietterich, T, Becker, S, and Gharahmani, Z (eds.). 2001. Advances in Neural Information Processing Systems 14. Cambridge, MA: MIT Press.

Rayson, Paul, Andrew Wilson, Tony McEnery, Andrew Hardie and Shereen Khoja (eds). 2001**.** Proceedings of the Corpus Linguistics 2001 conference. UCREL technical paper number 13. UCREL, Lancaster University

Redington, Martin, Nick Chater and Steven Finch. Distributional Information: A Powerful Cue for Acquiring Syntactic Categories. Cognitive Science, Vol 22 (4), pp 425-469. Cognitive Science Society. 1998.

Roberts, A. 2002. Automatic acquisition of word classification using distributional analysis of content words with respect to function words. Technical report, School of Computing, University of Leeds.

Roberts, A. and Atwell, E. 2005. ‘aConCorde: Towards a proper concordance of Arabic’ in P. Danielsson and M. Wagenmakers (eds.) Proceedings of the Corpus Linguistics 2005 Conference, University of Birmingham, UK.

Scott, M. 2004. WordSmith Tools 4.0.

URL: http://www.lexically.net/downloads/version4/html/index.html

Sethuraman, J. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Shepard, Roger N. 1980. *Multidimensional scaling, tree-fitting, and clustering.* Science, 210, pp390-398.

Simov, Kiril and Petya Osenova (eds.) 2003. Proceedings of the The Workshop on Shallow Processing of Large Corpora (SProLaC 2003) held in conjunction with the Corpus Linguistics 2003 conference. UCREL technical paper number 17. UCREL, Lancaster University.

Sinclair, J.M. 1987. Looking up; an Account of the COBUILD Project. London: Collins ELT.

L. Sinclair (ed). 1998 Collins Spanish Dictionary, Plus Grammar. HarperCollins, Glasgow.

Smith, Nicholas, Sebastian Hoffmann and Paul Rayson. 2008. Corpus Tools and Methods, Today and Tomorrow: Incorporating Linguists' Manual Annotations. Literary and Linguistic Computing journal, vol.23(2), pages 163-180.

Solan, Z. 2006. Unsupervised Learning of Natural Languages. PhD Thesis. Tel-Aviv University.

Solan, Z., D. Horn, E. Ruppin and S. Edelman. 2005. Unsupervised learning of natural languages, in Proc. Natl. Acad. Sci., 102.

Steedman, M. Constituency and Coordination in a Combinatory Grammar. In: Baltin, M.R and Kroch, A.S (Eds.), Alternative Conceptions of Phrase Structure. University of Chicago. 1989. pp. 201-231.

Sutcliffe R, Koch H, McElligott (eds). 1996. Industrial parsing of software manuals. Amsterdam: Rodopi

Tu, K. and Honavar, V. 2008. Unsupervised Learning of Probabilistic Context-Free Grammar using Iterative Biclustering. In *Proceedings of the 9th international Colloquium on Grammatical inference: Algorithms and Applications* (Saint-Malo, France, September 22 - 24, 2008). A. Clark, F. Coste, and L. Miclet, Eds. Lecture Notes In Artificial Intelligence, vol. 5278. Springer-Verlag, Berlin, Heidelberg, 224-237.

Van den Bosch, A. 1999. Careful abstraction from instance families in memory-based language learning. Journal of Experimental and Theoretical Artificial Intelligence, 11:3, special issue on Memory-Based Language Processing, Daelemans, W, guest ed.. pp. 339-368.

Van Zaanen, M and Adriaans, P.W. 2001. Comparing Two Unsupervised Grammar Induction Systems: Alignment-Based Learning vs. EMILE. Technical Report: TR2001.05, School of Computing, University of Leeds.

Van Zaanen, M. 2002. Bootstrapping Structure into Language: Alignment-Based Learning. PhD Thesis, School of Computing, University of Leeds..

Vervoort, M.R. 2000. Games, Walks and Grammars. Ph.D thesis, Universiteit van Amsterdam.

Watkinson, S and Manandhar, S. 2001. A Psychologically Plausible and Computationally Effective Approach to Learning Syntax, CoNLL'01, the Workshop on Computational Natural Language Learning, ACL/EACL 2001.

Watkinson, S and Manandhar, S. 2001. Translating treebank annotation for evaluation. In: Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL/EACL 2001.

Wiechmann, Daniel, and Stefan Fuhs. 2006. Concordancing software. Corpus Linguistics and Linguistic Theory journal, vol 2-1, pp. 109-130.

Worboys, Thomas. 2007. Integration of JBootCat and aConCorde for web mining. BSc Final Year Project dissertation, University of Leeds.

Wood, M. 1993. Categorial Grammars. Routledge. London.

Zernik, U. 1991. 'Tagging word senses in corpus' in U. Zernik (ed.) Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon, pp. 91–112. Hillsdale, NJ: Lawrence Erlbaum.

Zipf, G.K. 1949. *Human Behaviour and the Principle of Least Effort*. Addison Wesley Press. New York.

Zupan, J. 1982. Clustering of Large Data Sets. Chichester: John Wiley and Sons.

Appendix

The University of Leeds *Regulations for and advice on the format for the presentation of theses and dissertations for higher degrees* requires that: " … where a student has used work from a solely or jointly-authored publication to form part of a chapter of the thesis then s/he **must** also include the work, as published, bound into the thesis as an Appendix at the end." The appendix includes the following:

Roberts, Andrew; Atwell, Eric. Unsupervised Grammar Inference Systems for Natural Language. Research Report number 2002.20, School of Computing, University of Leeds. 2002.

Roberts, Andrew. CL2003: the International Conference on Corpus Linguistics. ELSnews: The Newsletter of the European Network in Human Language Technologies, vol. 12.2, pp. 6-8. 2003.

Roberts, Andrew; Atwell, Eric. The use of corpora for automatic evaluation of grammar inference systems in: Archer, D, Rayson, P, Wilson, A & McEnery, T (editors) Proceedings of CL2003: International Conference on Corpus Linguistics, pp. 657-661 Lancaster University. 2003.

Roberts, Andrew; Al-Sulaiti, Latifa; Atwell, Eric. aConCorde: Towards an open-source, extendable concordancer for Arabic. Corpora, vol. 1, pp. 39-57. 2006.

University of Leeds
SCHOOL OF COMPUTING
RESEARCH REPORT SERIES
Report 2002.20

Unsupervised Grammar Inference Systems for Natural Language

by

Andrew Roberts[1] **& Eric Atwell**[2]

December 2002

[1] andyr@comp.leeds.ac.uk
[2] eric@comp.leeds.ac.uk

Abstract

In recent years there have been significant advances in the field of Unsupervised Grammar Inference (UGI) for Natural Languages such as English or Dutch. This paper presents a broad range of UGI implementations, where we can begin to see how the theory has been put in to practise. Several mature systems are emerging, built using complex models and capable of deriving natural language grammatical phenomena. The range of systems is classified into: models based on Categorial Grammar (GraSp, CLL, EMILE); Memory Based Learning models (FAMBL, RISE); Evolutionary computing models (ILM, LAgts); and string-pattern searches (ABL, GB). An objectively measurable statistical comparison of performance Of the systems reviewed is not yet feasible. However, their merits and shortfalls are discussed, as well as a look at what the future has in store for UGI.

1 Introduction

Gold's seminal paper (Gold, 1967) showed that it was theoretically impossible to extract a definitive grammar from examples of a target language, unless selected negative counterexamples were also available. Encouragingly, this theoretical hurdle has not deterred research: the natural language learning (NLL) community has witnessed rapid advances in unsupervised grammar inference.

Interesting developments have arisen from the psychological perspective of this task: research has been driven to devise psychologically plausible models of natural language acquisition.

Grammar inference is not just restricted to its classical domain of syntactic pattern recognition, with many useful functions within other levels of Natural Language Processing, such as speech processing. It has expanded in to other important areas, for example, information retrieval (Freltag, 1997; Hong and Clark, 2001) and gene analysis (Dong and Searls, 1994).

This paper focuses on recent UGI implementations. Some are pitched as UGI systems in their own right, others could be classed as solutions to subtasks that would be useful to language learning systems. It is worth noting that this is not a comprehensive review of every such system, but more of a snapshot of some of the interesting avenues of research being explored. Highly supervised systems, regardless of their performance, have not been included, nor have visualisation techniques that make it easier for human experts to discover grammar structure from text (Belkin and Goldsmith, 2002; Elliott et al, 2001). We also exclude systems which only infer word-classifications without attempting to learn structure, such as (Atwell, 1983; Hughes and Atwell, 1994; Roberts, 2002).

2 Categorial Grammar

A categorial grammar is a simply a grammar rather than a learning paradigm. However, CG clearly lends itself to unsupervised learning as it has been adopted

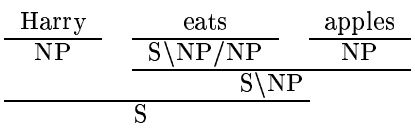

Figure 1: A simple sentence parsed in CG

as the foundation for many systems. One of the main reasons for this that the lexicon and the grammar are acquired in the same task; the psychology literature (Bates and Goodman, 1997) suggests that the two are not separate mental processes. This is a bonus for those researchers striving to produce a realistic psychological model of human language acquisition, and also for those who wish to implement simpler and more efficient algorithms for what is still a complex task.

CGs were first proposed by Ajdukiewicz (1935) and have since matured and modified. Steedman's generalised version (Steedman, 1989) serves well for a brief overview. A CG comprises of two components. Firstly, the *categorial lexicon* is essentially a dictionary which associates each word within the lexicon a syntactic and semantic category. Secondly, the *combinatory rules* provide the functional application of the grammar, and allow more complex categories to be created from the simpler ones.

$$\begin{aligned} S/b &\longrightarrow a \\ S\backslash a &\longrightarrow b \end{aligned}$$

These operators provide the freedom to transform rules, allowing you to isolate and manipulate parts that would otherwise be inaccessible (Adriaans, 1999). Fig. 1, taken from Steedman (1989), illustrates how a simple sentence is parsed.

2.1 GraSp

GraSp (Henrichsen, 2002) is a learning algorithm designed specifically for inducing grammars from large, unlabelled corpora. Its long term goal is to provide insight to the innateness debate. In this instance, the hypothesis is that there is no such linguistic innateness.

Henrichsen used a variant of the Gentzen-Lambek categorial grammar, which was enhanced with non-classical rules for isolating a residue of uninterpretable sequent elements. Empty categories are also permitted in this version which are not normally allowed in CG due to the *principle of adjacency*: combinatory rules may only apply to entities which are linguistically realised and adjacent (Steedman, 1989).

The learning algorithm begins by assigning each word type with its own unique category. The learning process applies changes in the lexicon by adding, removing and manipulating the basic categories using the CG operators (/,

\ or *). The changes are guided by a measure of disorder. $Dis(\Sigma)$ returns the number of uninterpretable atoms in the sequent Σ. The update process is iterative. GraSp monitors the measure of disorder before applying each update, and the process will halt as when the update no longer improves the disorder of the lexicon.

No quantifiable measurements of GraSp's accuracy were published. Therefore, commenting on its performance is obviously difficult. Whilst rigorous evaluation may not have taken place, GraSp clearly has many merits in that it does succeed in learning linguistic features from unlabelled corpora. Henrichsen describes the output being rich in "microparadigms and microstructure", which inter-connect to form a complex grammar.

2.2 CLL

CLL (Watkinson and Manandhar, 2001) is not only concerned with developing a computationally feasible language learner, but one that is also psychologically plausible too. Therefore, the algorithm used by CLL was designed to also make way for a model of human language learning facilities, as well as being a computational learning tool.

CLL is trying to emulate a child with respect to its acquisition of its first language. The psychology influence in the research refers mainly to the environment in which the learner learns. This deals with the type of language a child is likely to encounter and the effect of language teaching. The conclusion reached: "Hence, we have a learner that is unsupervised, positive only and does not have a teacher." Unfortunately, the algorithm was arguably built with too much 'innateness', which reduces its credibility as an unsupervised process. The provision of a complete set of lexical categories was quite justly acknowledged by the authors as being "too strong a bias to be psychologically plausible". Additionally, the algorithm is given a set of closed-class words (with categories) at the start of the learning process. Two different sizes of the initial lexicon were tried, 31 and 348.

The learning algorithm functions by taking an example sentence from a corpus, which is then parsed using a n-best probabilistic chart parser (developed from a standard stochastic CKY algorithm). This can result in a number of possible parses, of which it is then up to the parse selector to decide which one would benefit the lexicon the most. The metric which decides the 'goodness' of a parse is based on which creates the most compressive lexicon. To do this, it must also evaluate the effect of the newly modified lexicon by reparsing any examples that may be affected. Whilst it appears to be a costly approach, it does ensure the most compressive lexicon.

Watkinson and Manandbar created a relatively robust approach to evaluating their results (Watkinson and Manandhar, 2001). As the Penn Treebank corpus was the source of the text to learn, its annotation was translated into CG annotation, so that it could be compared with the output of the learning algorithm. Whilst the newly annotated corpus was considered a gold standard, it was converted automatically, and therefore liable to error. The best perfor-

mance attained by CLL was 51.9% accuracy (this is with an initial 348 word lexicon). While this performance is still relatively low, considering the difficulty of the problem, and using a complex corpus, to perform above 50% is still a recognisable achievement.

2.3 EMILE

EMILE (Adriaans, 1992, 1999) has been around for some time now. It will continue to be with us because it is a well executed algorithm and performs well and efficiently. EMILE has been updated through the years, and is currently at version 4.1 — although this latest version has been implemented by Vervoort (2000).

For EMILE to be in this section, it clearly relies on a categorial grammar. A given input set of example sentences is converted into a CG of basic categories. After applying first order explosion, each sentence is examined to discover how it can be broken up into subexpressions (using the standard CG operators). The resulting set of subexpressions is passed to an oracle. The reason for this is because EMILE uses a teacher/child metaphor. Therefore, the system can ask the oracle which ones are valid.

Any subexpressions that can be substituted into the same contexts, and still be valid are said to be of the same type. Therefore, the next step employed is to cluster to rules passed by the oracle and cluster them into types. The final phase is rule induction. The clustering has resulted in a variety of basic and complex rules, however, they tend to relate to specific types. Thus the rule induction step generalises them to general types, with the outcome being a shallow context-free grammar.

EMILE tends to produce accurate results due to the fact that it waits for enough evidence to be found before constructing grammar rules. However, according to Adriaans' calculations, in order for his system to acquire a language with 50,000 words, it would need learn from a sample of 5 million sentences. Assuming an average of 15 words per sentence, then a 75 million word corpus is required. My initial thoughts was that figure was too large. However, it is quite reasonable considering EMILE is generating a grammar to cope with 50,000 words considering the complexity of the task. It does mean that EMILE is a slow learner (compared to some other systems, such as ABL), as was concluded in van Zaanen and Adriaans (2001). Experiments conducted on the ATIS corpus produced precision of 51.6%.

3 Memory Based Learning (MBL)

The MBL paradigm attempts to discover ways in which you can abstract information from a given data set, whilst maintaining accuracy. The hope is to develop methods that perform at least equal to pure MBL (where all information is retained).

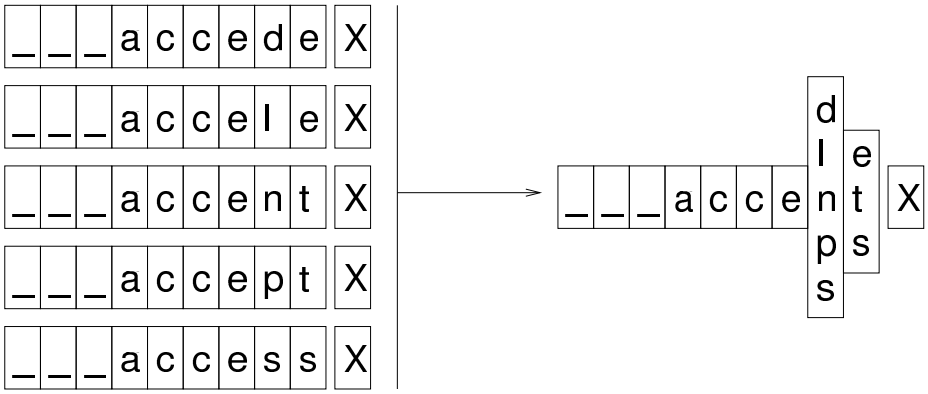

Figure 2: *Example of family creation in FAMBL taken from Van den Bosch (1999). This shows instances of grapheme-phoneme occurrences being merged into a single family expression.*

Pure MBL tends to give the best performance in analysis of unseen data, due to the very fact that it never forgets any training examples. An advantage of computerised systems is that it is often feasible to keep every instance - memory limits are seemingly infinite. So, why bother with MBL? Abstraction should result is smaller and more efficient learning models. After all, humans do not have a capacity for pure MBL, therefore, MBL should offer a more psychologically plausible approach too.

3.1 FAMBL (Family Based Learning)

Classification systems that are equipped with a forgetting facility, use it to not only perform more efficiently, but to avoid over-fitting, which can be detrimental to accuracy. However, default parameters for this process often generalise too much for learning tasks which produces poor performance (Van den Bosch, 1999).

Therefore, the key is to use careful (or weak) abstraction, whereby instances can be abstracted without doing harm (e.g., forgetting exceptions would be careless (Daelemans et al, 1999)). The way in which FAMBL achieves this is to transform the instance base into *instance families*. A family is a cluster that has been classified using k-NN (Nearest Neighbour). The instances surround a given instance are its family. Fig. 2 gives an example of how the instances are then merged into hyper-rectangles that define a family expression.

The FAMBL algorithm randomly selects instances individually, that are not a member of a family. For each one found, its family is determined and a family expression is created from those instances. It continues to generate families until the instance base doesn't contain any instances that do not belong to a family.

FAMBL has not yet been used as a full-scale grammar induction system. It has, however, been applied successfully to a variety of relevant tasks in language

learning, including morphological segmentation, base-NP chunking, PP attachment, and POS tagging. For example, the experimentation with POS resulted in an accuracy of 97.87%, and family-abstraction yielded a reduction of 75% memory compared to pure-MBL.

3.2 RISE (Rule Induction from a Set of Exemplars)

RISE (Domingos, 1995) is a multi-strategy approach, which comprises of MBL and rule induction. Rules begin as being instance specific. It then begins to generalise by looking at each rule and searching for other instances that fall within the same class. Any instances that satisfy this are merged and their rules are generalised.

In order to ensure the rules deduced are productive, RISE estimates the 'goodness' by computing its apparent accuracy using its class prediction strength with Laplace correction. Performance is constantly monitored by the algorithm and generalisation ceases if the apparent accuracy worsens.

4 Artificial Life: Evolutionary Optimisation

Nature has allowed humans to acquire the abilities to learn, understand and communicate using language by process of evolution. With that precedent, it should therefore by possible for us to apply similar techniques to create *Language Acquisition Devices*: LADs. Of course, its feasibility is the matter of debate.

LADs are already complicated, but adding an extra discipline of modelling evolution brings a new dimension of difficulty. The payoff is that once the system is setup, natural selection will take over and allow optimal language learning conditions to emerge.

4.1 Iterated Learning Model (ILM)

The idea behind Kirby's work is to take away the emphasis of biological evolution and he believes too much importance has been placed upon it. The alternative is to treat languages as adaptively evolving systems (Kirby, 2002).

If language is like an organism in its own right, then you begin to see that it has its own set of selected pressures. Humans are its host, and its method of transmission is via human communication. A successful language is one that can be learnt, understood and used, for the benefit of its hosts.

Therefore, Kirby and Hurford (2002) suggests that language is not only subject to biological natural selection, but is the result of three complex adaptive systems, as he illustrated in fig. 3:

> "There clearly are interactions: for example, biological evolution provides the platform on which learning takes place, what can be learnt influences the languages that can persist through cultural evolution, and the structure of the language of a community will

influence the selection pressures on the evolving language users." (ibid)

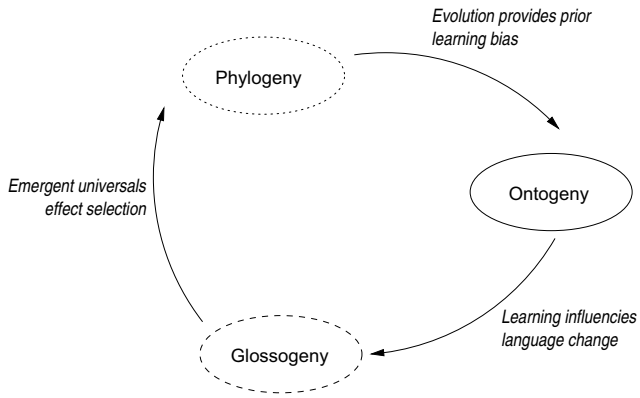

Figure 3: *Interactions of the three adaptive systems.*

ILM is composed of four elements:

1. A meaning space
2. A signal space
3. One or more language-learning agents
4. One or more language-using adult agents

The adult agent as a set of meanings for which is must convert into signals to then be transmitted to the learner (both agents have been initialised with random parameters). The speaker makes an assumption that the hearers signal-to-meaning mapping will be approximately that of the speakers. Therefore, the principle behind transmission is the generate signals that maximise the confidence of the given meaning. Once the speaker has finished, it is then redundant, and the learner is promoted. A new learner is added and the cycle continues.

This is a somewhat simplified version of events, but the overall result is after a few hundred generations — after much randomness — meaningful structure emerges.

Kirby's research on the whole is looking at linguistic evolution rather than creating a tool for grammar inference (although one does lead to the other). And as such, there are no large scale runs of using the ILM to acquire grammar from a large corpus, and to evaluate it in any way. However, its inclusion is still relevant because it is an interesting approach - one that could be adapted to evolve grammar inference devices.

4.2 Language Agents (LAgts)

Language agents (LAgts) are born from the research carried out by Briscoe (2000). An equally intriguing approach to grammar induction through the application of alife, populations of LAgts are simulated. LAgts are language learners, generators and parsers.

Each LAgt adopts a Categorial Grammar as its underlying framework, albeit an extended version known as Generalized Categorial Grammar (GCG), (Wood, 1993). They are used to provide a relationship between the LAgt's universal grammar and the specification of a given grammar; Briscoe is clear which side of the 'innateness' fence he stands on, which is why LAgts possess a universal grammar. They are embedded in a default inheritance network, and are represented as a sequence of *p-settings*. Each setting can be encoded as *True, False* or *?* (which indicates that it is yet to be specified). They are partially ordered too, which means atomic categories can be in an arbitrary position, although more complex types, where directionality is significant, then ordering must be preserved within that type. There is also a distinction between Absolute, Default and Unset parameters.

Simulations are run to model the evolving population of LAgts. A successful interaction is generally one where a random agent generates a sentence based on its current grammar, which can be parsed by another randomly selected agent. Such a pair are said to have compatible *p-settings*. Each LAgt has a set lifespan of 10 *interaction cycles*. Between the ages of 4-10, an agent can reproduce new agents, and from aged 5 onwards, an agent stops learning and its grammar is therefore fixed. Agents older than 10 are removed from the simulation.

During their learning phase, it is possible for LAgts to alter any of their parameters that were given Default of Unset attributes from their conception. There is a cost associated with an update, which is why successful agents tend to changed by one point of their initial *p-settings*. This results in the classical strategy of inheritance where parents pass on their genes and not any acquired characteristics.

Once again, performance related information about the LAgts' actual abilities to acquire grammar accurately is difficult to extract. Much of Briscoe's work has been to experiment with the seemingly unlimited number of factors that affect evolving systems (coevolution, migration, acquisition effectiveness etc). However, languages did emerge from the learners that were described as 'full language': that is, 'close to an attested language'. Ideally, Briscoe would have elaborated as to just *how* 'close' this is. There were seven full languages available for experimentation. A language is given to one or more adult LAgts (depending on the simulation) who then communicate with the learners, and so on.

5 String Pattern Searches

Some systems do not fit conveniently into the above groups, and illustrate the greater breadth of approaches to UGI.

5.1 ABL (Alignment Based Learning)

ABL (van Zaanen, 2002) is a learning paradigm in its own right. It is based on the principle of substitutability, whereby two constituents are of the same type, then they could be substituted. Of course, the system is unsupervised, and therefore, does not know types. Thus, the principle is reversed so that if two constituents can be substituted, they are of the same type. Fig. 4 shows an example from van Zaanen and Adriaans (2001) of two segments from two sentences are declared as being of the same type.

What is a family fare
What is the payload of an African Swallow

What is (a family fare)x
What is (the payload of an African Swallow)x

Figure 4: Example of bootstrapping structure in ABL

Much complexity is added due to alignment learning phase finding constituents that overlap each other. This problem is overcome using a *selection learning* phase. This is where the ABL algorithm determines the correct (or at least the best fit) constituent using probabilistic methods. Selection is decided upon calculating the probability of the words in the overlapping constituent and its type. A Verberti-style algorithm is also used to search through all possible combinations of overlapping constituents and select the best one. ABL performs well. The computational demands of its algorithm mean that it is not suitable for large corpora (>100K sentences). However, its greedy nature means it will learn quickly. An accuracy of 62% was recorded when ABL was given the OVIS corpus to process. Versions of the ABL system have also been tested with the Penn Treebank and ATIS corpora.

5.2 GB (Grammatical Bigrams)

GB or Grammatical Bigrams were proposed by Paskin (Paskin, 2001), in the hope of creating a simple language learning model, and therefore, making the actual learning process tractable. Independence assumptions are introduced to reduce complexity, although at the same time it increases the model's bias.

The Grammatical Bigram model uses the Dependency Grammar formalism to describe the relationship between pairs of words. One word in this link is the head, and the other is its dependent. A dependency parse is a directed graph, consisting of a set of such relationships. No word can be dependent on more than one head (the root of a sentence has no dependency). Also, a word

cannot be dependent on itself, making the links acyclic. If the dependents of a head word are completely independent of each other, and their order, then this independence assumption results in a much simpler model of grammar and so the parser is spared of that complexity.

The parser is used to learn grammar from labelled corpora. However, for unsupervised learning, the EM algorithm is used to learn the optimal parameters (probabilities of dependency for a given head). Other statistics are computed using an adapted version of the Inside-Outside algorithm that works in $O(n^3)$ time.

Unfortunately, Grammatical Bigrams are suited more to the generalisation of labelled data than to unsupervised induction. When given an unlabelled corpus of Wall Street Journal articles, and its output evaluated against the annotated Wall Street Journal section of the Penn Treebank, GB yielded an accuracy of only 39.7%. Clearly, the compromise in making the model computationally efficient results in a grammar model that is still too approximate to represent the sorts of structures it sees in the input corpus.

6 Discussion

This review has attempted to analyse the range of underlying algorithms used by various approaches:

1. GraSp (Henrichsen, 2002): minimising "disorder" in CG lexicon;
2. CLL (Watkinson and Manandhar, 2001) : stochastic CKY parser optimisation from a given lexicon;
3. EMILE (Adriaans, 1999): rule induction from candidates, guided by "oracle";
4. FAMBL (Daelemans et al, 1999): weak abstraction of common subexpressions into "families";
5. RISE (Domingos, 1995): Memory Based Learning or patterns, with rule induction;
6. ILM (Kirby and Hurford, 2002): evolutionary optimisation of a language-space;
7. LAgts (Briscoe, 2000): evolutionary optimisation of language-acquisition software agents;
8. ABL (van Zaanen, 2002): rule induction from substitutable subexpressions;
9. GB (Paskin, 2001): stochastic optimisation of dependency-pair sets.

A review ought to include a comparative evaluation of the alternative approaches and systems; but we could find few objective metrics or evaluative features reported across the source literature. Most researchers appeal to a linguist's "looks good to me" evaluation (Hughes and Atwell, 1994), (Jurafsky and Martin, 2000): they demonstrate that their systems can infer some examples of grammatical constructs which seem similar those found in linguists' grammars. Unfortunately, this is a subjective qualitative assessment, and does not yield a percentage score which can be compared.

Early research on unsupervised word-class inference, clustering words into classes e.g. (Atwell, 1983; Atwell and Drakos, 1987; Finch, 1993) also appealed to "looks good to me" evaluation; but more recent research has tried to measure inferred word-classes against a human-tagged corpus (Hughes and Atwell, 1994; Roberts, 2002). Other areas of Language Engineering also try to evaluate rival systems against a "Gold Standard" human-annotated corpus; so why not Unsupervised Grammar Inference? Only four of the projects surveyed report accuracy measured against a human-parsed corpus:

1. CCL (Watkinson and Manandhar, 2001): 51.9% on Penn Treebank;
2. Grammatical Bigrams (Paskin, 2001): 39.7% on Penn Treebank;
3. EMILE (Adriaans, 1992; Vervoort, 2000): 51.6% on ATIS Corpus;
4. ABL (van Zaanen, 2002): 62% on OVIS corpus (lower with Penn and ATIS).

Even these percentage scores cannot be compared meaningfully as they are based on different alignment-measures, and used different corpora and human parsing schemes. Parsing schemes used in human-annotated treebanks can capture a variety of grammatical information, including some or all of the following (Leech et al, 1996): (a) Bracketing of segments; (b) Labelling of segments; (c) Showing dependency relations; (d) Indicating functional labels; (e) Marking sub-classification of syntactic segments; (f) Deep or 'logical' information; (g) Information about the rank of a syntactic unit; (h) Special syntactic characteristics of spoken language. In their review of corpus parsing schemes, Atwell et al (2000) conclude:

> "unlike the tagging schemes, it does not make sense to make an application-independent comparative evaluation. No single standard can be applied to all parsing projects. Even the presumed lowest common denominator, bracketing, is rejected by some corpus linguists and dependency grammarians. The guiding factor in what is included in a parsing scheme appears to be the author's theoretical persuasion or the application they have in mind."

So, an objective measure of alignment against a human-parsed "gold standard" Treebank may not be feasible or even desirable. In fact, one intriguing potential of Unsupervised Grammar Inference is that it may yield analyses

which fit the data better than traditional grammarians' parse-categories; but if we measure against an established Treebank, any such innovation will be penalised. Unsupervised Grammar Inference has potential applications with unknown languages e.g. (Atwell and Elliott, 2001), for which a high score in learning English grammar may be inappropriate.

An alternative possible metric which suggests itself is "how much has been learnt": some measure of the difference between size or scale of the initial assumption or "learning bias" (Jurafsky and Martin, 2000) and the final grammar which has been inferred. For example, Watkinson and Manandhar (2001) contrasted CCL experiments with initial lexicons of 31 and 348 words, and found that the latter yielded a larger grammar. Unfortunately, other sources did not report comparable "starting assumption" and "final grammar" metrics.

7 Future of Grammar Inference

The next big step within the UGI community is to design and develop a robust evaluation procedure. Many of the systems featured in this paper did not publish performance results of their experiments, favouring a '*looks good to me*' approach. This is certainly not an attempt to say expert linguistic evaluation is somehow inferior to an automatic, computerised approach. However, it is rather subjective, and if carried out by the author of the UGI system, likely to be partial to some bias.

An automatic evaluation tool — if designed correctly — would allow consistent comparison between rival systems. To be able to quantify performance would allow GI designers and developers to ensure that future updates actually provide greater accuracy, and quickly, so that research is not led down a dead end if the results of a system looked good, but performance was in fact degrading. However, it is clearly fraught with difficulties, which is the likely reason why many have steered clear.

With respect to UGI itself, we can look forward to great advances in the long term. The computational complexity of the algorithms will become less of a burden with optimisation and increased computational resources. We also look forward to wider applications on different datasets, as Unsupervised Grammar Inference is more widely recognised as a powerful data-mining technique.

References

Adriaans, P.W. *Language Learning from a Categorial Perspective*. Ph.D. thesis, Unversiteit van Amsterdam. 1992.

Adriaans, P.W. *Learning shallow context-free languages under simple distributions*. ILLC Report PP-1999-13, Institute for Logic, Language and Computation, Amsterdam.

Ajdukiewicz, K. *Die syntaktische Konnexität.* Studia Philosophica. 1. 1935. pp. 1-27.

Atwell, E. *Constituent likelihood Grammar.* ICAME Journal, 7, 983, pp. 34-65.

Atwell, E and Drakos, N. *Pattern recognition applied to the acquisition of a grammatical classification system from unrestricted English text.* In Maegaard, B. (Ed.), Proceedings of the Third Conference of European Chapter of the Association for Computational Linguistics, 1987, pp. 56-63.

Atwell, E, Demetriou, G, Hughes, J, Schiffrin, A, Souter, C and Wilcock, S. *A comparative evaluation of modern English corpus grammatical annotation schemes.* ICAME Journal 24, 2000, pp. 7-23.

Atwell, E. and Elliott, J. *A corpus for interstellar communication.* In Rayson, P, Wilson, A, McEnery, T, Hardie, A, and Khoja, S. (Eds.), Proceedings of CL2001: International Conference on Corpus Linguistics. UCREL Technical Paper 13, Lancaster University, 2001, pp. 31-39.

Bates, E. and Goodman, J.C. *On the Inseparability of Grammar and the Lexicon: Evidence from Acquisition, Aphasia, and Real-time Processing.* Language and Cognitive Processes, 12, 1997, pp. 507-584.

Belkin, M. and Goldsmith, J. *Using eigenvectors of the bigram graph to infer morpheme identity.* Proceedings of the Morphology/Phonology Learning Workshop of ACL-02. Association for Computational Linguistics. 2002.

Briscoe, E.J. *Grammatical Acquisition: Inductive Bias and Coevolution of Language and the Language Acquisition Device.* Language, 76(2). 2000. pp. 245-296.

Daelemans, W, Van den Bosch, A and Zavrel, J. *Forgetting exceptions is harmful in language learning.* Machine Learning, 11. 1999. pp. 11-43.

Domingos, P. *The RISE 2.0 system: A case study in multistrategy learning.* Technical Report 95-2, Department of Information and Computer Science, University of California. 1995.

Dong, S and Searls, D.B. *Gene Structure Prediction by Linguistic Methods.* Genomics. 1994.

Elliott, J, Atwell, E and Whyte, W. *Visualisation of Long Distance Grammatical Collocation Patterns in Language.* In IV2001: 5th International Conference on Information Visualisation, London, UK. 2001.

Finch, S. *Finding structure in language.* PhD thesis, Edinburgh University. 1993.

Freltag, D. *Using Grammatical Inference to Improve Precision in Information Extraction.* In Working Papers of the ICML-97 Workshop on Automata Induction, Grammatical Inference and Language Acquisition. 1997.

Gold, E.M. *Language Identification in the Limit*. Information and Control. 10. 1967. pp. 447-474.

Henrichsen, P.J. *GraSp: Grammar Learning from unlabelled speech corpora*. In: Roth, D and Van den Bosch, A. (Eds), Proceedings of CoNLL-2002, Taipei, Taiwan, 2002, pp. 22-28.

Hong, T.W and Clark, K.L. *Using Grammatical Inference to Automate Extraction from the Web*. In Principles of Data Mining and Knowledge Discovery. 2001. pp. 216-227.

Hughes, J and Atwell, E. *The automated evaluation of inferred word classifications*. In Cohn, A. (Ed), Proceedings of ECAI'94: 11th European Conference on Artificial Intelligence. John Wiley, 1994, pp535-539.

Kirby, S and Hurford, J. The emergence of linguistic structure: an overview of the iterated learning model. In: Cangelosi, A and Parisi, D (Eds.) *Simulating the Evolution of Language*. 2002. pp. 121-148.

Kirby, S. *Natural Language from Artificial Life*. Artificial Life, 8(2). 2002. pp. 185-215.

Jurafsky, D and Martin, J. *Speech and Language Processing*. Prentice-Hall, 2000.

Leech, G, Barnett, R and Kahrel, P. *EAGLES Final Report and guidelines for the syntactic annotation of corpora*. European Expert Advisory Group on Language Engineering Standards (EAGLES) Report EAG-TCWG-SASG/1.5, 1996.

Paskin, M.A. *Grammatical Bigrams*. In Dietterich, T, Becker, S, and Gharahmani, Z (eds.), Advances in Neural Information Processing Systems 14. Cambridge, MA: MIT Press. 2001.

Roberts, A. *Automatic acquisition of word classification using distributional analysis of content words with respect to function words* Technical Report, School of Computing, University of Leeds, 2002.

Steedman, M. Constituency and Coordination in a Combinatory Grammar. In: Baltin, M.R and Kroch, A.S (Eds.), *Alternative Conceptions of Phrase Structure*. University of Chicago. 1989. pp. 201-231.

Van den Bosch, A. *Careful abstraction from instance families in memory-based language learning*. Journal of Experimental and Theoretical Artificial Intelligence, 11:3, special issue on Memory-Based Language Processing, Daelemans, W, guest ed. 1999. pp. 339-368.

Van Zaanen, M and Adriaans, P.W. *Comparing Two Unsupervised Grammar Induction Systems: Alignment-Based Learning vs. EMILE*. Technical Report: TR2001.05, School of Computing, University of Leeds. 2001.

Van Zaanen, M. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD Thesis, School of Computing, University of Leeds. 2002.

Vervoort, M.R. *Games, Ealks and Grammars*. Ph.D thesis, Universiteit van Amsterdam. 2000.

Watkinson, S and Manandhar, S.*A Psychologically Plausible and Computationally Effective Approach to Learning Syntax*, CoNLL'01, the Workshop on Computational Natural Language Learning, ACL/EACL 2001.

Watkinson, S and Manandhar, S.*Translating treebank annotation for evaluation*. In: Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL/EACL 2001.

Wood, M. *Categorial Grammars*. Routledge. London. 1993.

Conference report

CL2003: the International Conference on Corpus Linguistics

Andrew Roberts, *University of Leeds*

After the initial success of the first Corpus Linguistics conference in 2001 in honour of Geoffrey Leech's 65th birthday, Tony McEnery et al. decided to make it a more permanent fixture in the linguistics calendar, and stage the conference every two years. Hence, it was time for the sequel to commence, and, as a result, hundreds of corpus linguists were seen flocking to Lancaster. The peaceful university campus, located out of town, with its countryside atmosphere, was a lovely setting for the event. Luckily, the weather also blessed us with fine sunshine throughout – much to the relief of the many international participants who were expecting the infamous British climate (i.e., wet and cold!).

Around this time two years ago, Geoffrey Sampson wrote an article in *ELSNews* regarding the future role of ICAME (in particular, their conferences). He remarked that ICAME's focus on English research, coupled with its restrictions on conference sizes to ensure a *friendly* atmosphere, means that not only does it miss out on the ever increasing (and exciting) shift towards non-English language research, but also the next generation of researchers, with energy and fresh ideas, but who are not yet established enough to join the ICAME clique. He went on to comment about the success of CL2001 that had taken place shortly before, and predicted success for its future. ICAME's weaknesses are CL's strengths, which is why Sampson's prediction was correct.

The conference kicked off with a day of workshops covering development of learner corpora and multilingual corpora, corpus-based approaches to figurative language, and shallow processing of large corpora (SProLaC). The following four days saw papers presented in three parallel sessions. With approximately 95 papers and 30 posters on offer, it would be impractical to go into any real depth about them. Needless to say, all areas of the field were well represented. There were reports of new resources and research that is being developed for European minority languages and South-East Asian languages. The usual suspects, such as tagging, parsing, disambiguation, grammars, and information extraction were well covered. Unsurprisingly, there was also a strong focus on corpus development, annotation, and tools. Even if translation studies, exploiting corpora, and semantics are added to the list, it is still

Corpus Linguistics researchers from Leeds University, alongside Geoffrey Leech (far right), original raison d'être for the Corpus Linguistics conference. Andrew Roberts is on the far left.

not exhaustive – which simply illustrates the breadth of the conference. Of course, the fact that each piece of research presented was, by definition, linked to corpora, means that there was a common thread throughout, and so, as varied as the topics were, they never felt disjointed.

Invited speakers were Michael Hoey, Nancy Ide, Susan Hunston, Geoffrey Sampson, and Nicoletta Calzolari. All are well known within the field and offered fascinating and well received talks. Probably the most infamous presentation of the conference was by Tony McEnery, covering his studies of 'the f-word' within the BNC. It was even rumoured that Eric Atwell, a speaker in a parallel session, was recommending to his audience to go and hear Tony's instead as it was more interesting!

The hospitality was excellent throughout the conference, especially the catering, which was of a particularly high standard. Naturally, coffee breaks and meal times were the main social occasions. None more so than on the third day, when we were all whisked away by coach to the magnificent Ashton Memorial within the beautiful Williamson Park, which sits high above the main town centre. From here, people could enjoy the lovely scenery, most notably the mountains of the Lake District National Park. The memorial building itself wasn't particularly large, and with so many people in it, personal space was quickly becoming a luxury. If there was anybody with whom you weren't acquainted, you were by the end of the night, which was a good thing in my opinion!

The conference was very well organised, and was probably as close to optimal as you could get. Whilst there were approximately 200 participants (representing 30 countries), the atmosphere was still very friendly and informal. Also, three does appear to be the magic number in terms of the number of parallel sessions. At times, choosing one of three talks could be a difficult decision, but at least it gives a degree of flexibility. Any more and I think that people may begin getting frustrated at missing too many talks, especially when there is more than one presentation of interest at a given time – as is often the case at some of the larger conferences. Therefore, congratulations should go to Tony McEnery, Dawn Archer, Paul Rayson, Andrew Wilson, plus the many other local staff who helped to ensure an enjoyable and interesting conference. We look forward to CL2005!

FOR INFORMATION

Andy Roberts is a research student at the University of Leeds.

Email: andyr@comp.leeds.ac.uk

Web: www.comp.leeds.ac.uk/andyr

Proceedings of this and previous CL conferences and the SProLaC workshop are available as UCREL technical papers as follows:

2003: Dawn Archer, Paul Rayson, Andrew Wilson and Tony McEnery (eds.). Proceedings of the Corpus Linguistics 2003 conference. UCREL technical paper number 16. UCREL, Lancaster University

2001: Paul Rayson, Andrew Wilson, Tony McEnery, Andrew Hardie and Shereen Khoja (eds). Proceedings of the Corpus Linguistics 2001 conference. UCREL technical paper number 13. UCREL, Lancaster University

SProLaC: Kiril Simov and Petya Osenova (eds) (2003). Proceedings of the The Workshop on Shallow Processing of Large Corpora (SProLaC 2003) held in conjunction with the Corpus Linguistics 2003 conference. UCREL technical paper number 17. UCREL, Lancaster University.

contd from p.11

the assistance of Paola Baroni and Monica Monachini.

One of the objectives of this Workshop is to launch the ICCWLRE (International Co-ordination Committee for Written Language Resources and Evaluation).

The Workshop, originally planned as an ACL 2003 Workshop, will be held in Paris on 28th and 29th August 2003.

Further information about both the ENABLER Network and the ENABLER/ELSNET Workshop can be found at the website.

FOR INFORMATION

ENABLER

Web: www.enabler-network.org

Workshop

Email:
Nicoletta Calzolari: glottolo@ilc.cnr.it
Alessandro Lenci: alessandro.lenci@ilc.cnr.it
Steven Krauwer: steven.krauwer@elsnet.org
Paola Baroni: eagles@ilc.cnr.it
Monica Monachini: monica.monachini@ilc.cnr.it

The use of corpora for automatic evaluation of grammar inference systems

Andrew Roberts
andyr@comp.leeds.ac.uk

Eric Atwell
eric@comp.leeds.ac.uk

School of Computing
University of Leeds
Leeds
LS2 9JT
United Kingdom

Abstract

The evaluation of grammar inference systems is clearly a non-trivial task, as it is possible to have more than one correct grammar for a given language. The 'looks good to me' approach, carried out by computational linguists analysing their own grammar inference system results, has prevailed for many years. This paper explores why this method has been so popular, in terms of its strengths, and also why it is no longer adequate as a reliable means to measuring performance. Corpus based methods, that can be performed automatically, are investigated to see how they can meet the needs of this difficult problem.

1 Introduction

In the past few years, the Natural Language Learning community have produced systems targeted at the complex task of Grammar Inference (GI): automatically inferring or learning grammatical descriptions of a language from a corpus of language examples. A low-level GI task is to group or cluster words into tentative categories according to their distributional behaviour, e.g., Atwell and Drakos (1987), Hughes and Atwell (1994) and Roberts (2002). A more ambitious aim of GI is to propose grammatical phrase structure; a number of mature solutions have emerged, for example, GraSp (Henrichsen 2002), CLL (Watkinson and Manandhar 2001a), ABL (van Zaanen 2001) and EMILE (Adriaans 1992). All systems mentioned aim to be unsupervised, and thus can only rely on raw (unlabelled) text for their learning process. Results from these systems are promising – a variety of linguistic phenomena are induced.

Despite the advances in this field, the issue of thorough evaluation has been poorly address. Evaluation within Natural Language Processing tasks in general is problematic, not least because there is no obvious single correct output to measure against. For example, for PoS-tagging and parsing, different linguists advocate different tagsets and parsing schemes, making it difficult to compare accuracy metrics (Atwell 1996; Atwell *et al*. 2000). Ambiguity poses a similar threat in GI: the basic problem is given a training corpus, there is no single correct grammar that represents it. In the majority of systems, a 'looks good to me' approach has been used: success is illustrated by presenting some grammar classifications or structures proposed by the system which appeal to the linguist's intuition.

There is a need to develop methods for consistent evaluation of GI systems. Without a reliable way of determining the performance of such systems, it will become increasingly difficult to assess how competently they are doing the task they are supposed to do. Nor will it be trivial to compare two or more systems, which would be very valuable in deciding which techniques and algorithms work best for Natural Language Learning.

This paper investigates the feasibility of harnessing corpora that can be used to develop a standard mechanism (which aims to be reliable and accurate) for evaluating GI systems in the future. The next section explores the 'looks good to me' approach mentioned earlier to understand why it is so widely used, its merits and any shortcomings. Section three introduces various approaches for evaluating GI. A discussion takes place in section four to highlight important issues from the approaches reviewed in this paper. Section five provides brings the paper to a close with a conclusion.

2 Looks good to me

The success of this approach is essentially its apparent simplicity. A system performs its GI procedures on a piece of unstructured text, and its resulting grammar can be analysed by a computational linguist, generally the computational linguist who built the Grammar Inference system under scrutiny. A qualitative evaluation takes place based on the linguistic intuitions of the evaluator, by highlighting features or structures in the learner output which look "good", or reminiscent of structures in a recognised linguistic theory. Of course, the apparent simplicity is due to the fact that a linguist possesses the required skills and experience to easily deduce whether the grammar in question contains a plausible structure. Researchers who came into the field of GI from a computing/AI/machine learning background are less likely to be as proficient at evaluating grammars using this approach.

'Looks good to me' is an important method of evaluation, and certainly should not be discredited just due to its lack of automation. Verification of a system carried out independently by one or more linguists would provide – in most cases – a more reliable measure of performance. It is also arguably the most resource efficient, since it can be evaluated on different languages without the need of structured corpora (van Zaanen 2001). Examples of this method in use can be found in. (Finch and Chater 1992; Losee 1996; Vervoort 2000; Henrichsen 2002).

However, the method has many disadvantages. Evaluation of this nature is mainly conducted by the developer of the system rather than someone independent to the work. Thus, there is a high chance of bias whereby the systems successes are highlighted and shortcomings are glossed over, making it almost impossible to gain an accurate picture of system performance. Even without any bias, the process is time consuming and would rely on the subjectivity of the expert(s), and may also be prone to unknown external factors that can affect humans. And finally, it does not offer the facility for comparing different systems which would be of great benefit.

3 Automatic evaluation

Ideally, the process of evaluation should be performed automatically which will save on time and on the necessity of an expert linguist in the chosen language. This section focuses on methods that harness corpora for this purpose.

3.1 'Gold standard' treebank

This method is currently the most common to be adopted. It works by extracting the original natural language sentences from an existing treebank (the 'gold standard'), and using it as input to a given GI system. The structured sentences produced by the GI system are then compared to the structure found in the original treebank. Common metrics include recall (measures the completeness of the learned grammar) and precision (measures the correctness of the learned grammar) that can be calculated to give an objective measure of system performance. This method has been used in (Brill 1993; Déjean 2000; van Zaanen 2001)

Although on the surface, this method appears sound, and simple to implement, there are in fact many issues that cause it to be insufficient for it to be reliable and accurate. Treebanks are expensive (both in time and money) to create, and so they are not in plentiful supply. The material within also tends to be exclusive to a particular domain, e.g., the ATIS treebank consists of sentences on questions and imperatives on air traffic, or the Penn Treebank is taken mostly from articles in the Wall Street Journal. If a GI system is not designed to learn from raw text of the same genre or subject,, then it is not an ideal candidate for a 'gold standard'. The way in which the treebank has been structured poses a problem. Just because it has been labelled the 'gold standard' does not necessarily mean that anything that differs is wrong – it may simply be different, but equally correct. However, such differences will clearly affect the measure of the systems' performance. It does also beg the question about whether certain treebanks are credible enough to be used for the purposes of evaluation. One must be confident that the quality and validity of an annotated is sufficiently high, otherwise, it is pointless evaluating systems with corpora that contain errors or are not well structured.

On a slightly more technical level, a large step to overcome is the likely discrepancy between the actual annotation schemes of the GI system to be evaluated and the 'gold standard'. For example, an increasingly common grammar being employed within GI is the *categorial grammar* (Ajdukiewicz 1935). However, there is no corresponding treebank whose annotation formalism uses CG, which means the only useful measurements that can be acquired from this mismatch are metrics such as *crossing-brackets*, but so much information is wasted. Therefore, a system of translation is required to either convert the original treebank to the annotation scheme used by the GI system, or the other way around, in order for a more precise comparison to commence.. This was the tactic taken by Watkinson and Manandhar (2001b) in their evaluation of CLL, whereby they set about establishing a 'gold standard' by translating the Penn Treebank annotation scheme into one using categorial grammar markup .

3.2 Multi-annotated corpora

To overcome some of the main disadvantages of the 'gold standard' approach, the next logical step is to develop a multi-annotated corpus, as exemplified in Atwell *et al*. (2000). The premise is that the same corpus is parsed by a variety of systems, rather than merely one as all common treebanks are. Providing all parses are trusted to be correct, then upon evaluation of a GI system, it is less likely to be penalised just because it produces different but equally correct parses, because it is being compared to a variety of such parses.

This step should make the evaluation fairer, however, it does add a new set of complexities. Not least due to the aforementioned annotation translation problem, which multiplies for each unique scheme used by the various parsers. Ideally, the same annotation would be used for all parses within the treebank – the 'gold standard' annotation scheme. But this too is fraught with difficulties, especially as it would be much simpler to accomplish a multi-annotated corpus by using off-the-shelf parsers.

Another issue is how to compare a given parsed sentence to be evaluated with a set of known correct parses. The best case is that a perfect match will be found. The worse case is no match. However, for many cases, it will be somewhere in between, where a number of partial matches are likely to exist. A best match approach seems logical, i.e., of the candidate parses, the one that is the most similar is the one that is then considered for the actual metric calculations.

3.3 Multi-corpora

In a similar vein to the above approach, the idea for using more than one corpus is primarily to avoid the domain specific issue that was addressed earlier (see section 3.1). Unlike other corpora, such as the Brown or LOB, which span many subjects in different contexts (newspaper articles, novels etc), well known treebanks fail to compare in size and variety. Therefore, this method aims to take advantage of smaller individual treebanks and combine into a single, large one. This would certainly make it fairer for general purpose learning systems, although perhaps there is less of a need for more focused systems.

Having said that, an alternative tactic is to treat each of the contributing corpora as individual and each as with the normal 'gold standard' method. You would then have a more detailed type of benchmark style scoring system, where not only is there an overall performance measure against all the corpora on test, it can also be seen on which type of corpora the GI system works best (or worse) on.

Of course, the most thorough approach would be if the smaller treebanks were also multi-annotated as described in the previous section. A 'multi-multi-annotated corpus' – if you like – would be a magnificent resource. This subtly shifts the aim of evaluation: the result or output is not a single overall "accuracy score", but an exploration of a range of parsing schemes and genres to highlight similarities and differences between GI output and a variety of accepted linguistic analysis schemes. Ideally such an analysis could be facilitated by a parsed-corpus exploration toolkit: a tool to compare parses in GI output and linguist-annotated multi-multi-corpus, and automatically highlight differences.

4 Discussion

There does not seem to be a perfect solution to the problem of reliable evaluation, whether it be manual or automatic. GI and parsing is very subjective, and it is unlikely that a method will be created that will satisfy everyone. However, what should be unanimous is that a system of evaluation needs to exist. It may not be perfect, but at least a fair and consistent scheme can be created.

Unfortunately, it is clear that the most thorough automated evaluation approaches could be such a burden to developers to implement that they will opt for a simpler approach. Which is why it may be worthwhile to outsource the task of creating the ultimate 'gold standard' into a research project within its own right. The creation of a collection of multi-annotated corpora as the central resource, as well as a variety of translation interfaces for commonly used annotation schemes, that allow easy comparison between output of a GI system and the 'gold standard' corpus. It could be packaged like an *evaluation toolkit* – a black box that is publicly available, easily accessible, and simple to add on as the final phase within the pipeline of the GI system.

Alternatively, a multi-multi-annotated corpus could be achieved by getting the GI community – or at least a group of GI researchers – to work together, each contributing their preferered "target" analysis schemes. Sutcliffe et al (1996) describes an analogous evaluation exercise for a range of (non-ML) parsers: each researcher was asked to "bring along" their parser analyses of an agreed set of test sentences to a joint workshop, and then highlight and discuss successes (and failings) of their systems. Perhaps in a future GI workshop, GI researchers could be invited to bring along their preferred evaluation treebanks to challenge each other, and to each present both "looks good to me" and quantitative evaluations of systems for comparison.

5 Conclusion

The 'looks good to me' approach, despite the critical slant within this paper, is not an inferior one. However, it will not meet the demands for robust NLL evaluation. Which is why corpus based approaches have been presented as the most feasible and reliable way of essentially trying to emulate 'looks good to me' whilst eliminating bias and providing consistency.

There is a great need for researchers within the field to begin addressing accurate evaluation techniques. The current 'gold standard' method (as described in section 3.1) is too basic and will simply favour GI systems that behave similar to the way the 'gold standard' treebank was parsed. Greater flexibility is required, and a method like the multi-multi-annotated corpus (see section 3.3) will be a beneficial innovation to do just that. Instead of replacing "looks good to me", the two approaches should be combined, so that evaluation is no longer a simple quest for a single "score", but instead an exploration of strengths (and weaknesses) of GI systems.

References

Adriaans P W 1992 *Language learning from a categorial perspective*. PhD thesis, Unversiteit van Amsterdam.

Ajdukiewicz K 1935 Die syntaktische Konnexiät. *Studia Philosophica* 1:1-27.

Atwell E 1996 Comparative evaluation of grammatical annotation models. In Sutcliffe R, Koch H, McElligott (eds), *Industrial parsing of software manuals*. Amsterdam: Rodopi, pp 25-46.

Atwell E, Demetriou G, Hughes J, Schiffrin A, Souter C, Wilcock S 2000 A comparative evaluation of modern English corpus grammatical schemes. *ICAME Journal*, 24:7-23.

Atwell E, Drakos N 1987 Pattern Recognition applied to the acquisition of a grammatical classification system from unrestricted English text. In *Proceedings of EACL: Third conference of the European chapter of the association for computational linguistics*, New Jersey.

Brill E 1993 Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of the 31st annual meeting of the Association for Computational Linguistics (ACL)*, Columbus, Ohio, USA, pp 259-265.

Déjean H 2000 ALLiS: a symbolic learning system for natural language learning. In Cardie C, Daelemans W, Nédellec C, Tjong Kim Sang E (eds), *Proceedings of the fourth conference on computational natural language learning and of the second learning language in logic workshop*. Lisbon, Portugal, pp 95-98.

Finch S, Chater N 1992 Bootstrapping syntactic categories using statistical methods. In Daelemans W, Powers D (eds), *Backgrounds and experiments in machine learning and natural language: Proceedings first SHOE workshop.* Institute for Language Technology and AI, Tilburg University, pp 230-235.

Henrichsen P J 2002 GraSp: Grammar learning from unlabelled speech corpora. In Roth D, van den Bosch A (eds), *Proceedings of CoNLL-2002*. Taipei, Taiwan, pp 22-28.

Hughes J, Atwell E 1994 The automated evaluation of inferred word classifications. In Cohn A (ed), *Proceedings of ECAI '94: 11th European conference on artificial intelligence*, John Wiley, Chichester, pp 535-540.

Losee R M 1996 Learning syntactic rules and tags with genetic algorithm for information retrieval and filtering: An empirical basis for grammatical rules. *Information processing & management, 32(2):185-197.*

Roberts A 2002 *Automatic acquisition of word classification using distributional analysis of content words with respect to function words*. Technical report, School of Computing, University of Leeds.

Sutcliffe R, Koch H, McElligott (eds). 1996 *Industrial parsing of software manuals*. Amsterdam: Rodopi

van Zaanen M 2001 *Bootstrapping structure into language: alignment-based learning*. PhD thesis, School of Computing, University of Leeds.

Vervoort M R 2000 *Games, walks and grammars*. PhD thesis, Unversiteit van Amsterdam.

Watkinson S, Manandhar S 2001a A psychologically plausible and computationally effective approach to learning syntax. In *CoNLL '01: the workshop on computational natural language learning*, ACL/EACL.

Watkinson S, Manandhar S 2001b Translating treebank annotation for evaluation. In *Proceedings of the workshop on evaluation methodologies for language and dialogue systems*, ACL/EACL

aConCorde: Towards an open-source, extendable concordancer for Arabic

Andrew Roberts, Latifa Al-Sulaiti
and Eric Atwell[1]

Abstract

There is, currently, a surge of activity surrounding Arabic corpus linguistics. As the number of available Arabic corpora continues to grow, there is an increasing need for robust tools that can process this data, whether for research or teaching. One such tool that is useful for both of these purposes is the concordancer – a simple tool for displaying a specified target word in its context. However, obtaining one that can reliably cope with the Arabic language had proved difficult. Also, there was a desire to add some novel features to the standard concordancer to enhance its usefulness within the classroom – easy-to-use root- and stem-based concordance and integration to corpus clustering algorithms are two examples. Therefore, *aConCorde* was created to provide such a tool to the community.

1. Introduction

> 'If a corpus is to be useful, we obviously need to be able to search it quickly and automatically to find examples of a particular linguistic phenomenon (say, a word), to sort the set of examples as required, and to present the resulting list to the user. The kind of program which performs these tasks is a concordancer, and the output it produces is known as a concordance. A concordance is a list of all the examples of the target item (the linguistic phenomenon being searched for), normally accompanied by enough context to enable a human being to study the item's occurrence in detail.'
>
> (Leech and Fligelstone, 1992)

In effect, a concordancer is a tool for summarising the contents of corpora based on words of interest to the user. Lexicographers can use the evidence of a concordance to establish whether a word has multiple senses, and also to help define its meaning (Sinclair, 1987; Zernik, 1991). Concordance tools can be beneficial for data-driven language learning (Johns, 1990).

A number of studies have demonstrated that providing access to corpora and concordancers benefits students learning a second language. Just as lexicographers and linguists can find insights into grammatical structure by studying concordance output, so can language learners (Dodd, 1997). Cobb *et al*. (2001) detailed the improved rates of vocabulary acquisition when using a software tool of which a concordancer was a major component.

[1] *Correspondence to*: Andrew Roberts, *e-mail*: andyr@comp.leeds.ac.uk
address: School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom

The simplicity and usefulness of concordancers is such that they should be a valuable tool for any linguist. Unfortunately, as is the general trend within corpus linguistics, research priorities have focused on European languages (although there has been a recent shift towards developing tools in non-European languages such as Chinese, Japanese and Korean). This has meant that there are fewer tools for other languages such as Arabic. When this project began, to the best of our knowledge, there were no readily available concordancers that function correctly with the Arabic language. Arabic is an international language rivalling English in the number of mother-tongue speakers (al-Sulaiti and Atwell, 2006). Graddol (1997) predicts that the number of Arabic mother-tongue speakers will rise from 200 million in 1995 to about 480 million by 2050, whereas English mother-tongue speakers will rise from 360 million to about 500 million within the same period. Considering the growth of interest in Arabic, we felt that Arabic linguists deserved a useful, usable concordancer tool. To this end, the *aConCorde* project was established.

2. Why is Arabic Concordancing Hard?

It is fair to say that for non-Arabic developers, writing tools to process Arabic is more difficult. This perhaps explains why many tools developed for Western languages do not extend well to Arabic. The two most commonly perceived difficulties with Arabic language processing are:

1. The cursive nature of Arabic script means that letters are represented differently depending on whether they occur at the beginning, middle or end of a word. Characters within words are always joined, and never written individually. Vowels are "second-class" characters, in that they can be omitted, and are left for the human reader to infer from context.
2. Arabic is written from right-to-left, as opposed to the majority of other languages that are, of course, written left-to-right.

The process of transliteration is a common approach to resolving these issues. The Buckwalter system (amongst others) will convert Arabic script to an equivalent based on the Roman alphabet (Buckwalter, 2002). It means that the text orientation is also converted, too: the transliterated text reads left-to-right. However, this alone does not fill in missing vowels from the source text. From a computational perspective, transliteration has historically been a necessary step: computers used to be restricted to ASCII (or similar) character sets (i.e., Roman alphabets). However, thanks to the popularity of the Unicode standards since the early 1990s, most modern operating systems and products now support multi-lingual text encodings, and the fonts to display Unicode characters are increasingly common. Implementation of bi-directional text components are also part of most modern operating systems and platform-independent programming languages like Java.

Due to these developments, there is little reason why existing concordance tools have not been adapted to take advantage of the above technologies in order to render the *aConCorde* project redundant even before it had begun. One implementation that sought to provide multi-lingual concordancing, was *xconcord* (Ogden and Bernick, 1996; Boualem *et al*., 1999), developed at the Computing Research Laboratory (CRL), New Mexico University. The CRL produced their own graphical components that were able to display Unicode text and they worked

successfully at displaying Arabic text. Even though *xconcord* was ahead of its time when it was first developed in 1996, it did not become a mainstream application. It would still be a useful tool today, but it is hindered because it was written for the Sun Solaris platform. This is not widely used – especially by the average linguist – since this type of system is typically used on expensive high-powered workstations and servers. Unfortunately, therefore, it was not possible to experiment with it as resources required were not available. Development ceased many years ago, but the program can be downloaded[2].

The real difficulty in Arabic concordancing lies not in the displaying of results, but in the inadequacy of searching for Arabic word stems. Commonly, when performing concordance searches, the user wants to see the usage of words all from the same stem. Concordancers tend to offer *wildcard* searches, for example, *perform** (where the asterisk means any character zero or more times) would match *perform*, *performs*, *performer*, *performers*, *performing*, etc. However, Arabic morphology is substantially more complex than English. Arabic words are derived from a root of typically three consonants, and each root has a set of patterns which can add additional characters as an affix, a prefix or even an infix. An example would be the transliterated Arabic root *ktb* (write). Patterns associated with that root can produce words semantically linked, including verbs like *kataba* (he wrote) and *naktubu* (we write), and many nouns like *kitAb* (book), *maktuub* (letter) and *maktaba* (library)[3]. However, performing a wildcard search like **k*t*b** would return too many matches, many of which are not associated with the *ktb* root, simply because they happen to contain these consonants.

With the exception of the Qur'an, the lack of vowels in modern written text provides difficulties too. This leaves a great deal of ambiguity that is only resolved when looking at the context of a given word. An English example may be *fr* that could be *for*, *fir*, *fur*, *far*, *four*, *fear*, *fair*, *fire*, *afar*, *afore*, etc. Yet, if you can only search for *fr* even though you are only interested in *far*, it would clearly be frustrating to have to read through irrelevant results. It is unlikely that these specific issues will be resolved by generic concordancers.

3. Arabic Concordance with *MonoConc*, *Wordsmith*, *Xaira* and *aConCorde*

For the task of concordance, the market leaders are *MonoConc* (Lawler, 2000) and *WordSmith* (Scott, 2004). Both are commercial products and are well-established tools for text analysis. They are only available on the Microsoft Windows platform. *Xaira* (Burnard, 2004) is a relative newcomer but stems from the well-known *SARA* toolkit for users of the British National Corpus. This section primarily seeks to compare the ability of retrieving concordance output from Arabic corpora using these established tools.

The corpus used in the comparisons is the Corpus of Contemporary Arabic, or CCA (al-Sulati, 2004), consisting of 850,000 words of various text-types in over 400 files. This corpus is annotated according to the TEI standards using XML markup, and all files are encoded in the 8-bit Unicode standard, UTF-8. A key aim of the CCA was that it should be for use within a pedagogical context to supplement the teaching of Arabic as a foreign language using corpus-based approaches.

---

[2] http://crl.nmsu.edu/software/

[3] Many sources cover Arabic grammar, however, issues about Arabic and computing are well introduced by Khoja (2003) and de Roeck (2002).

3.1 *MonoConc*

Developed by a linguist, Michael Barlow, for language teachers and researchers, *MonoConc* has proved to be very popular for language analysis because it comes with many additional features, such as collocation discovery and tag-sensitive searches. The *MonoConc* website[4] states, 'Some users have been using *MonoConc/Pro* with Arabic. I don't have any details of their procedures.' Unfortunately, despite initial optimism, using Arabic texts with *MonoConc* was not totally successful. Issues that arose during experiments were as follows:

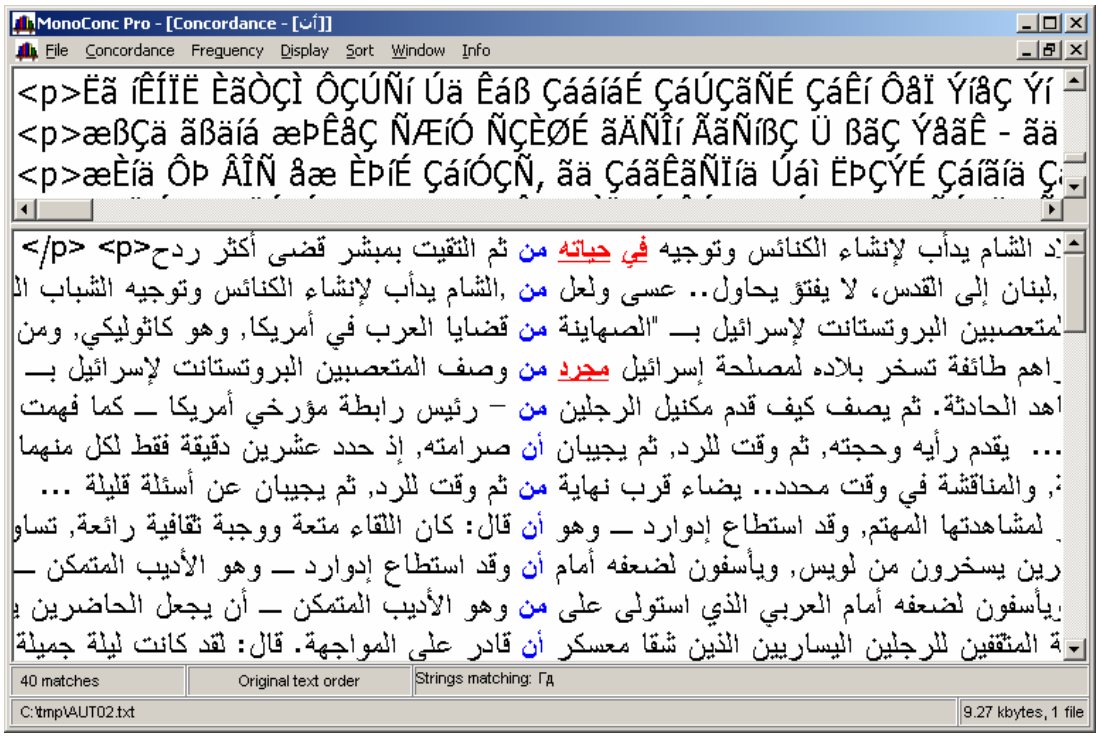

Figure 1: Arabic concordance using *MonoConc*[5]

Unicode support. It appeared that *MonoConc* does not support texts encoded in UTF-8. It was therefore necessary to convert texts to the Windows Arabic Codepage-1256 before the program would display the Arabic text correctly.

Incorrect concordance order. Perhaps the most significant problem for performing Arabic concordance is that the output is in the wrong order (See Figure 1). *MonoConc* does not take the right-to-left nature of Arabic into account when displaying results. For an English reader, it would be equivalent to seeing the output: "on the mat. [sat] The cat", rather than "The cat [sat] on the mat." (where the bracketed word is the target word). This was observed by Hoogland (2003) during the Nijmegen Dutch-Arabic Dictionary Project, 'This seemed a serious shortcoming in the beginning, but soon we experienced that we got used to this very quickly.' Hoogland *had* to get used to it because it was better to have a broken concordance

[4] http://www.monoconc.com
[5] Discovered collocates are underlined, although they should not be there. For unknown reasons the text in the top pane did not display using Arabic fonts.

than none at all. It is clearly far from ideal – it would be a problem for teaching purposes in particular as it would confuse the learner. (N.B. When the concordance results are saved to a file, and this file is viewed with a text editor, the ordering issue disappears.)

'Noisy collocates'. A strange artefact appeared in the concordance output whenever a match included a discovered collocate. It was default behaviour for the software to highlight found collocates. However, the only problem was that these collocates were actually inserted into the wrong position within the concordance context, and as a result caused a certain amount of interference. This was remedied by turning off the option to highlight collocates, after which the highlighted words would then vanish from the context.

Addition of unwanted terms. Despite a simple search term being submitted to *MonoConc*, sometimes it included matches within the results that are not equal to the target word. This behaviour appears to be random as it only occurs with some words and not others. This is illustrated, too, in Figure 1 where the central column contains tokens that were different from the search term.

3.2 *WordSmith*

WordSmith was first released in 1996 and is still developed by a linguist, Mike Scott. *WordSmith* is currently at version 4 and sports three tools: Wordlist, Keyword and Concord. The final tool is of interest in this report, but it should be obvious what the others do. The documentation notes that *WordSmith* supports Unicode, which obviously lends itself to the analysis of the CCA. A demo version is available, which was used for this report. It retains all the functionality of the full version, but significantly limits the number of results returned when performing queries.

A degree of success was initially achieved when using the tool during September 2005, in that it was possible to get the tool to display Arabic script – which means its Unicode support was working. Unfortunately, like *MonoConc* it displayed the prior and posterior contexts in the opposite order required for a right-to-left language. The matches are displayed in two columns, the left column must be read from right-to-left first, and then the reader can continue with the second column, which begins with the target word, followed by the rest of the context (see Figure 2).

We were only recently made aware that *WordSmith* had in fact been updated to cope better with Arabic concordance. We had initially overlooked this because the version number for the latest issue of *WordSmith* had not been incremented to indicate any change. We are pleased to report that the tool does in fact work properly with Arabic text, using the correct ordering. That said, it did require a little more configuring, such as enabling right-to-left support at the level of the Windows operating system. You also need to spend some time in the *WordSmith* preferences dialogs to add Arabic to its supported language set and then select it to be used within the current session. Other Arabic colleagues who we asked to experiment with *WordSmith* had great difficulty achieving a result. The feeling was that whilst the support was clearly there, it was not necessarily as accessible as they would have liked it.

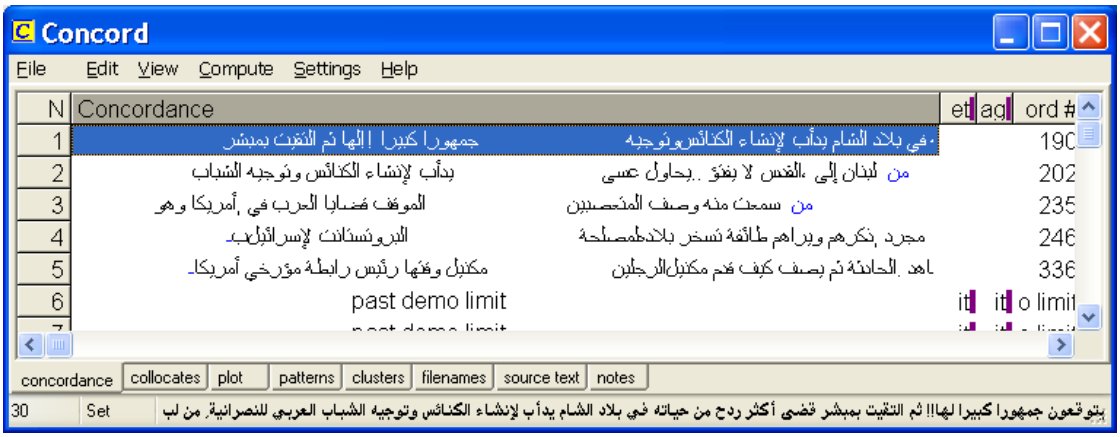

Figure 2: "Incorrect" Arabic concordance as displayed by *WordSmith* (now corrected in the latest version)

3.3 *Xaira*

Xaira is a new tool designed to supersede *SARA* (Dodd, 1997; Aston and Burnard, 1998) – an application for text analysis on the British National Corpus. *Xaira* is no longer tied to the one corpus, instead opting for a more generalised application. It takes advantage of Unicode and XML technologies to achieve this (Burnard and Dodd, 2003). At the time of writing, its first version is still in beta testing. It can perform complex searches as it can filter results according to the XML annotation.

Figure 3: Example of *Xaira* producing correct concordance output

For example, a spoken corpus is commonly annotated with information such as the gender of the current speaker. Therefore, *Xaira* is able to perform searches based on male language usage and compare it to female usage.

Xaira does not suffer any problems in that it displays the entire concordance in the correct order (see Figure 3). However, in order to get to the point of looking at results, the user must go through a relatively long procedure, using an additional tool to prepare the source texts into the format expected by the software. This need not be a problem for dedicated computational linguists, but may deter more casual users, such as linguists or language teachers wanting to use corpus and concordance evidence only occasionally.

3.4 *aConCorde*

aConCorde (Roberts and Atwell, 2005) is neither a commercial product nor a research project. In terms of features, it is relatively basic when compared to the systems already discussed. The design aim was to ensure it was as multi-lingual as possible, with extra emphasis on the Arabic language. It will be no surprise, therefore, that it works well for right-to-left languages (see Figure 4). No additional configuration of both the operating system or the application is required. *aConCorde* cannot detect the language of selected texts, but it can detect easily whether the language is left-to-right or right-to-left. Therefore, it will adjust its concordance display automatically depending on the texts currently being analysed.

An additional feature that is useful for Arabic users is the provision of an Arabic interface. Not only does this provide Arabic translations for all the menus, buttons *etc*., but even switches the entire application layout to right-to-left. In theory it is relatively easy to add additional language support, although it can be difficult to find willing translators.

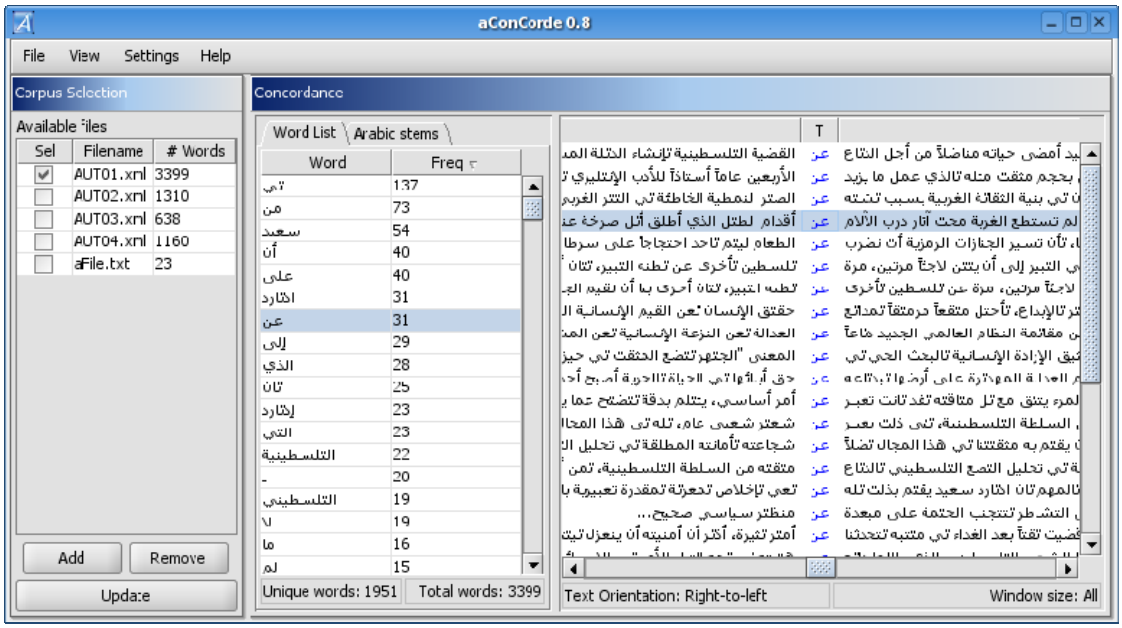

Figure 4: Example of *aConCorde* displaying Arabic concordance correctly

3.5 Discussion

The *aConCorde* project was started solely because we could not find a concordancer to work correctly with Arabic. The *Xaira* software has been demonstrated to work with Arabic too, so it could be argued that there was no need for *aConCorde*. Unfortunately, *Xaira* was only discovered *after* development had already begun. (And the same goes for the recent discovery of *WordSmith*'s recent update.) Initially, it might seem that *Xaira* and *WordSmith* have now rendered *aConCorde* redundant, but there is a market for both. Packages like *Xaira* are very sophisticated, but with increased power and flexibility comes additional complexity. It certainly lends itself to research purposes requiring complex queries for fine-grained language analysis. However, it is less appropriate in the language teaching environment. Teachers and students may be put off by the complex interface and the amount of effort needed to get a concordance output. By contrast, *aConCorde* is relatively simple to get up and running – literally select a corpus to use, then either type in your query or select *word* from the word frequency panel and you will be presented with your results. This approach could well be more suited to this audience. Also, *aConCorde* does provide special searches that are specific to Arabic analysis in the form of root/stem-based concordance, and this is not found in any other package. Finally, the *aConCorde* software is free to use by anyone (as is the source code).

4. The *aConCorde* System

The *aConCorde* project was originally conceived as a prototype to illustrate the application of new developments in software engineering, specifically Java programming language support for language-independent (and direction-independent) processing of Unicode character sets, rapid development of graphical user interfaces, and ease of integration with and into other open-source code and tools. This makes it much easier to write tools to support not just the Arabic language, but practically any language. Within one afternoon, a very simple multi-lingual concordance engine was programmed, and during the following afternoon, a basic graphical user interface was added. The user is able to switch between a native English or Arabic interface. All Arabic text is displayed correctly, all concordance is output as expected by an Arabic reader, and there is no transliteration required for the input or output. This prototype was very crude and would not have been adequate for proper use in the real world – it was simply a quick experiment to prove that the display of Arabic concordance was in fact fairly trivial. Of course, developing a more robust, feature-packed and user-friendly tool is infinitely more difficult and time consuming.

4.1 *aConCorde* features

Subsequent improvements and extensions have allowed *aConCorde* to mature into a more reliable and useful tool. The *aConCorde* system contains a number of features that make it to stand out from competing products, including:

- full Arabic support (no need to transliterate to Roman alphabet before concordance)
- English and Arabic native interfaces

- multi-platform functionality – can run on most major operating systems. It supports an extensive range of character encodings, including Unicode (UTF-16 and UTF-8), Windows Arabic (CP1256), IBM Arabic (CP420), MacArabic, ISO Latin/Arabic (ISO 8859-6) and ASCII encoding
- multi-format support that allows you to load text files, XML files, HTML files, RTF files and MS-Word files directly
- can save results either as a plain text file, or HTML file that can keep the alignment of the concordance
- word frequency analysis
- concordance that can be sorted on left or right contexts
- a range of query options: key-word, phrase, proximity, boolean, wildcard and Arabic root/stem queries, and
- *aConCorde* is freeware and open-source (released under the General Public License[6]).

The *aConCorde* system is built using the Java programming language. Java allows the programmer to produce software for the most popular operating systems without any extra effort, which is why *aConCorde* will run correctly on Microsoft Windows, Linux or Mac OS (and others) providing the user has the Java Runtime Environment installed. It is important to note that many earlier Arabic tools were reliant on Arabic Windows (a localised version of Microsoft Windows with support for Arabic script and right-to-left interfaces). *aConCorde* will of course run on standard Windows or Arabic Windows.

The Java platform is also one of the reasons why producing a multi-lingual capable application was straightforward. Internationalisation features were an important aspect of the design of Java. For instance, Java's internal mechanism to store text uses the Unicode standard, and all visual components have in-built support for left-to-right or right-to-left languages (some components can even cope with Japanese top-to-bottom text orientation).

4.2 Root- and stem-based concordance

As discussed in Section 2 the wildcard searches that make it reasonably simple for stem-based concordance in Western languages do not scale universally. To recap, an Arabic root is typically a sequence of three consonants (although two, four and five consonant roots exist) which forms the basis on which words are constructed. For example, the root *ktb* (write) has stems derived such as *katab* (write) and *iktatab* (register). From the stem you can derive the various morphological forms using affixes, infixes and suffixes, e.g., *Taktub* (she writes), *yaktub* (he writes), *aktub* (I write), and so on.

4.2.1 Buckwalter's morphological analyser

Buckwalter's morphological analyser is distributed by the LDC and has been used by many projects including the LDC's Arabic Treebank project (Maamouri *et al*., 2004).

[6] http://www.gnu.org/copyleft/gpl.html

It consists of three lexicons (affixes, suffixes and stems) and three sets of rules that specify how these lexicons can be intersected to derive Arabic word tokens. The analyser reads in a file and processes each word in isolation. By utilising the lexicons and rules, it can break down the input token and produce all possible valid morphological solutions. In cases where there is more than one solution, it does not offer any hints as to which is the most probable analysis.

The databases and the analyser work with Buckwalter's transliteration system. Until recently, direct input/output of Arabic characters in computer systems has not been trivial. Buckwalter's system is a one-to-one mapping of the Roman alphabet and the Arabic alphabet. Conveniently, there is also a one-to-one mapping to the Unicode character encoding standard.

There are other transliteration systems in use although there is no single official transliteration standard. Buckwalter's transliteration is not recognised within the Arabic linguistic community at large (Beesley, 2003), since it is relatively modern, and aimed at the computational processing of Arabic. Its use of Latin punctuation symbols for Arabic letters is clearly less intuitive to a human reader than some of the more phonetic-based transliteration systems. For example, Buckwalter uses '$' for Sheen, whereas others, such as Qalam (Heddaya, 1985), use 'sh'.

4.2.2 Buckwalter and *aConCorde*

If a user wished to produce a concordance for all words derived from the stem *katab*, a wildcard search approach is, clearly, not feasible. The alternative is to search for a manually hand-crafted list of derived words. However, with *aConCorde*, this burden is removed. Using the databases of Buckwalter's morphological analyser, *aConCorde* can provide the roots and stems *a priori* to the user. They can then select to search for one or more stems, or even everything within a given root. Buckwalter's databases were converted from Buckwalter's transliteration alphabet to Unicode so that the root/stems are displayed in native Arabic (see Figure 5).

There is a limitation here in that the user is always presented with the full database, even if many of the terms are not present within the corpus being currently analysed. When loading an Arabic corpus, it might be preferable to analyse each word and determine its stem and root, and then only have these entries visible. However, real-time stemming of a corpus in Arabic is difficult due to the complex morphology. For example, it is quite easy to remove accidentally those affixes that were in fact part of the word. That said, there are stemmers that exist, such as Khoja's (2003) stemmer used in her APT tagger.

4.3 Similar terms and word clusters

A novel feature that is specific to *aConCorde* is the use of similarity functions. These are borrowed from the information retrieval (IR) domain. They are typically used to find similar documents based on the terms contained within. In *aConCorde* its low-level model, from an IR perspective, regards sentences as "documents". The concordancer has a set of term-frequency vectors and these can be plotted into a vector space. During the concordance, the user can specify their search term, select a sample line from the concordance and its term-vector will be compared to the others

using cosine similarity (Jurafsky and Martin, 2000: 650) and then be presented with similar sentences from within the corpus.

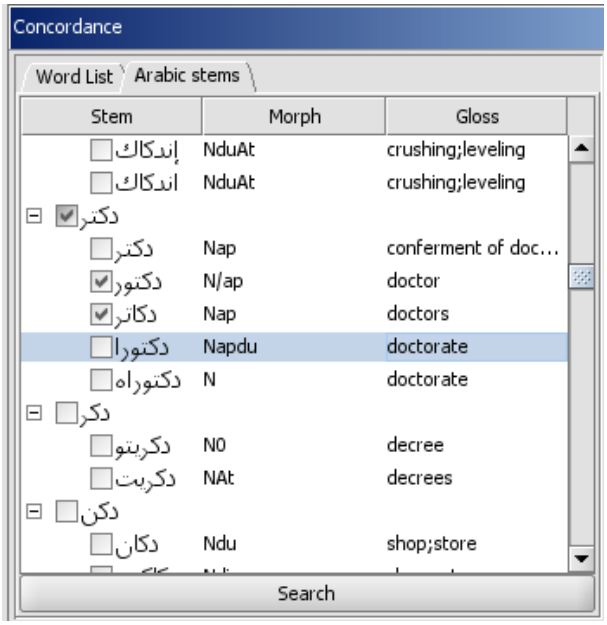

Figure 5: Example of root/stem selection within *aConCorde* using Buckwalter's stem database

4.3.1 Clustering and *aConCorde*

aConCorde provides a means for the user to interface external clustering code and display the output within the concordancer itself (see Figure 6). Section 5 illustrates the advantage of Java open-source development: more potential features could be readily added to concordance software to enable alternative perspectives and analyses of a corpus.

The term similarity and clustering features were implemented primarily for the use of concordance within the learning environment. Concordance has already proved successful within the classroom, but this concordancer can provide additional information to the user to assist in building vocabularies and grammatical patterns.

5. Clustering

The principle of clustering algorithms is to divide a set of objects into clusters. By using an appropriate measure of similarity, a good clustering algorithm will place objects that are similar into the same cluster, whereas dissimilar ones are clustered into different groups. It has the advantage of being truly unsupervised, i.e., it requires no prior knowledge about the data being clustered. Clustering is a broad subject and has been widely applied. Within computational linguistics, it has been used successfully in syntactic (Atwell and Drakos, 1987; Finch and Chater, 1992; Hughes, 1994) and semantic (Ibrahimov *et al*., 2001) classification and information retrieval systems.

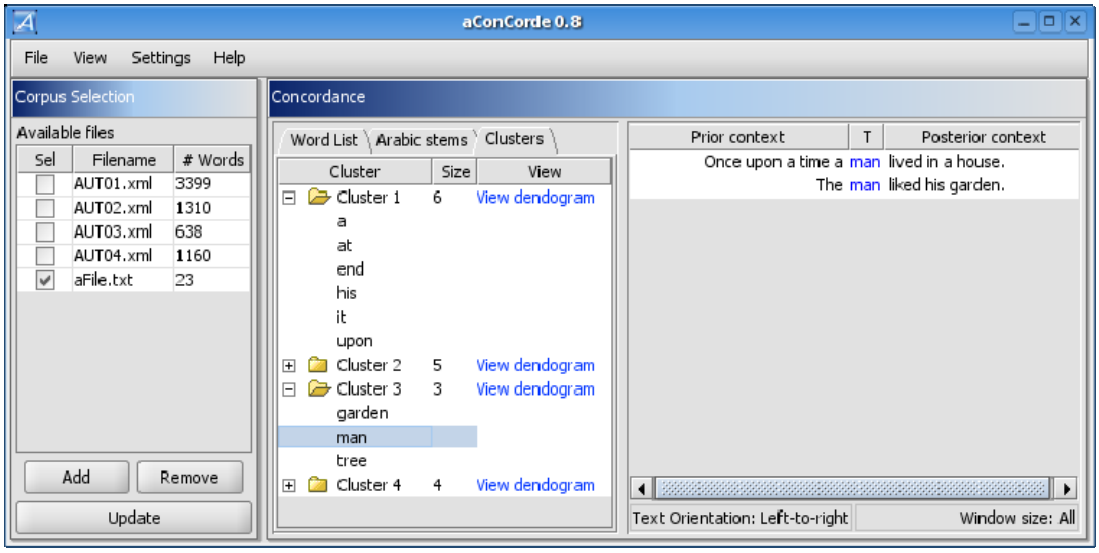

Figure 6: Cluster view within *aConCorde*

Clustering is a mathematical procedure that merges points in a vector space that are close to each other. To cluster words, a method is required to represent a given word into a vector to be plotted. The approach used here is that of Roberts (2002), which records collocation frequencies of a given content word relative to function words, for a specified window size. This data can be converted simply to a vector ready to be clustered.

5.1 Roberts' Method

Function words have an important role in language. The words themselves contain little meaning but instead specify relationships between other (content) words. Function words tend to be used frequently. The rationale behind Roberts' method is that if function words are so important and frequent, all content words within a corpus should co-occur with them a number of times, even within a small window. An hypothesis follows from this: words that co-occur similarly with the same function words could be considered grammatically similar, in that they share the same word class.

For a given corpus, *T*, you can construct a set of function words, *F* (see Section 5.2), and a set of content words, *C* (i.e., a word list of all the words in the corpus). A window size, *w*, must also be selected[7]. Select a content word, *c*, from *C* and then select a function word, *f*, from *F*. Scan the corpus for every instance of *c*. For each one, check if *f* co-occurs within the window, *w*. If it does, record the relative position from the target content word. Figure 7 illustrates such an example, showing how the content word 'first' co-occurs with the function word 'the' in a window size of four.

[7] In this paper, a window size of *n* means *n* words before and *n* words after a target word, thus spanning 2*n*+1 words in total, including the target word.

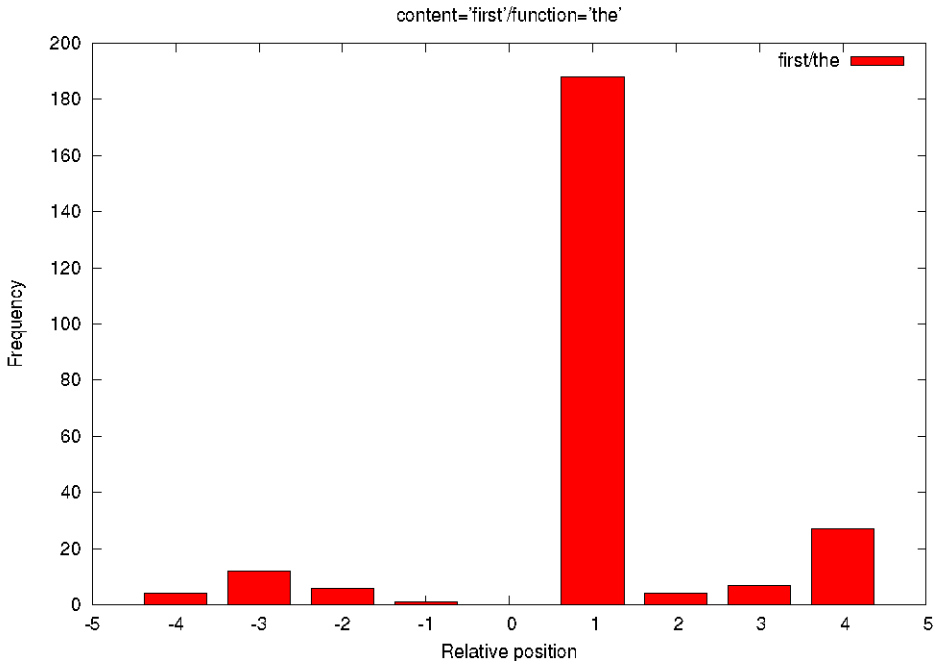

Figure 7: Graph showing how *first* is positioned relative to *the*

By treating each possible position of the target content word as a separate dimension, it is possible to represent the graph as a vector storing the frequency of occurrence for each dimension.

$$\begin{bmatrix} 4 \\ 12 \\ 6 \\ 1 \\ 0 \\ 188 \\ 4 \\ 7 \\ 27 \end{bmatrix}$$

An equivalent vector is needed for every function word in *F*, and the vectors concatenated to create a single vector that represents a profile of the content word, *w*, with respect to the set of function words, *F*. Once this has been applied to all of the words in *C*, a full set of vectors has been acquired, and these represent points in a high-dimensional vector space ready for the clustering phase.

5.2 Acquiring Function Words

The simplest way to acquire function words is to find an expert of the chosen language who can provide you with a list of them. This should be a finite, closed-class list. In practice, the list does not have to be perfect or complete, as clustering is an imperfect approximation. To acquire in a more automatic fashion, selecting the fifty most frequent words of a corpus will yield a reasonable list with adequate precision and recall. This is likely to return a few frequent content words, too.

Some function words are not that frequent, but are used consistently throughout a corpus. A fairer automatic method is to take a corpus and split it into a set of subcorpora. For each subcorpus, generate a word list, then the words that appear in all of the subcorpora are classed as function words. The exact proportions for the subcorpus size can depend on the corpus itself. For a million words, multi-genre corpus, having each subcorpus representing one percent of the overall corpus gives reasonable results (Roberts, 2006).

5.3 Clustering Algorithms

There are a range of clustering algorithms available. Hierarchical agglomerative algorithms are well-suited to word clustering. Each word in the vector space starts off as an individual one-object cluster. The algorithm evaluates each pair of points to find the closest, which are deemed the most similar, and merges them to form a single cluster. The way in which algorithms compute the distances between clusters is typically the differentiating factor (Everitt, 1993).

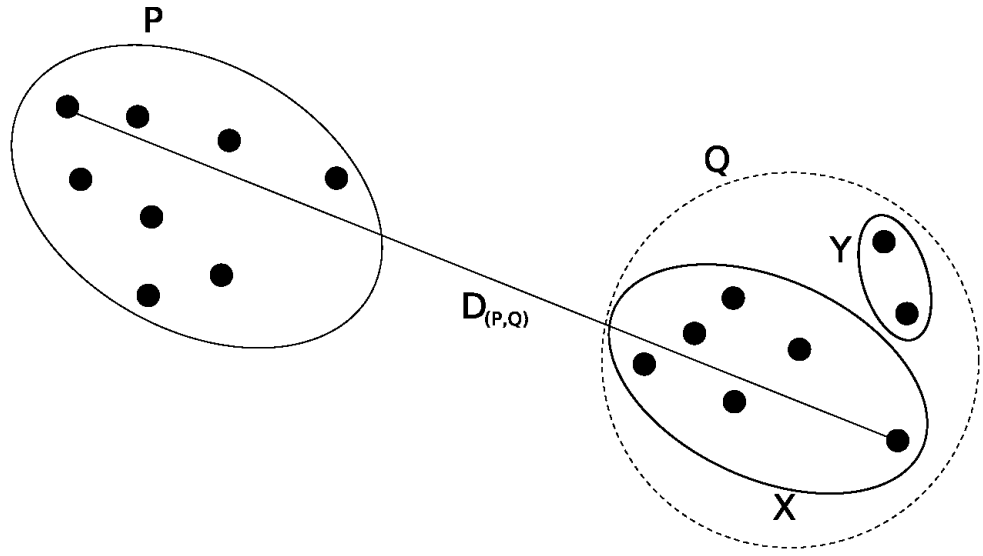

Figure 8: Diagram illustrating 'furthest neighbour' clustering algorithm

Figure 8 shows an example of the Complete Linkage algorithm. This is also known as the *furthest neighbour* method since its measure of distance between two clusters is to find the furthest two points within. Some of the more sophisticated, albeit, less simple to visualise, include Group Average and Ward's Method, which are good performers in this task (Zupan, 1982; Hughes and Atwell, 1994). The results return as a set of clusters, the size of which can vary. A nice feature of hierarchical methods is that you can profile the clustering process and generate a dendrogram that reveals how the cluster was formed, e.g., which terms were the most similar (see Figure 9).

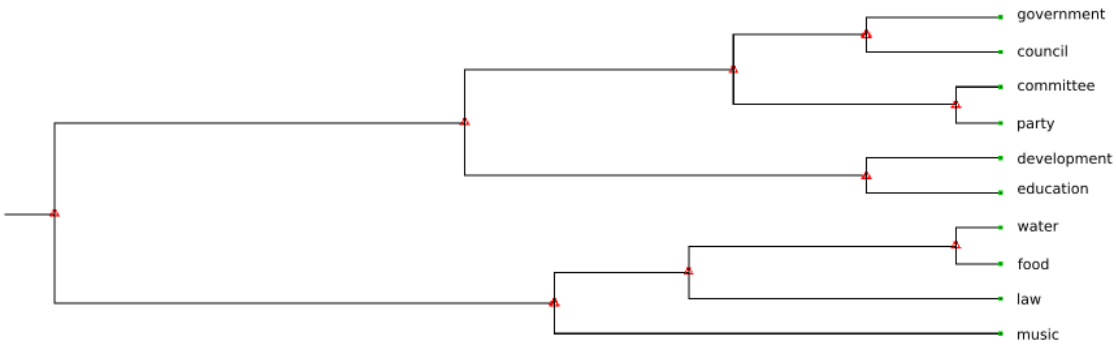

Figure 9: An example of a cluster visualised as a dendrogram

Even though clustering is an unsupervised approach, such algorithms have reported accuracies of 85+ percent in correctly grouping words into course-grained parts-of-speech (Roberts, 2002). Figures 10 and 11 show a few samples of the clustering output. A common drawback with clustering is that it can be prone to merging two well-defined clusters, (a set of nouns with a set of adjectives, for example), because at a given stage, they happened to be the 'closest'. The clustering algorithm has to know, as it were, when to stop, and this can be difficult for the program to judge automatically. If the algorithm stops too soon, the result is a large number of clusters each containing only a few words, which is unlikely to be very useful; and if the algorithm stops too late, there will be a small set of clusters each containing many words with not much (apparently) in common.

Cluster 61	
dr	
miss	
mr	
sir	
mrs	
NOUN	**100%**

Cluster 82	
her	
his	
my	
your	
herself	
him	
them	
himself	
PRON	**100%**

Cluster 49	
committee	
council	
government	
party	
development	
education	
food	
water	
law	
music	
NOUN	**100%**

Figure 10: Examples of clusters obtained from the LOB corpus

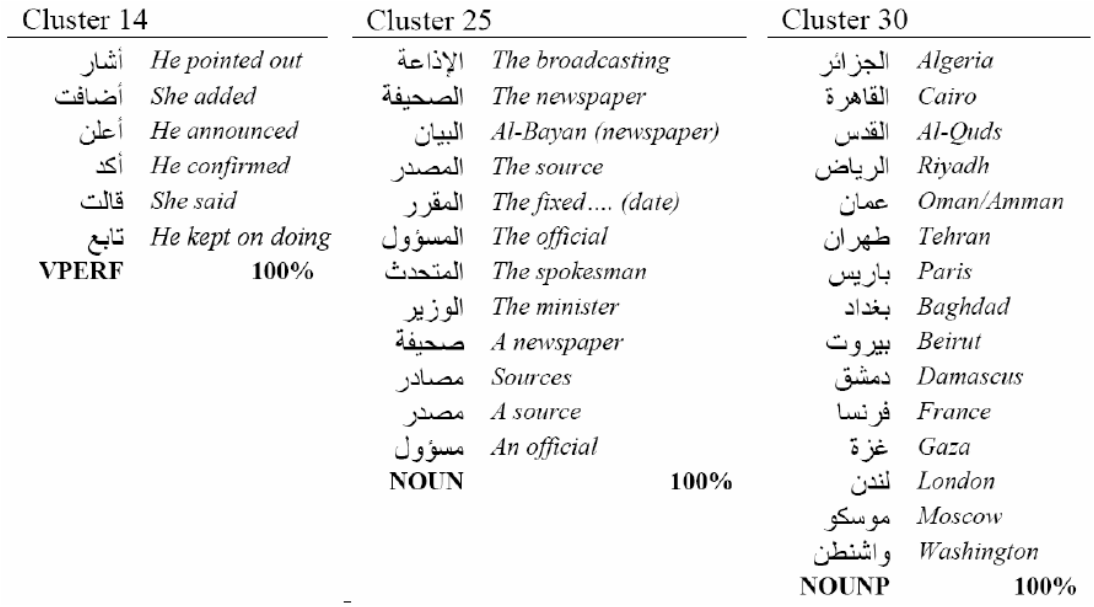

Figure 11: Examples of clusters obtained from the Arabic Treebank corpus

6. Conclusion

This paper has summarised many of the issues regarding the difficulty of Arabic concordance. The *aConCorde* project attempts to resolve some of the core issues, for example, interactive searching that displays the Arabic text correctly and stem-based searching. However, at the time of writing, *aConCorde* is still limited in terms of its features compared to the more established products on the market, these include:

- Mark-up: *aConCorde* is generally ignorant of mark-up annotation within a corpus. Whilst it can successfully parse XML and HTML files, it essentially works by filtering out the tags to isolate the content. If a corpus was annotated with part-of-speech tags *aConCorde* would not recognise them and they would be ignored (even if they were encoded in XML), or treated as "words", as if part of the text itself. Heavily annotated corpora could therefore benefit from some pre-processing to strip away such mark-up.
- Full context: *aConCorde* does not have the functionality to allow the user to see the full context of a selected concordance item. The current display is to see the word within the sentence it was found, and that is the largest scope currently implemented.
- Arabic root/stem database is always exhaustive rather than reflecting only the tokens available in the corpus that is currently loaded. This could be a disadvantage in some circumstances as the user may have to browse through many stems that have no coverage.
- Clustering integration is at present crude and takes a long time to gather and display the data.
- Computational resources: *aConCorde* can be resource intensive with large datasets. Whilst the software utilises extremely scalable indexing technologies[8] similar to those found in modern databases (e.g., binary

[8] Courtesy of the Lucene indexing library: http://lucene.apache.org/

inverted indexes), loading corpora (10,000 words per seconds), or gathering the concordance for highly frequent words, takes time (retrieving 3,000 results could take a few seconds). However, once corpora are loaded, those files do not need to be reloaded and will be available immediately for all subsequent instantiations of the concordance software.

Naturally, the development of *aConCorde* will continue, and the limitations mentioned here are high priority issues that will be addressed in future issues. However, *aConCorde* has already made significant progress. The development of a simple user-interface with novel exploration features mean that it is well suited to a learning environment. With the foundation firmly laid, we can focus on other tasks in future, for example, improving and creating fresh approaches to intuitive morphological searching, or improving word similarity exploration through machine learning methods such as clustering. We also hope that the established concordance tools will realise the demand for products that cope with Arabic script, and adapt their software accordingly.

References

Al-Sulaiti, L. 2004. Designing and Developing a Corpus of Contemporary Arabic. Masters thesis, School of Computing, University of Leeds, UK.

Al-Sulaiti, L. and Atwell, E. 2006. 'The design of a Corpus of Contemporary Arabic', International Journal of Corpus Linguistics, 11 (2).

Aston, G. and Burnard, L. 1998. The BNC Handbook: Exploring the British National Corpus with SARA. Edinburgh: Edinburgh University Press.

Atwell, E. and Drakos, N. 1987. 'Pattern recognition applied to the acquisition of a grammatical classification system from unrestricted English text' in B. Maegaard (ed.) Proceedings of EACL: the Third Conference of European Chapter of the Association for Computational Linguistics. New Jersey: ACL.

Beesley, K. 2003. Xerox Arabic Morphological Analyzer Surface-Language (Unicode) Documentation. Xerox Research Centre Europe.

Boualem, M., Leisher, M. and Ogden, B. 1999. 'Concordancer for Arabic' in Arabic Translation and Localisation Symposium, Tunis, URL: http://crl.nmsu.edu/~mleisher/concord.pdf

Buckwalter, T. 2002. Buckwalter Arabic transliteration. URL: http://www.qamus.org/transliteration.htm

Burnard, L. 2004. 'BNC-Baby and Xaira' in Proceedings of the Sixth Teaching and Language Corpora conference, p. 84, Granada.

Burnard, L. and Dodd, T. 2003. 'Xara: an XML aware tool for corpus searching' in D. Archer, P. Rayson, A. Wilson and T. McEnery (eds.) Proceedings of the Corpus Linguistics 2003 Conference, pp. 142–44. University of Lancaster: UCREL.

Cobb, T., Greaves, C. and Horst, M. 2001. 'Can the rate of lexical acquisition from reading be increased? An experiment in reading French with a suite of online resources' in P. Raymond and C. Cornaire (eds.) Regards sur la didactique des langues secondes. Montréal: Éditions Logique.

de Roeck, A. 2002. 'Arabic for the absolute beginner', ELRA Newsletter, 7 (1).

Dodd, B. 1997. 'Exploiting a corpus of written German for advanced language learning' in A. Wichmann, S. Fligelstone, G. Knowles and T. McEnery (eds.) Teaching and Language Corpora, pp. 131–45. London: Longman.

Dodd, T. 1997. SARA: Technical Manual. URL: http://www.natcorp.ox.ac.uk/sara/TechMan/

Everitt, B. 1993. Cluster Analysis (third edition). London: Edward Arnold.

Finch, S. and Chater, N. 1992. 'Bootstrapping syntactic categories' in Proceedings of the 14th Annual Meeting of the Cognitive Science Society, pp. 820–25. Hillsdale, New Jersey.

Graddol, D. 1997. The Future of English? London: British Council.

Heddaya, A. 1985. Qalam: A convention for morphological Arabic-Latin-Arabic transliteration. URL: http://eserver.org/langs/qalam.txt

Hoogland, J. 2003. The Nijmegen Arabic/Dutch dictionary project—using the concordance program. URL: http://www.let.kun.nl/wba/Content2/1.4.6_Concordancing.htm

Hughes, J. 1994. Automatically Acquiring a Classification of Words. Ph.D. thesis, School of Computing, University of Leeds.

Hughes, J and Atwell, E. 1994. 'The automated evaluation of inferred word classifications' in A Cohn (ed.) Proceedings of ECAI94: 11th European Conference on Artificial Intelligence, pp. 535–40. Chichester: John Wiley.

Ibrahimov, O., Sethi, I. and Dimitrova, N. 2001. 'Clustering of imperfect transcripts using a novel similarity measure' in Proceedings of the SIGIR'01 Workshop on Information Retrieval Techniques for Speech Applications.

Johns, T. 1990. 'From printout to handout: Grammar and vocabulary teaching in the context of data-driven learning', Computer Assisted Language Learning, 10, pp. 14–34.

Jurafsky, D. and Martin, J. 2000. Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Upper Saddle River NJ: Prentice-Hall Pearson.

Khoja, S. 2003. APT: An Automatic Arabic Part-of-Speech Tagger. Ph.D. thesis, Computing Department, Lancaster University, UK.

Lawler, J. 2000. 'Review of MonoConc Pro 2.0 concordancing software', Linguist, 11 (1411).

Leech, G. and Fligelstone, S. 1992. 'Computers and corpus analysis' in C. Butler (ed.) Computers and Written Texts, pp. 115–40. Oxford: Blackwell.

Maamouri, M., Bies, A., Buckwalter, T. and Mekki, W. 2004. 'The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus' in NEMLAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

Ogden, B. and Bernick, P. 1996. 'Oleda: User-centered Tipster technology for language instruction' in Proceedings of the Tipster Pharse II 24 Month Workshop, VA, USA.

Roberts, A. 2002. Automatic acquisition of word classification using distributional analysis of content words with respect to function words. Technical report, School of Computing, University of Leeds.

Roberts, A. 2006. Effectiveness of Intersection Method for automatic acquisition of function words. Technical report, School of Computing, University of Leeds.

Roberts, A. and Atwell, E. 2005. 'aConCorde: Towards a proper concordance of Arabic' in P. Danielsson and M. Wagenmakers (eds.) Proceedings of the Corpus Linguistics 2005 Conference, University of Birmingham, UK.

Scott, M. 2004. WordSmith Tools 4.0. URL: http://www.lexically.net/downloads/version4/html/index.html

Sinclair, J.M. 1987. Looking up; an Account of the COBUILD Project. London: Collins ELT.

Zernik, U. 1991. 'Tagging word senses in corpus' in U. Zernik (ed.) Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon, pp. 91–112. Hillsdale, NJ: Lawrence Erlbaum.

Zupan, J. 1982. Clustering of Large Data Sets. Chichester: John Wiley and Sons.