Automatic Acquisition of Training Data for Statistical Parsers

Susan Howlett and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{show0556,james}@it.usyd.edu.au

Abstract

The limitations of existing data sets for training parsers has led to a need for additional data. However, the cost of manually annotating the amount and range of data required is prohibitive. For a number of simple facts like those sought in Question Answering, we compile a corpus of sentences extracted from the Web that contain the fact keywords. We use a state-of-the-art parser to parse these sentences, constraining the analysis of the more complex sentences using information from the simpler sentences. This allows us to automatically create additional annotated sentences which we then use to augment our existing training data.

1 Introduction

Determining the syntactic structure of sentences is a necessary step in analysing the content of text for a range of language processing tasks such as Question Answering (Harabagiu et al., 2000) and Machine Translation (Melamed, 2004).

The structures that a parsing system assigns to sentences are governed by the grammar used. While some parsers make use of hand-crafted grammars, e.g. Riezler et al. (2002) and Briscoe et al. (2006), these typically cannot accommodate a wide variety of sentences and increasing coverage incurs significant development costs (Cahill et al., 2008). This has led to interest in automatic acquisition of grammars from raw text or automatically annotated data such as the Penn Treebank (Marcus et al., 1993).

Automatically-acquired grammars may be classified according to whether the learning algorithm used to estimate the language model is supervised or unsupervised. Supervised algorithms, e.g. Collins (1999) and Charniak (2000), require a large number of sentences already parsed according to the desired formalism, while unsupervised approaches, e.g. Bod (2006) and Seginer (2007), operate directly on raw text. While supervised approaches have generally proven more successful, the need for annotated training data is a major bottleneck.

Although the emergence of the Penn Treebank as a standard resource has been beneficial in parser development and evaluation, parsing performance drops when analysing text from domains other than that represented in the training data (Sekine, 1997; Gildea, 2001). In addition, there is evidence that language processing performance can still benefit from orders of magnitude more data (e.g. Banko and Brill (2001)). However, the cost of manually annotating the necessary amount of data is prohibitive.

We investigate a method of automatically creating annotated data to supplement existing training corpora. We constructed a list of facts based on factoid questions from the TREC 2004 Question Answering track (Voorhees, 2004) and the ISI Question Answer Typology (Hovy et al., 2002). For each of these facts, we extracted sentences from Web text that contained all the keywords of the fact. These sentences were then parsed using a state-of-the-art parser (Clark and Curran, 2007).

By assuming that the same grammatical relations always hold between the keywords, we use the analyses of simple sentences to constrain the analysis of more complex sentences expressing the same fact. The constrained parses then form additional training data for the parser. The results here show that parser performance has not been adversely affected; when the scale of the data collection is increased, we expect to see a corresponding increase in performance.

2 Background

The grammars used in parsing systems may be classified as either hand-crafted or automatically acquired. Hand-crafted grammars, e.g. Riezler et al. (2002) and Briscoe et al. (2006), have the advantage of not being dependent on any particular corpus, however extending them to unrestricted input is difficult as new rules and their interactions with existing rules are determined by hand, a process which can be prohibitively expensive (Cahill et al., 2008).

In contrast, the rules in acquired grammars are learned automatically from external resources, typically annotated corpora such as the Penn Treebank (Marcus et al., 1993). Here, the system automatically determines all the rules necessary to generate the structures exemplified in the corpus. Due to the automatic nature of this process, the rules determined may not be linguistically motivated, leading to a potentially noisy grammar.

Systems that learn their model of syntax from annotated corpora like the Penn Treebank are termed *supervised* systems; in *unsupervised* learning algorithms, possible sentence structures are determined directly from raw text. Bod (2006) presents an unsupervised parsing algorithm that out-performs a supervised, binarised PCFG and claims that this heralds the end of purely supervised parsing. However these results and others, e.g. Seginer (2007), are still well below the performance of state-of-the-art supervised parsers; Bod explicitly says that the PCFG used is not state-of-the-art, and that binarising the PCFG causes a drop in its performance.

By extracting the grammar from an annotated corpus, the bottleneck is shifted from writing rules to the creation of the corpus. The 1.1 million words of newswire text in the Penn Treebank has certainly been invaluable in the development and evaluation of English parsers, however for supervised parsing to improve further, more data is required, from a larger variety of texts.

To investigate the importance of training data in language processing tasks, Banko and Brill (2001) consider the problem of confusion set disambiguation, where the task is to determine which of a set of words (e.g. {*to*, *too*, *two*}) is correct in a particular context. Note that this is a problem for which large amounts of training data can be constructed extremely cheaply. They compare four different machine learning algorithms and find that with a training corpus of 1 billion words, there is little difference between the algorithms, and performance appears not to be asymptoting. This result suggests that for other language processing tasks, perhaps including parsing, performance can still be improved by large quantities of additional training data, and that the amount of data is more important than the particular learning algorithm used.

Quantity of data is not the only consideration, however. It has been shown that there is considerable variation in the syntactic structures used in different genres of text (Biber, 1993), suggesting that a parser trained solely on newswire text will show reduced performance when parsing other genres.

Evaluating parser performance on the Brown corpus, Sekine (1997) found that the best performance was achieved when the grammar was extracted from a corpus of the same domain as the test set, followed by one extracted from an equivalent size corpus containing texts from the same class (fiction or non-fiction), while parsers trained on a different class or different domain performed noticeably worse.

Comparing parsers trained on the Penn Treebank and Brown corpora, Gildea (2001) found that a small amount of training data from the same genre as the test set was more useful than a large amount of unmatched data, and that adding unmatched training data neither helped nor hindered performance.

To improve parsing performance on non-newswire genres, then, training data for these additional domains are necessary. The Penn Treebank project took 8 years and was an expensive exercise (Marcus et al., 1993), so larger and more varied annotated corpora are not likely to be forthcoming. Since unsupervised approaches still under-perform, it is necessary to explore how existing annotated data can be automatically extended, turning supervised into *semi-supervised* approaches.

3 Parsing with CCG

Combinatory Categorial Grammar (CCG; Steedman (2000)) is a mildly context sensitive, lexicalised grammar formalism, where each word in the sentence is assigned a *lexical category* (or *supertag*), either *atomic* or *complex*, which are then combined

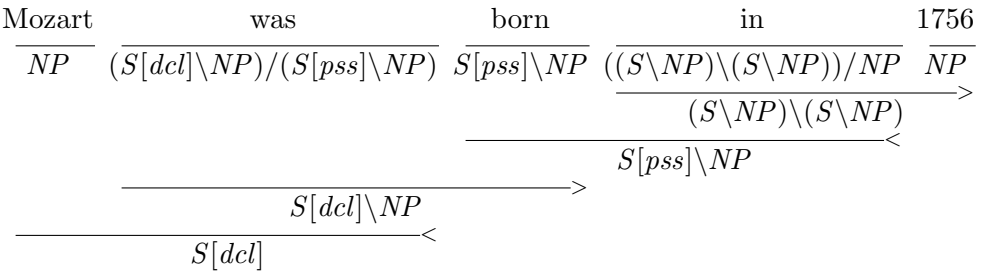

Figure 1: CCG derivation for the sentence *Mozart was born in 1756.*

using a small number of generic *combinatory rules*.

The possible atomic categories are N, NP, PP and S, representing nouns, noun phrases, prepositional phrases and clauses, respectively. They may carry additional features, e.g. $S[dcl]$ represents a declarative clause. Complex categories are binary structures consisting of two categories and a slash, in the form X/Y or $X\backslash Y$. These represent constituents that when combined with a constituent with category Y (*argument* category) form a constituent with category X (*result* category). The forward and backward slashes indicate that the Y is to be found to the right and left of X, respectively. Examples of complex categories are NP/N (determiner) and $(S\backslash NP)/NP$ (transitive verb).

The basic combinatory rules are *forward* and *backward application*, where complex categories are combined directly with their arguments to form the result. That is, $X/Y\ Y \Rightarrow X$ and $Y\ X\backslash Y \Rightarrow X$. Instances of these rules are marked by the symbols $>$ and $<$ respectively. A derivation illustrating these rules is given in Figure 1. Here, the word *in* with category $((S\backslash NP)\backslash(S\backslash NP))/NP$ combines with *1756*, an NP, using forward application to produce the constituent *in 1756* with category $(S\backslash NP)\backslash(S\backslash NP)$.

In addition, CCG has a number of rules such as *forward composition* ($>\mathbf{B}$) ($X/Y\ Y/Z \Rightarrow X/Z$) and type-raising capabilities (e.g. $X \Rightarrow T/(T\backslash X)$, represented by $>\mathbf{T}$) that increase the grammar's expressive power from context free to mildly context sensitive. These allow more complex analyses where, instead of the verb taking its subject NP as an argument, the NP takes the verb as an argument. These additional rules mean that the sentence in Figure 1 may have alternative CCG analyses, for example the one shown in Figure 2. This *spurious ambiguity* has been shown not to pose a problem to efficient parsing with CCG (Clark and Curran, 2007).

As each of the combinatory rules is applied, predicate-argument dependencies are created, of the form $\langle h_f, f, s, h_a, l\rangle$, where f is the category that governs the dependency, h_f is the word carrying the category f, s specifies which dependency of f is being filled, h_a is the head word of the argument, and l indicates whether the dependency is local or long-range. For example, in Figure 1, the word *in* results in the creation of two dependencies, both local:

$$\langle \textit{in},\ ((S\backslash NP)\backslash(S_{\textit{l}}\backslash NP))/NP_2,\ 1,\ \textit{born},\ -\rangle$$
$$\langle \textit{in},\ ((S\backslash NP)\backslash(S_{\textit{l}}\backslash NP))/NP_2,\ 2,\ \textit{1756},\ -\rangle$$

The values for s refer back to the subscripts given on the category f. Although the spurious derivations such as those in Figure 2 have a different shape, they result in the creation of the same set of dependencies, and are therefore equally valid analyses.

The predicate-argument dependencies to be formed are governed by variables associated with the categories nested within the category f. For example, *in* is more fully represented by the category $(((S_Y\backslash NP_Z)_Y\backslash(S_{Y,1}\backslash NP_Z)_Y)_X/NP_{W,2})_X$. These variables represent the heads of the corresponding constituents; the variable X always refers to the word bearing the lexical category, in this case *in*.

During the derivation, dependencies form between the word filling variable X and the word filling the variable directly associated with the dependency. In this example, the first dependency forms between *in* and the word filling variable Y, and the second forms between *in* and the word filling W. When *in* combines with *1756* through forward application, *1756* is identified with the argument $NP_{W,2}$, causing W to be filled by *1756*, and so a dependency forms between *in* and *1756*.

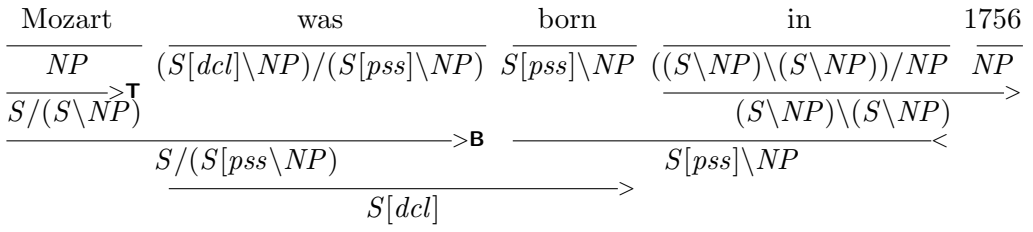

Figure 2: Alternative derivation for the sentence in Figure 1, illustrating CCG's spurious ambiguity.

4 The C&C Parser

The parser we use (Clark and Curran, 2007) is a state-of-the-art parser based on CCG. The grammar is automatically extracted from CCGbank, a conversion of the Penn Treebank into CCG (Hockenmaier, 2003). As the training data is ultimately from the Penn Treebank, the grammar is subject to the same difficulties mentioned in Section 2.

The two main components of this parser are the supertagger, responsible for assigning each word its possible CCG lexical categories, and the parser, which combines the lexical categories to form the full derivation, using the CKY chart parsing algorithm as described in Steedman (2000). The supertagger and parser are tightly integrated so that, for example, if a spanning analysis is not found, more lexical categories can be requested.

The third component, the decoder, chooses the most probable of the spanning analyses found by the parser, according to the statistical model used. Clark and Curran (2007) discuss three separate models, one of which involves finding the most probable derivation directly, while the other two involve optimising the dependency structure returned. Here we use the first model, called the normal-form model.

Clark and Curran (2007) evaluate their parser in two different ways. Their main method of evaluation is to compare the parser's predicate-argument dependency output against the dependencies in CCGbank. They calculate labelled and unlabelled precision, recall and F-score for the CCG dependencies plus category accuracy, the percentage of words assigned the correct lexical category.

This method of evaluation, however, does not allow easy comparison with non-CCG systems. Evaluating a CCG parser using the traditional metrics of precision, recall and F-score over Penn Treebank bracketings is problematic since CCG derivations, being binary-branching, can have a very different shape from the trees found in the Penn Treebank. This is particularly true of the spurious derivations, which will be heavily penalised even though they are correct analyses that lead to the correct predicate-argument dependencies.

To address this problem, Clark and Curran (2007) also evaluate their parser against the Briscoe and Carroll (2006) re-annotation of the PARC Dependency Bank (DepBank; King et al. (2003)), which contains 700 sentences from section 23 of the Penn Treebank, represented in the form of grammatical relations, e.g. `ncsubj` (non-clausal subject) and `dobj` (direct object). To do this, they convert the predicate-argument dependency output of the parser into grammatical relations, a non-trivial, many-to-many mapping. The conversion process puts the C&C parser at a disadvantage, however the its performance still rivals that of the RASP parser (Briscoe et al., 2006) that returns DepBank grammatical relations natively. Figure 3 illustrates the grammatical relations obtained from the analysis in Figure 1.

5 Getting Past the Bottleneck

Although some work has aimed at reducing the cost of training the C&C parser (Clark and Curran, 2006), the question remains of whether annotated training data can be obtained for free.

In their first entry to the TREC Question-Answering task, Brill et al. (2001) reasoned that while finding the correct answer in the given corpus may sometimes require sophisticated linguistic or logical processing, a preliminary answer found on the Web can help to identify the final answer in the corpus. They argue that the sheer size of the Web means that an answer can often be found using simple or shallow processing. We exploit the same idea of the redundancy of information on the Web here.

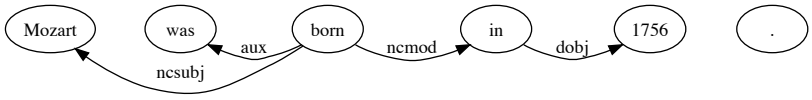

Figure 3: Grammatical relations for *Mozart was born in 1756.*

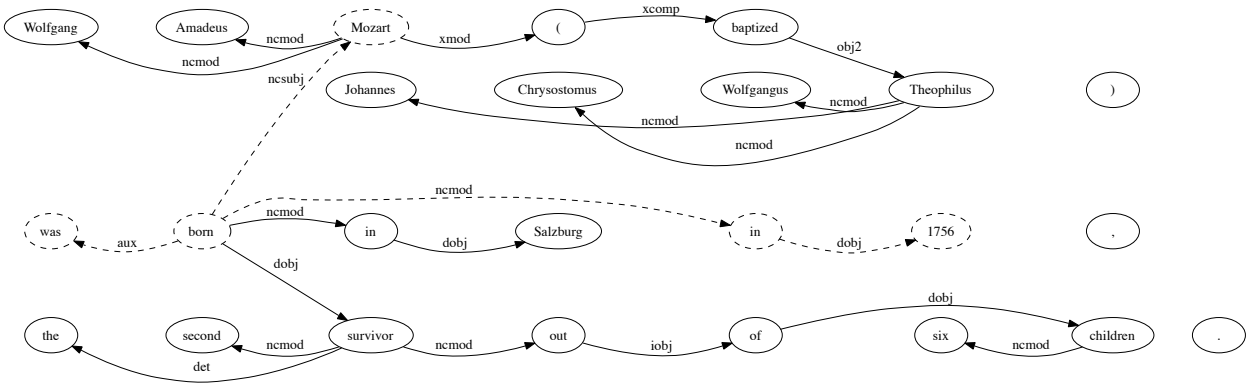

Figure 4: Desired analysis for the more complex sentence, with the analysis in Figure 3 indicated by dashed lines.

A parsing model trained on existing data can already confidently parse a simple sentence such as *Mozart was born in 1756* (Figure 3). Given the size of the Web, many such sentences should be easily found. A longer sentence, such as *Wolfgang Amadeus Mozart (baptized Johannes Chrysostomus Wolfgangus Theophilus) was born in Salzburg in 1756, the second survivor out of six children*, is more complex and thus contains more opportunities for the parser to produce an incorrect analysis. However, this more complex sentence contains the same grammatical relations between the same words as in the simple sentence (Figure 4).

Similar to the *one sense per collocation* constraint (Yarowsky, 1993), we assume that any sentence containing the words *Mozart*, *born* and *1756* will contain the same relationships between these words. We hypothesise that by constraining the parser to output an analysis consistent with these relationships, the correct analysis of the complex sentences can be found without manual intervention. These complex sentences can then be used as additional training data, allowing the parser to learn the general pattern of the sentence. For this process, we use grammatical relations rather than CCG dependencies as the former generalise across the latter and are therefore more transferable between sentences.

6 Method

Our procedure may be outlined as follows:

1. Manually select facts and identify keywords.
2. Collect sentences from HTML documents containing all keywords of a given fact.
3. Manually identify grammatical relations connecting fact keywords.
4. Parse all sentences for the fact, using these grammatical relations as constraints.
5. Add successful parses to training corpus.

Further detail is given below. Section 6.1 covers items 1 and 2, 6.2 item 3 and 6.3 items 4 and 5.

6.1 Sentence collection

First, we compiled a list of 43 facts based on factoid questions from the TREC 2004 Question Answering track (Voorhees, 2004) and the ISI Question Answer Typology (Hovy et al., 2002). In order for our hypothesis to hold, it is necessary for the facts used to refer to relatively unambiguous entities. For each fact, we identified a set of keywords and submitted these as queries through the Google SOAP Search API. The query was restricted to files with `.html` or `.htm` extensions to simplify document processing. Each unique page returned was saved.

Prior to splitting the documents into sentences, HTML tags were stripped from the text and HTML character entities replaced with their character equivalents. In order to incorporate some HTML markup information in the sentence boundary identification process, each page was split into a number of chunks by dividing at certain manually-identified HTML tags, such as heading and paragraph markers, which are unlikely to occur mid-sentence. Each of these chunks was then passed through the sentence boundary identifier available in NLTK v.0.9.3[1] (Kiss and Strunk, 2006). Ideally, the process would use a boundary identifier trained on HTML text including markup, to avoid these heuristic divisions.

To further simplify processing, sentences were discarded if they contained characters other than standard ASCII alphanumeric characters or a small number of additional punctuation characters. We also regarded as noisy and discarded sentences containing whitespace-delimited tokens that contained fewer alphanumeric characters than other characters.

Sentences were then tokenised using a tokeniser developed the C&C parser for the TREC competition (Bos et al., 2007). Sentences longer than 40 tokens were discarded as this might indicate noise or an error in sentence identification. Finally, sentences were only kept if each fact keyword appeared exactly once in the tokenised sentence, to avoid having to disambiguate between repetitions of keywords.

At the conclusion of this process, each fact had between 0 and 299 associated sentences, for a total of 3472 sentences. 10 facts produced less than 10 sentences; the average yield of the remaining 33 facts was just over 100 sentences each. This processing does result in the loss of a considerable number of sentences, however we believe the quality and level of noise in the sentences to be a more important consideration than the quantity, since there are still many facts that could yet be used.

6.2 Constraint identification

For each of the 33 facts that yielded more than 10 sentences, we identified a set of grammatical relations that connects the keywords in simple sentences. These sets contained between two and six grammatical relations; for example the relations for the Mozart fact were `(ncsubj born Mozart)`, `(ncmod born in)` and `(dobj in 1756)`. Since we assume that the keywords will be related by the same grammatical relations in all sentences (see Section 5), we use these grammatical relations as constraints on the analyses of all sentences for the corresponding fact. Future work will automate this constraint identification process.

To avoid the need to disambiguate between instances of particular words, when constraints are applied each word must be identified in the sentence uniquely. Although sentences that contained repetitions of fact keywords were discarded during the collection stage, the constraints identified in this stage of the process may incorporate additional words. In the Mozart fact, the constraints contain the word *in* in addition to the keywords *Mozart*, *born* and *1756*. The sentence in Figure 4, for example, contains two instances of *in*, so this sentence is discarded at this point, along with other sentences that contain multiple copies of a constraint word.

6.3 Parsing with constraints

Having identified the grammatical relations to use as constraints, the sentences for the corresponding fact are parsed with the constraints enforced using the process described below. Sentences from two of the 33 facts were discarded at this point because their constraints included grammatical relations of type `conj`, which cannot be enforced by this method.

As described in Section 3, dependencies are formed between variables on lexical categories. For example, a dependency is created between the words *in* and *1756* in Figure 1 by filling a variable W in the category of *in*. To force a particular dependency to be formed, we determine which of the two words will bear the category that lists the dependency; this word we call the *head*. We identify the variable associated with the desired dependency and pre-fill it with the second word. The parser's own unification processes then ensure that the constraint is satisfied.

Since the constraints are expressed in terms of grammatical relations and not CCG dependencies, each constraint must first be mapped back to a number of possible dependencies. First, the head word may be assigned several possible categories by the supertagger. In this case, if any category does not license the desired dependency, it is removed from

[1] http://nltk.sourceforge.net

Keywords	# Pages	# Sents	Normal # Parse	Normal # Fail	Constrained # Discarded	Constrained # Parse	Constrained # Fail
Armstrong landed Moon 1969	805	20	20	0	14	0	6
CNN Cable News Network	286	62	62	0	41	8	13
Canberra capital Australia	601	78	77	1	64	0	14
Columbus discovered America	683	299	291	8	0	174	125
Martin Luther King Jr assassinated 1968	675	6	6	0	–	–	–
Mozart born 1756	734	161	160	1	93	30	38
hydrogen lightest element	582	85	83	2	51	9	25
Total (all 43 facts)	24934	**3472**	3421	51	–	–	–
Total (31 facts for which constraints successfully applied)	19019	**3237**	3190	47	1661	**662**	914

Table 1: Number of sentences acquired for a selection of facts. Figures are given for the number of sentences parsed and failed by the normal (unconstrained) parser as an indication of parser coverage on Web sentences.

the list; all others have a variable filled. In this way, no matter which category is chosen, the constraint is enforced. Secondly, one grammatical relation may map to some dependencies governed by the first word and some governed by the second. That is, in some constraints, either word may operate as the head. To account for this, we parse the sentence several times, with different combinations of head choices, and keep the highest probability analysis.

When parsing with constraints, not all sentences can be successfully parsed. Those where a spanning analysis consistent with the constraints cannot be found are discarded; the remainder are added to the training corpus. From the 31 facts still represented at this stage, we obtained 662 sentences.

7 Results

Table 1 traces the quantity of data throughout the process described in Section 6, for a sample of the facts used. It also gives totals for all 43 facts and for the 31 facts represented at completion.

The first column lists the fact keywords. We used a range of fact types, including abbreviations (CNN stands for Cable News Network) and identities (Canberra is the capital of Australia, hydrogen is the lightest element) as well as actions (Armstrong landed on the Moon in 1969) and passive structures (Martin Luther King, Jr. was assassinated in 1969). This last example is one where fewer than 10 sentences were collected, most likely due to the large number of keywords.

The # Pages column in Table 1 shows the number of unique Web pages saved for each fact, while # Sents indicates the number of unique sentences containing all fact keywords, after tokenisation. The next two figures show the coverage of the baseline parser on the Web sentences. The # Discarded column indicates the number of these sentences that were discarded during the constraining process due to constraint words being absent or appearing more than once. The final two columns show how many of the remaining sentences the parser found a spanning analysis for that was consistent with the constraints.

It is clear that there is considerable variation in the number of sentences discarded during the constraining process. For some facts, such as the Columbus fact, the keywords form a chain of grammatical relations with no additional words necessary. In such cases, no sentences were discarded since any sentences that reached this point already met the unique keyword restriction.

The main reason for sentences being discarded during constraining is that some relationships are expressed in a number of different ways. Prepositions and punctuation are frequently responsible for this. For example, the date of the Moon landing can be expressed as both *in 1969* and *on July 20, 1969*, while the expansion of the abbreviation CNN may be expressed as *CNN: Cable News Network*, *CNN (Cable News Network)* and indeed *Cable News Net-*

Model	LP	LR	LF	LF (POS)	SENT ACC	UP	UR	UF	CAT ACC	COV
Baseline	85.53	84.71	85.12	83.38	32.14	92.37	91.49	91.93	93.05	99.06
New	85.64	84.77	85.21	83.54	32.03	92.41	91.47	91.94	93.08	99.06

Table 2: Performance of the baseline (Clark and Curran (2007) normal-form model) and new parser models on CCGbank section 23. Most results use gold standard POS tags; LF (POS) uses automatically assigned tags.

work (CNN). By selecting only one set of constraints, only one option can be used; sentences involving other formats are therefore discarded. To overcome these difficulties, it would be necessary to allow several different constraint chains to be applied to each sentence, taking the highest probability parse from all resulting analyses. This could also be used to overcome the word-uniqueness restriction.

To evaluate the effectiveness of the 662 additional sentences, we trained two models for the C&C parser, a baseline using the parser's standard training data, sections 02–21 of CCGbank, and a second model using CCGbank plus the extra data. For details of the training process, see Clark and Curran (2007). Following Clark and Curran (2007), we evaluate the performance of the two parsing models by calculating labelled and unlabelled precision, recall and F-score over CCGbank dependencies, plus coverage, category accuracy (percentage of words with the correct lexical category) and sentence accuracy (percentage of sentences where all dependencies are correct).

The difficulty we face here is that the additional data is Web text, while the evaluation data is still newswire text. Since genre effects are important, we cannot expect to see a large difference in results in this evaluation. Future work will compare the two models on a corpus of Web sentences collected according to the same procedure, using different facts.

Table 2 summarises the results of our evaluation. The figures for the performance of the baseline system are the latest for the parser, and slightly higher than those given in Clark and Curran (2007). It is clear that the results for the two systems are very similar and that adding 662 sentences, increasing the amount of data by approximately 1.7%, has had a very small effect. However, now that the automated procedure has been developed, we are in a position to substantially increase this amount of data without requiring manual annotation.

8 Conclusion

We have described a procedure for automatically annotating sentences from Web text for use as training data for a statistical parser. We assume that if several sentences contain the same set of relatively unambiguous keywords, for example *Mozart*, *born* and *1756*, then those words will be connected by the same chain of grammatical relations in all sentences. On this basis, we constrain a state-of-the-art parser to produce analyses for these sentences that contain this chain of relations. The constrained analyses are then used as additional training data for the parser.

Aside from the initial identification of facts, the only manual step of this process is the the choice of constraints. The automation of this step will involve the identification of reliable sentences, most likely short sentences consisting of only a single clause, from which the relation chain can be extracted. The chain will need to be supported by a number of such sentences in order to be accepted. This is the step that truly exploits the redundancy of information on the Web, as advocated by Brill et al. (2001).

Once the choice of constraints has been automated, we will have a fully automatic procedure for creating additional annotated training data for a statistical parser. Our initial results show that the training data acquired in this manner can be used to augment existing training corpora while still maintaining high parser performance. We are now in a position to increase the scale of our data collection to determine how performance is affected by a training corpus that has been significantly increased in size.

Acknowledgements

We would like to thank the anonymous reviewers and the Language Technology Research Group at the University of Sydney for their comments. The first author is supported by a University of Sydney Honours scholarship.

References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33.

Douglas Biber. 1993. Using register-diversified corpora for general language studies. *Computational Linguistics*, 19(2):219–241, June.

Rens Bod. 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*, pages 865–872.

Johan Bos, James R. Curran, and Edoardo Guzetti. 2007. The Pronto QA system at TREC-2007. In *Proceedings of the Sixteenth Text REtreival Conference*.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Y. Ng. 2001. Data-intensive question answering. In *Proceedings of the Tenth Text REtreival Conference*, pages 393–400, November.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 41–48, July.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia, July.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124, March.

Eugene Charniak. 2000. A maxmium entropy inspired parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 144–151, June.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, December.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Milhalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtreival Conference*.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Eduard Hovy, Ulf Hermjakob, and Deepak Ravichandran. 2002. A question/answer typology with surface text patterns. In *Proceedings of the DARPA Human Language Technology conference*, pages 247–251.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora*, pages 1–8.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 653–660.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional gramar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 384–391, Prague, Czech Republic, June.

Satoshi Sekine. 1997. The domain dependence of parsing. In *Proceedings of the fifth conference on Applied Natural Language Processing*, pages 96–102.

Mark Steedman. 2000. *The Syntactic Process*. Language, Speech, and Communication series. The MIT Press, Cambridge, MA.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Question Answering track. In *Proceedings of the Thirteenth Text REtreival Conference*.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271.