Evaluating and Integrating Treebank Parsers on a Biomedical Corpus

Andrew B. Clegg and Adrian J. Shepherd
School of Crystallography
Birkbeck College
University of London
London WC1E 7HX, UK
{a.clegg,a.shepherd}@mail.cryst.bbk.ac.uk

Abstract

It is not clear *a priori* how well parsers trained on the Penn Treebank will parse significantly different corpora without retraining. We carried out a competitive evaluation of three leading treebank parsers on an annotated corpus from the human molecular biology domain, and on an extract from the Penn Treebank for comparison, performing a detailed analysis of the kinds of errors each parser made, along with a quantitative comparison of syntax usage between the two corpora. Our results suggest that these tools are becoming somewhat over-specialised on their training domain at the expense of portability, but also indicate that some of the errors encountered are of doubtful importance for information extraction tasks.

Furthermore, our inital experiments with unsupervised parse combination techniques showed that integrating the output of several parsers can ameliorate some of the performance problems they encounter on unfamiliar text, providing accuracy and coverage improvements, and a novel measure of trustworthiness.

Supplementary materials are available at http://textmining.cryst.bbk.ac.uk/acl05/.

1 Introduction

The availability of large-scale syntactically-annotated corpora in general, and the Penn Treebank[1] (PTB; Marcus et al., 1994) in particular, has enabled the field of stochastic parsing to advance rapidly over the course of the last 10-15 years. However, the newspaper English which makes up the bulk of the PTB is only one of many distinct genres of writing in the Anglophone world, and certainly not the only domain where potential natural-language processing (NLP) applications exist that would benefit from robust and reliable syntactic analysis. Due to the massive glut of published literature, the biomedical sciences in general, and molecular biology in particular, constitute one such domain, and indeed much attention has been focused recently on NLP in this area (Shatkay and Feldman, 2003; Cohen and Hunter, 2004).

Unfortunately, annotated corpora of a large enough size to retrain stochastic parsers on do not exist in this domain, and are unlikely to for some time. This is partially due to the same differences of vocabulary and usage that set biomedical English apart from the *Wall Street Journal* in the first place; these differences necessitate the input of both biological and linguistic knowledge on biological corpus annotation projects (Kulick et al., 2004), and thus require a wider variety of annotator skills than general-English projects. For example, *5′* (pronounced "five-prime") is an adjective in molecular biology, but *p53* is a noun; *amino acid*

[1] http://www.cis.upenn.edu/~treebank/

is an adjective-noun sequence[2] but *cadmium chloride* is a pair of nouns. These tagging decisions would be hard to make correctly without biological background knowledge, as would the prepositional phrase attachment decisions in Figure 1.

Although it is intuitively apparent that there are differences between newspaper English and biomedical English, and that these differences are quantifiable enough for biomedical writing to be characterised as a sublanguage of English (Friedman et al., 2002), the performance of conventionally-trained parsers on data from this domain is to a large extent an open question. Nonetheless, papers have begun to appear which employ treebank parsers on biomedical text, essentially untested (Xiao et al., 2005). Recently, however, the GENIA project (Kim et al., 2003) and the Mining the Bibliome project (Kulick et al., 2004) have begun producing small draft corpora of biomedical journal paper abstracts with PTB-style syntactic bracketing, as well as named-entity and part-of-speech (POS) tags. These are not currently on a scale appropriate for retraining parsers (compare the ~50,000 words in the GENIA Treebank to the ~1,000,000 in the PTB; but see also Section 7.2) but can provide a sound basis for empirical performance evaluation and analysis. A collection of methods for performing such an analysis, along with several interesting results and an investigation into techniques for narrowing the performance gap, is presented here.

1.1 Motivation

We undertook this project with the intention of addressing several questions. Firstly, in order to deploy existing parsing technologies in a bioinformatics setting, the biomedical NLP community needs a comprehensive assessment of performance – which parser(s) to choose, what accuracy each should be expected to achieve etc., along with information about the different situations in which each parser can be expected to perform well or poorly. Secondly, assuming there is a performance deficit, can any simple steps be taken to mitigate it? Thirdly, what engineering issues arise from the idiosyncracies of biomedical text?

The differences discovered in the behaviour of each parser, either between domains or between different software versions on the same domain, will also be of interest to those in the computational linguistics community who are involved in parser design. These values will give a comparative index of the flexibility of each parsing model on being presented with out-of-domain data, and may help parser developers to detect signs of overtraining or, analogously, 'over-design' for one narrow genre of English. It is hoped that our findings can assist those better equipped than ourselves in properly investigating these phenomena, and that our analysis of the problems encountered can shed new light on the thorny problem of parser evaluation.

Finally, several questions arise from the use of multiple parsers on the same corpus that are of both theoretical and practical interest. Does agreement between several parsers indicate that a sentence has been parsed correctly, or do they tend to make the same mistakes? How best can the output of an ensemble of parsers be integrated, in order to boost performance above that of the best single member? And what additional information can be gleaned from comparing the opinions of several parsers that can help make sense of unfamiliar text?

2 Evaluation methodologies

We initially chose to rate the parsers in our assessment by several different means which can be grouped into two broad classes: constituent- and lineage-based. While Sampson and Babarczy (2003) showed that there is a limited degree of correlation between the per-sentence scores assigned by the two methods, they are independent enough that a fuller picture of parser competence can be built up by combining them and thus sidestepping the drawbacks of either approach. However, overall performance scores designed for competitively evaluating parsers do not provide much insight into the aetiology of errors and anomalies, so we developed a third approach based on production rules that enabled us to mine the megabytes of syntactic data for enlightening results more ef-

[2] According to some annotators at least; others tag *amino* as a noun, although one would not speak of **an amino*, **some amino* or **several aminos*.

a. *[This protein] [binds the DNA [by the TATA box [on its minor groove.]$_2$]$_1$]*

b. *[This protein] [binds the DNA [by the TATA box]$_1$ [at its C-terminal domain.]$_2$]*

Figure 1: These two sentences are biologically clear but syntactically ambiguous. Only the knowledge that the C-terminal domain is part of a protein, whereas the TATA box and minor groove are parts of DNA, allows a human to interpret them correctly, by attaching the prepositional phrases 1 and 2 at the right level.

fectively. All the Perl scoring routines we wrote are available from our website.

2.1 Constituent-based assessment

Most evaluations of parser performance are based upon three primary measures: labelled constituent precision and recall, and number of crossing brackets per sentence. Calculation of these scores for each sentence is straightforward. Each constituent in a candidate parse is treated as a tuple $\langle lbound, LABEL, rbound\rangle$, where *lbound* and *rbound* are the indices of the first and last words covered by the constituent. Precision is the proportion of candidate constituents that are correct and is calculated as follows:

$$P = \frac{\#\text{ true positives}}{\#\text{ true positives} + \#\text{ false positives}}$$

Recall is the proportion of constituents from the gold standard that are in the candidate parse:

$$R = \frac{\#\text{ true positives}}{\#\text{ true positives} + \#\text{ false negatives}}$$

The crossing brackets score is reached by counting the number of constituents in the candidate parse that overlap with at least one constituent in the gold standard, in such a way that one is not a subsequence of the other.

Although this scoring system is in wide use, it is not without its drawbacks. Most obviously, it gives no credit for partial matches, for example when a constituent in one parse covers most of the same words as the other but is truncated or extended at one or both ends. Indeed, one can imagine situations where a long constituent is truncated at one end and extended at the other compared to the gold standard; this would incur a penalty under each of the above metrics even though some or even most of the words in the constituent were correctly categorised. One can of course suggest modifications for these measures designed to account for particular situations like these, although not without losing some of their elegance. The same is true for label mismatches, where a constituent's boundaries are correct but its category is wrong.

More fundamentally, it could be argued that by taking as it were horizontal slices through the syntax tree, these measures lose important information about the ability of a parser to recreate the gross grammatical structure of a sentence. The height of a given constituent in the tree, and the details of its ancestors and descendants, are not directly taken into account, and it is surely the case that these broader phenomena are at least as important as the extents of individual constituents in affecting meaning. However, constituent-based measures are not without specific advantages too. These include the ease with which they can be broken down into scores per label to give an impression of a parser's performance on particular kinds of constituent, and the straightforward message they deliver about whether a badly-performing parser is tending to over-generate (low precision), under-generate (low recall) or mis-generate (high crossing brackets).

2.2 Lineage-based assessment

In contrast to this horizontal-slice philosophy, Sampson and Babarczy (2003) advocate a vertical view of the syntax tree. By walking up the tree structure from the immediate parent of a given word until the top node is reached, and adding each label encountered to the end of a list, a 'lineage' representing the word's ancestry can be retrieved. Boundary symbols are inserted into this lineage before the highest constituent that begins on the word, and after the highest constituent that ends on the word, if such conditions apply; this allows potential ambiguities to be avoided, so that the tree as a whole has one and only one corresponding set of 'lineage strings' (see Figure 2).

Using dynamic programming, a Levenshtein edit distance can be calculated between each word's lineage strings in the candidate parse and the gold standard, by determining the smallest number of symbol insertions, deletions and substitutions required to transform one of the strings into the other. The leaf-ancestor (LA) metric, a similarity score ranging between 0 (total parse failure) and 1 (exact match), is then calculated by taking into account the lengths of the two lineages:

$$LA = 1 - \frac{dist(lineage_1, lineage_2)}{len(lineage_1) + len(lineage_2)}$$

The per-word score can then be averaged over a sentence or a whole corpus in order to arrive at an overall performance indicator. Besides avoiding some of the limitations of constituent-based evaluation discussed above, one major advantage of this approach is that it can provide a word-by-word measure of parser performance, and thus draw attention easily to those regions of a sentence which have proved problematic (see Section 6.2 for an example). The algorithm can be made more sensitive to near-matches between phrasal categories by tuning the cost incurred for a substitution between similar labels, e.g. those for 'singular noun' and 'proper noun', rather than adhering to the uniform edit cost dictated by the standard Levenshtein scheme. In order to avoid over-complicating this study, however, we chose to keep the standard penalty of 1 for each insertion, deletion or substitution.

One drawback to leaf-ancestor evaluation is that although it scores each word (sentence, corpus) between 0 and 1, and these scores are presented here as percentages for readability, it is misleading to think of them as percentages of correctness in the same way that one would regard constituent precision and recall. Indeed, the very fact that it results in a single score means that it reveals less at first glance about the broad classes of errors that a parser is making than precision, recall and crossing brackets do. Another possible objection is that since an error high in the tree will affect many words, the system implicitly gives most weight to the correct determination of those features of a sentence which are furthest from being directly observable. One might argue, however, that since a high-level attachment error can grossly perturb the structure of the tree and thus the interpretation of the sentence, this is a perfectly valid approach; it is certainly complementary to the uniform scoring scheme described in the previous section, where every mistake is weighted identically.

2.3 Production-based assessment

In order to properly characterise the kinds of errors that occurred in each parse, and to help elucidate the differences between multiple corpora and between each parser's behaviour on each corpus, we developed an additional scoring process based on production rules. A production rule is a syntactic operation that maps from a parent constituent in a syntax tree to a list of daughter constituents and/or POS tags, of the general form:

$$LABEL_p \rightarrow LABEL_1 \ldots LABEL_n$$

For example, the rule that maps from the topmost constituent in Figure 2 to its daughters would be `S → NP VP`. A production is the application of a production rule at a particular location in the sentence, and can be expressed as:

$$LABEL_p(lbound, rbound) \rightarrow LABEL_1 \ldots LABEL_n$$

Production precision and recall can be calculated as in a normal labelled constituent-based assessment, except that a proposed production is a true positive if and only if there exists a production in the gold standard with the same parent label and boundaries, and the same daughter labels in the same order. (The respective widths of the daughter constituents, where applicable, are not taken into account, only their labels and order; any errors of width in the daughters are detected when they are tested as parents themselves.)

Furthermore, as an aid to the detection and analysis of systematic errors, we developed a heuristic

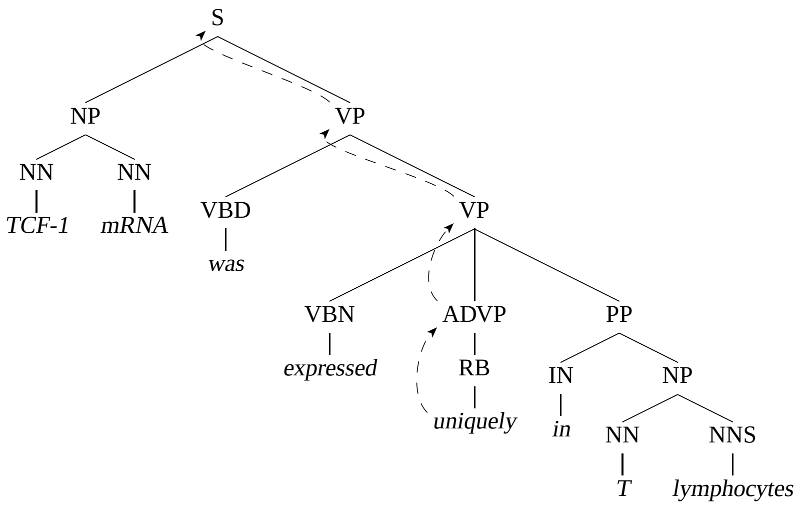


Figure 2: Skipping the POS tag, the lineage string for *uniquely* is: `[ ADVP ] VP VP S` . The left and right boundary markers record the fact that the `ADVP` constituent both starts and ends with this word.

for finding the closest-matching candidate productions $PROD_{c1} \ldots PROD_{cm}$ in a parse, in each case where a production $PROD_g$ in the gold standard is not exactly matched in the parse.

1. First, the heuristic looks for productions with correct boundaries and parent labels, but incorrect daughters. The corresponding production rules are returned.

2. Failing that, it looks for productions with correct boundaries and daughters, preserving the order of the daughters, but with incorrect parent labels. The corresponding production rules are returned.

3. Failing that, it looks for productions with correct boundaries but incorrect parent labels and daughters. The corresponding production rules are returned.

4. Failing that, it looks for all extensions and truncations of the production (boundary modifications such that there is at least one word from $PROD_g$ still covered) with correct parent and daughter labels and daughter order, keeping only those that are closest in width to $PROD_g$ (minimum number of extensions and truncations). The meta-rules `EXT_ALLMATCH` and/or `TRUNC_ALLMATCH` as appropriate are returned.

5. Failing that, it looks for all extensions and truncations of the production where the parent label is correct but the daughters are incorrect, keeping only those that are closest in width to $PROD_g$. The meta-rules `EXT_PARENTMATCH` and/or `TRUNC_PARENTMATCH` are returned.

6. If no matches are found in any of these classes, a null result is returned.

Note that in some cases, m production rules of the same class may be returned, for example when the closest matches in the parse are two productions with the correct parent label, one of which is one word longer than $PROD_g$, and one of which is one word shorter. It is also conceivable that multiple productions with the same parent or same daughters could occupy the same location in the sentence without branching, although it seems unlikely that this would occur apart from in pathologically bad parses. In any ambiguous cases, no attempt is made to decided which is the ‘real’ closest match; all m matches are returned, but they are downweighted so that each counts as $1/m$ of an error when error frequencies are calculated. In no circumstances are matches from different classes returned.

The design of this procedure reflects our requirements for a tool to facilitate the diagnosis and summarisation of parse errors. We wanted to be able to answer questions like "given that parser A has a low recall for `NP` $\rightarrow$ `NN NN` productions, what syntactic structures is it generating in their place? Why might this be so? And what effect might these errors have on the interpretation of the sentence?" Accordingly, as the heuristic casts the net further and further to find the closest match for a production $PROD_g$, the classes to which it assigns errors become broader and broader. Any match at stages 1–3 is not simply recorded as a substitution error, but a substitution for a *particular* incorrect production rule. However, matches at stages 4 and 5 do not make a distinction between different magnitudes of truncation and extension, and at stage 5 the information about the daughters of incorrect productions is discarded. This allowed us to identify broad trends in the data even where the correspondences between the gold standard and the parses were weak, yet nonetheless recover detailed substitution information akin to confusion matrices where possible.

Similar principles guided the decision not to consider extensions and truncations with different parent labels as potential loose matches, in order to avoid uninformative matches to productions elsewhere in the syntax tree. In practice, the matches returned by the heuristic accounted for almost all of the significant systematic errors suffered by the parsers (see Section 6) – null matches were infrequent enough in general that their presence in larger numbers on certain production rules was itself useful from an explanatory point of view.

2.4 Alternative approaches

Several other proposed solutions to the evaluation problem exist, and it is an ongoing and continually challenging field of research. Suggested protocols based on grammatical or dependency relations (Crouch et al., 2002), head projection (Ringger et al., 2004), alternative edit distance metrics (Roark, 2002) and various other schemes have been suggested. Many of these alternative methodologies, however, suffer from one or more disadvantages, such as specificity to one particular grammatical formalism (e.g. head-driven phrase structure grammar) or one class of parser (e.g. partial parsers), or a requirement for a specific manually-prepared evaluation corpus in a non-treebank format. In addition, none of them deliver the richness of information supplied by production-based assessment, particularly in combination with the other methods outlined above.

3 Comparing the corpora

The gold standard data for our experiments was drawn from the GENIA Treebank[3], a beta-stage corpus of 200 abstracts drawn randomly from the MEDLINE database[4] with the search terms "human", "blood cell" and "transcription factor". These abstracts have been annotated with POS tags, named entity classes and boundaries[5], and syntax trees which broadly follow the conventions of the PTB. Some manual editing was required to correct annotation errors and remove sentences with uncorrectable errors, leaving 1757 sentences (45406 tokens) in the gold standard. All errors were reported to the GENIA group.

For comparison purposes, we used the standard set-aside test set from the PTB, section 23. This consists of 56684 words in 2416 sentences.

To gain insight into the differences between the two corpora, we ran several tests of the grammatical composition of each. For consistency with the parser evaluation results, we stripped the following punctuation tokens from the corpora before gathering these statistics: period, comma, semicolon, colon, and double-quotes (whether they were expressed as a single double-quotes character, or pairs of opening or closing single-quotes). We also removed any super-syntactic information such as grammatical function suffixes, pruned any tree branches that did not contain textual terminals (e.g. traces), and deleted any duplicated constituents – that is, constituents with only one daughter that has the same label.

3.1 Sentence length and complexity

Having performed these pre-processing steps, we counted the distributions of sentence lengths (in

[3] `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/topics/Corpus/GTB.html`

[4] `http://www.pubmed.org/`

[5] The named entity annotations are supplied in a separate file which was discarded.

words) and sentence complexities, using the number of constituents, not counting POS tags, as a simple measure of complexity – although of course one can imagine various other ways to gauge complexity (mean tree depth, maximum tree depth, constituents per word etc.). The results are shown in Figure 3, and reveal an unexpected level of correlation. Apart from a few sparse instances at the right-hand tails of the two GENIA distributions, and a single-constituent spike on the PTB complexity distribution (due to one-phrase headings like *STOCK REPORT.*), the two corpora have broadly similar distributions of word count and constituent count. The PTB has slightly more mass on the short end of the length scale, but GENIA does not have a corresponding number of longer sentences. This ran contrary to our initial intuition that newspaper English would tend to be composed predominantly of shorter and simpler sentences than biological English.

3.2 Constituent and production rule usage

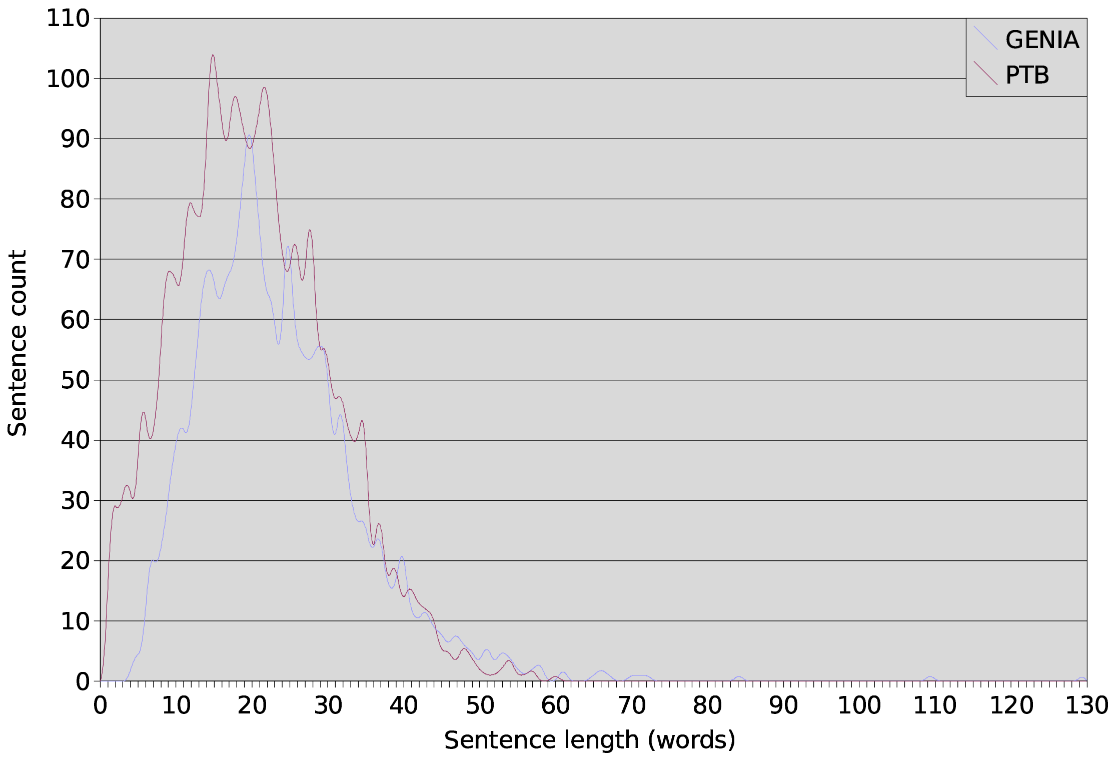
Next, we counted the frequency with which each constituent label appears in each corpus. The results are shown in Figure 4. The distributions are reasonably similar between the two corpora, with the most obvious difference being that GENIA uses noun phrases more often, by just over six percentage points. This may reflect the fact that much of the text in GENIA describes interactions between multiple biological entities at the molecular and cellular levels; conjunction phrases are three times as frequent in GENIA too, although this is not obvious from the chart as the numbers are so low in each corpus.

One surprising result is revealed by looking at Table 1 which shows production rule usage across the corpora. Although GENIA uses slightly more productions per sentence on average, it uses marginally fewer *distinct production rules* per sentence, and considerably fewer overall – 62% of the number of rules used in the PTB, despite being 73% of the size in sentences. These figures, along with the significantly different rankings and frequencies of the actual rules themselves (Table 2), demonstrate that there are important syntactic differences between the corpora, despite the similarities in length, complexity and constituent usage. Such differences are invisible to conventional constituent-based analysis.

The comparative lack of syntactic diversity in GENIA may seem counter-intuitive, since biological language seems at first glance dense and difficult. However, it must be remembered that the text in GENIA consists only of abstracts, which are tailored to the purpose of communicating a few salient points in a short passage, and tend to be composed in a somewhat formulaic manner. They are written in a very restricted register, compared to the range of registers that may be present in one issue of a newspaper – news articles, lifestyle features, opinion pieces, financial reports and letters will be delivered in very different voices. Also, some of the apparent complexity of biomedical texts is illusory, stemming from the unfamiliar vocabulary, and furthermore, a distinction must be made between syntactic and semantic complexity. Consider a phrase like *iron-sulphur cluster assembly transcription factor*, the name of a family of DNA-binding proteins, which is a semantically-complex concept expressed in a syntactically-simple form – essentially just a series of nouns.

4 Evaluating the parsers

The parsers chosen for this evaluation were those described originally in Collins (1999), Charniak (1999) and Bikel (2002). These were selected because they are up-to-date (having last been updated in 2002, 2003 and 2004 respectively), highly regarded by the computational linguistics community, and importantly, free to use and modify for academic research. Since part of our motivation was to detect signs of over-specialisation on the PTB, we assessed the current (0.9.9) and previous (0.9.8) versions of the Bikel parser individually. The current version was invoked with the new `bikel.properties` settings file, which enables parameter pruning (Bikel, 2004), whereas the previous version used the original `collins.properties` settings which were designed to emulate the Collins parser model 2 (see below). The same approach was attempted with the Charniak parser, but the latest version (released February 2005) suffered from fatal errors



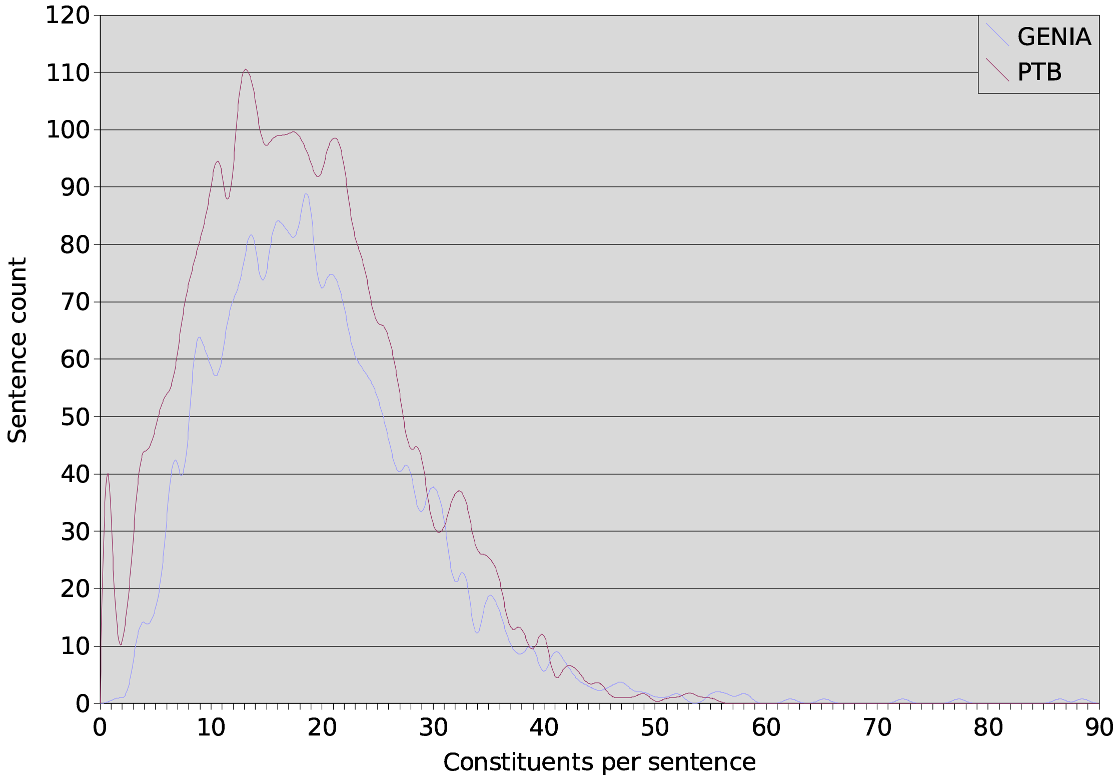


Figure 3: Sentence length and complexity distributions, GENIA vs. PTB.

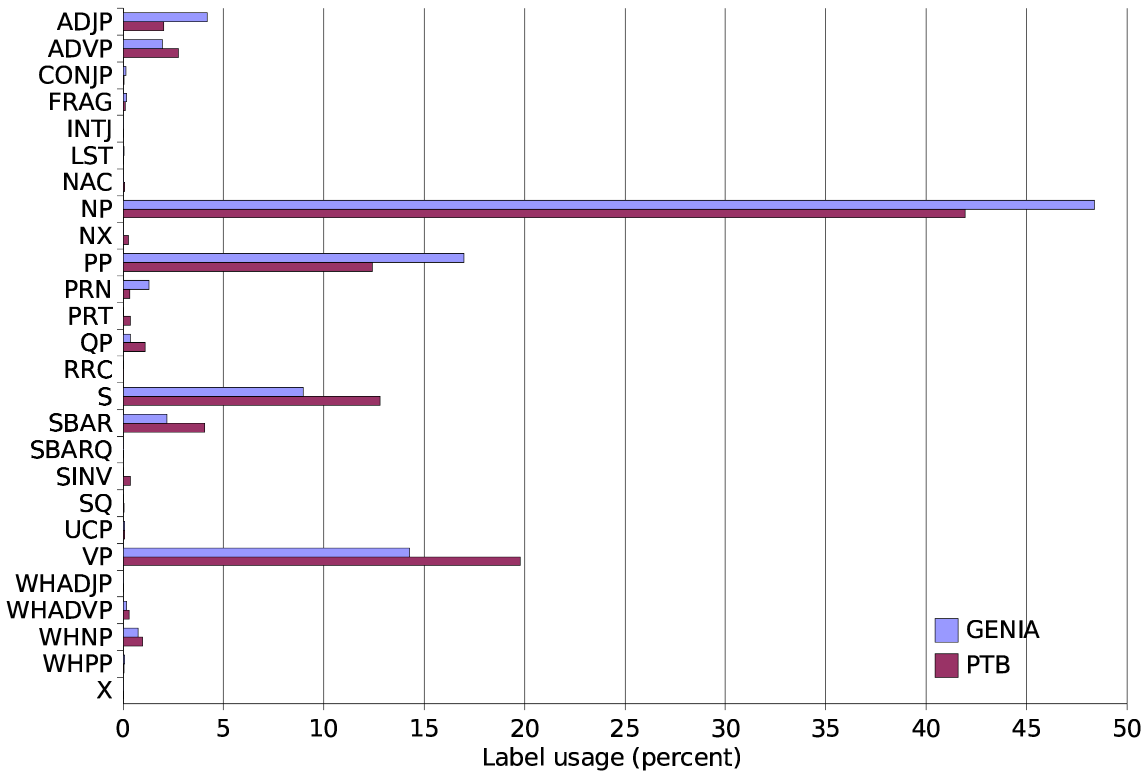


Figure 4: Constituent usages, GENIA vs. PTB.

	GENIA	PTB
Num. productions used in corpus	78831 (44.87/sent.)	96694 (40.02/sent.)
Num. distinct production rules	1364 (5.47/sentence)	2184 (5.55/sent.)

Table 1: Production and production rule usage in the two corpora.

on GENIA which could not be diagnosed in time for publication. Earlier versions of the Collins parser are not available; however, the distribution comes with three language models of increasing sophistication which were treated initially as distinct parsers.

Tweaking of parser options was kept to a minimum, aside from trivial changes to allow for unexpectedly long words, long or complex sentences (e.g. default memory/time limits), and differing standards of tokenisation and punctuation, although a considerable degree of pre- and post-processing by Perl scripts was also necessary to bring these into line. More detailed tuning would have massively increased the number of variables under consideration, given the number of compile-time constants and run-time parameters available to the programs; furthermore, it is probably safe to assume that each author distributes his software with an optimal or near-optimal configuration, at least for in-domain data.

4.1 Part-of-speech tagging

The Collins parser requires pre-tagged input, and although the Bikel parser can take untagged input, the author recommends the use of a dedicated POS tagger. For this reason, we pre-processed GENIA with MedPost (Smith et al., 2004), a specialised biomedical POS tagger that was developed and trained on MEDLINE abstracts. The supplied gold-standard POS tags were discarded as using them would not provide a realistic ap-

Top-25 production rules in GENIA (left) and PTB (right)				
Freq.	Rule	Rank	Freq.	Rule
6.36	PP → IN NP	1	4.80	PP → IN NP
3.32	NP → NN	2	2.95	S → NP VP
3.13	NP → NP PP	3	2.26	NP → NP PP
2.17	S → NP VP	4	2.15	TOP → S
2.03	TOP → S	5	1.86	NP → DT NN
1.23	NP → DT NN	6	1.43	S → VP
0.89	NP → NN NN	7	1.08	NP → PRP
0.85	NP → NP CC NP	8	0.92	ADVP → RB
0.82	S → VP	9	0.91	NP → NNP
0.76	VP → VBN PP	10	0.83	NP → NNS
0.74	ADVP → RB	11	0.81	VP → TO VP
0.70	NP → DT JJ NN	12	0.78	NP → NN
0.66	NP → NNS	13	0.74	NP → NNP NNP
0.58	NP → JJ NNS	14	0.63	SBAR → IN S
0.53	NP → JJ NN	15	0.62	NP → DT JJ NN
0.51	SBAR → IN S	16	0.60	NP → NP NP
0.51	PP → TO NP	17	0.57	SBAR → S
0.49	NP → DT NN NN	18	0.50	VP → VB NP
0.48	NP → NP PRN	19	0.48	NP → NP SBAR
0.48	ADJP → JJ	20	0.47	VP → MD VP
0.47	NP → NP PP PP	21	0.46	NP → JJ NNS
0.47	PRN → (NP)	22	0.41	SBAR → WHNP S
0.45	NP → NN NNS	23	0.40	PP → TO NP
0.44	NP → NP VP	24	0.33	VP → VBD SBAR
0.40	VP → VBD VP	25	0.32	NP → NP CC NP

Table 2: The most common production rules in the two corpora, in order, with the frequency of occurrence of each. Notice that several rules are much more common in one corpus than the other, such as VP → TO VP, which is the 11th most common rule in the PTB but doesn't make it into GENIA's list.

proximation of the kinds of scenario where parsing software would be deployed on unseen text. MedPost was found to tag GENIA with 93% accuracy.

Likewise, although the Charniak parser assigns POS tags itself and was developed and trained without exposure to a biological vocabulary, it was allowed to compete on its own terms against the other two parsers each in conjunction with MedPost. Although this may seem slightly unfair, to do otherwise would not reflect real-life usage scenarios. The parser tagged GENIA with an accuracy of 85%.

The PTB extract used was included preprocessed with the MXPOST tagger (Ratnaparkhi, 1996) as part of the Collins parser distribution; the supplied tagging scored 97% accuracy. The Charniak parser re-tagged this corpus with 96% accuracy.

4.2 Initial performance comparison

Having parsed each corpus with each parser, the output was post-processed into a standardised XML format. The same pruning operations performed on the original corpora (see Section 3) were repeated where necessary. TOP nodes (S1 nodes in the case of the Charniak parser) were removed from all files as these remain constant across every sentence. NAC and NX labels were replaced by NP labels in the parses of GENIA as the GENIA annotators use NP labels where these would occur. We then performed lineage- and

	Raw scores on GENIA (1757 sentences)					
Parser	LA score	Precision	Recall	F-measure	% perfect	% failure
Bikel 0.9.8	91.12	81.33	77.43	79.33	14.29	0.06
Bikel 0.9.9	65.30	81.68	55.75	66.27	11.21	25.04
Charniak	89.91	77.12	76.05	76.58	12.81	0.00
Collins 1	88.74	79.06	73.87	76.38	13.15	0.68
Collins 2	87.85	81.30	74.49	77.75	14.00	1.42
Collins 3	86.33	81.57	73.28	77.20	14.00	2.28
	Raw scores on PTB (2416 sentences)					
Parser	LA score	Precision	Recall	F-measure	% perfect	% failure
Bikel 0.9.8	94.45	88.09	88.13	88.11	33.44	0.04
Bikel 0.9.9	80.11	88.03	74.61	80.76	29.80	12.75
Charniak	94.36	88.09	88.28	88.18	35.06	0.00
Collins 1	94.17	86.80	86.70	86.75	31.13	0.00
Collins 2	94.36	87.29	87.20	87.24	33.49	0.04
Collins 3	94.25	87.28	87.10	87.19	33.11	0.08

Table 3: Initial performance comparison.

constituent-based scoring runs using our own Perl scripts.

The results of this experiment are summarised in Table 3, showing both the scores on both GENIA and the PTB. The LA score given is the mean of the leaf-ancestor scores for all the words in the corpus, and the precision and recall scores are taken over the entire set of constituents in the corpus. Initially, these measures were calculated per sentence, and then averaged across each corpus, but the presence of pathologically short sentences such as *Energy.* gives an unrepresentative boost to per-sentence averages. (Interestingly, many published papers do not make clear whether the results they present are per-sentence averages or corpus-wide scores.)

'Mean X' is simply the average number of crossing brackets per sentence. 'F-measure' (van Rijsbergen, 1979) is the harmonic mean of precision and recall; it is a balanced score that penalises algorithms which favour one to the detriment of the other, and is calculated as follows:

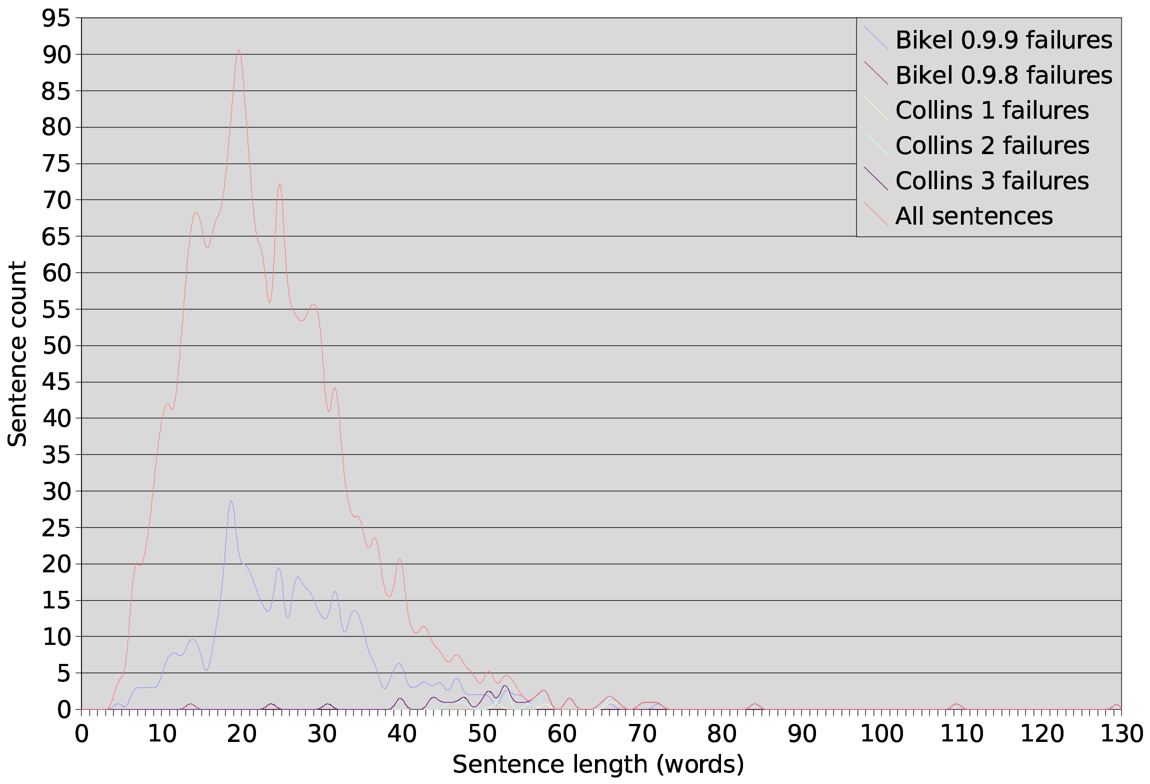
$$F = \frac{2 \times P \times R}{P + R}$$

4.3 Parse failures

Since most of the parsers suffered from a considerable number of parse failures in GENIA – sentences where no parse could be obtained – Table 4 shows recalculated scores based on evaluation of successfully-parsed sentences only. Conflating the performance drops caused by poorly parsed sentences with those caused by total failures gives an inaccurate picture of parser behaviour. In order to determine if there was any pattern to these failures, we plotted the number of parse failures for each parser against sentence length and sentence complexity (see Figure 5). These charts revealed some interesting trends.

There is a known problem with the Collins models 2 and 3 failing on two sentences in the PTB section 23 due to complexity, but this problem is exacerbated in GENIA, with even the simpler model 1 failing on a number of sentences, one of which was only 24 words long plus punctuation.

Overall, however, the failures do tend to cluster around the right-hand tails of the sentence length and constituent count distributions. Discounting such sentences, the three models do show a consistent monotonic increase in precision, recall and LA score from the simplest to the most complex, accompanied by a decrease in the number of crossing brackets per sentence. Interestingly, these intervals are much more pronounced on GENIA than on the PTB, where the performance seems to level off between models 2 and 3. Difficult sentences aside, then, it appears that the advanced fea-



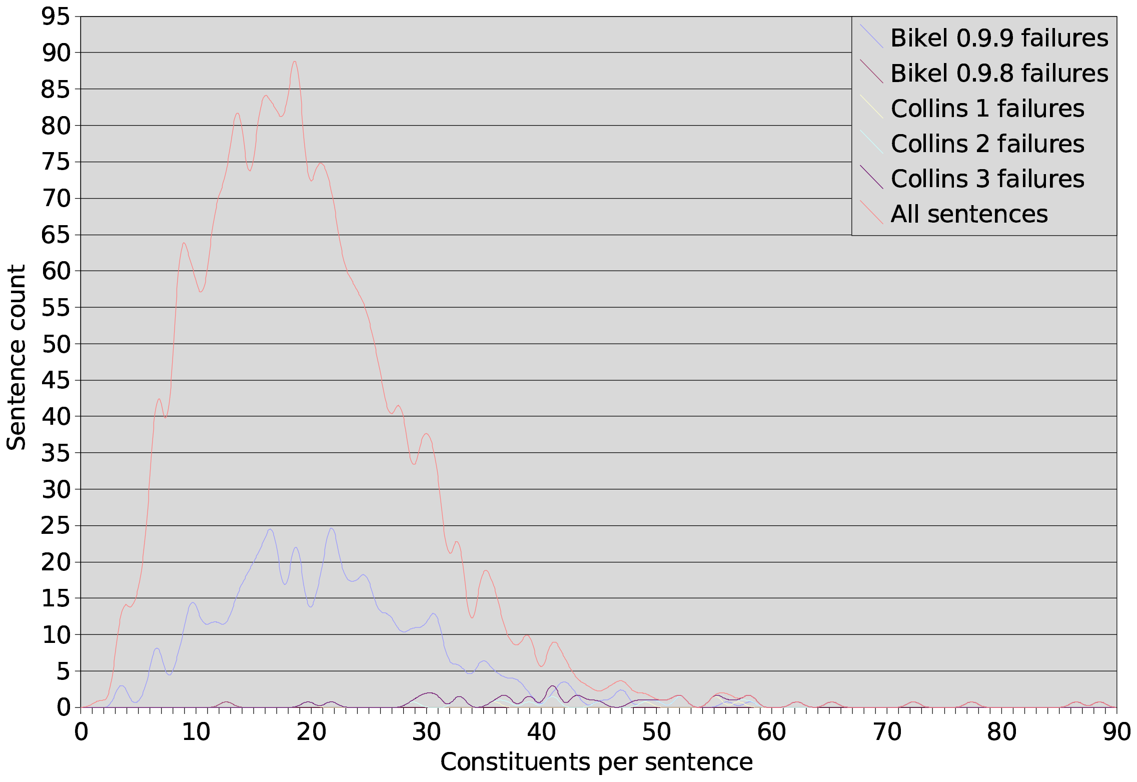


Figure 5: Parse failures on GENIA vs. sentence length and complexity for each parser.

	Scores on GENIA, successfully-parsed sentences only					
Parser	LA score	Precision	Recall	F-measure	Mean X	# parsed
Bikel 0.9.8	91.15	81.33	77.46	79.35	2.06	1756
Bikel 0.9.9	91.17	81.68	77.04	79.29	1.89	1317
Charniak	89.91	77.12	76.05	76.58	2.42	1757
Collins 1	90.53	79.06	75.35	77.16	2.29	1745
Collins 2	91.21	81.30	77.24	79.22	2.01	1732
Collins 3	91.32	81.57	77.42	79.44	1.95	1717
	Scores on PTB, successfully-parsed sentences only					
Parser	LA score	Precision	Recall	F-measure	Mean X	# parsed
Bikel 0.9.8	94.53	88.09	88.20	88.15	1.07	2415
Bikel 0.9.9	94.52	88.03	88.07	88.05	1.04	2108
Charniak	94.36	88.09	88.28	88.18	1.08	2416
Collins 1	94.17	86.80	86.70	86.75	1.23	2416
Collins 2	94.45	87.29	87.28	87.28	1.19	2415
Collins 3	94.44	87.28	87.27	87.28	1.18	2414

Table 4: Performance scores, discounting all parse failures. Scores for the Charniak parser, and Collins model 1 on the PTB, are shown again for comparison, although they did not fail on any sentences.

tures of models 2 and 3 are actually more valuable on this unfamiliar corpus than on the original development domain – provided that they do not trip the parser up completely.

While Bikel 0.9.8's failures are relatively few and tend to occur more often in longer and more complex sentences, like those of the Collins models, the distributions in Figure 5 for Bikel 0.9.9 follow the shapes of the distributions remarkably accurately. In other words, the length or complexity of a sentence does not seem to be a major influence on the ability of Bikel 0.9.9 to parse it. Undoubtedly, there is something more subtle in the composition of these sentences that confuses Bikel's updated algorithm, although we could not discern any pattern by eye. Perhaps this problem could be diagnosed by monitoring the parser in a Java debugger or modifying it to produce more verbose output, but such an examination is beyond the scope of this work.

Although version 0.9.9 fails on far fewer sentences in the PTB than in GENIA, it still suffers from two orders of magnitude more failures than any other parser on the same corpus. These results suggest that the author's claim that parameter pruning results in "no loss of accuracy" (Bikel, 2004) can only be taken seriously when the test set has been cleaned of all unparseable sentences; this impression is reinforced by the fact that the precision and recall scores reported by the author agree quite closely with our results on the PTB once the parse failures have been removed.

5 Combining the parsers

Given the poorer results of these parsers on GENIA than on the PTB, and the comparative lack of annotated data in this domain, it is important to consider ways in which performance can be enhanced without recourse to supervised training methods. Various experimental techniques exist for reducing or eliminating the need for labelled training data, particularly in the presence of several diverse parsers (or more generally, classifiers). These include active learning (Osborne and Baldridge, 2004), bagging and boosting (Henderson, 1999) and co-training (Steedman et al., 2003). In addition to these 'knowledge-poor' techniques, one can easily imagine domain-specific 'knowledge-rich' techniques that employ existing biological data sources and NLP methods in order to select, modify, or constrain parses (see Section 7.2). For this preliminary investigation, however, we concentrated on knowledge-poor methods originating in work on parsing the

PTB which could exploit the availability of multiple parsers whilst requiring no time-consuming retraining processes or integration with external resources. Perl implementations of the algorithms discussed below can be downloaded from our website.

5.1 Fallback cascades

In the Collins parser instructions, the author suggests stacking the three models in decreasing order of sophistication ($3 \rightarrow 2 \rightarrow 1$), and for each sentence, falling back to the next less sophisticated model each time a more sophisticated one fails to obtain a parse. The principle behind this is that the more complex a model is, the more often it will fail, but the better the results will be when it does return a parse. We implemented this system for the Collins models, and also for the Bikel parser, starting with version 0.9.9 and falling back to 0.9.8 on failure. Since the Charniak parser did not suffer any failures, we added it to each of these cascades as a last-resort level, to fill any remaining gaps.

As expected, the results for each cascade (Table 5) were comparable to their component parsers' scores on successfully-parsed sentences (Table 4), except with 100% coverage of the corpus. In each of the following parser integration methods, we used these fallback cascades to represent the Bikel and Collins parsers, rather than any of their individual parser models. The Bikel cascade was used as a baseline against which to test the results of each method for statistical significance, using a two-tailed dependent t-test over paired scores.

5.2 Constituent voting

Henderson (1999) reports good results when using a simple parse integration method called constituent voting, where a hybrid parse is produced by taking votes from all the parsers in an ensemble. Essentially, all the constituents proposed by the parsers are pooled, and each one is added to the hybrid parse if more than half of the parsers in the ensemble agree on it. The assumption behind this concept is that the mistakes made by the parsers are reasonably independently distributed – if different kinds of errors beset the parsers, and at different times, then a majority vote will tend to converge on the correct set of constituents.

We implemented a three-way majority vote ensemble between the Collins and Bikel cascades and the Charniak parser; the results are shown in Table 6. The most notable gain was in precision, as one would hope from an algorithm designed to screen out minority parsing decisions, but the scores also illustrate an interesting phenomenon. Although the ensemble took the lead on all the constituent-based performance indicators, it performed poorly on LA score. This demonstrates an important point about parser scoring metrics – that an algorithm designed to boost one measure of quality can do so without necessarily raising performance according to a different yardstick.

Part of the reason for this discrepancy may be a quirk of the constituent voting algorithm that constituent-based precision and recall scores gloss over. The trees it produces are not guaranteed to be well-formed under the grammars of any of the members of the ensemble; if, for example, the parsers cannot reach consensus about the exact boundaries of a verb phrase, a sentence without a VP constituent will be produced, leading to some unusual attachments at a higher level. Unlike the constituent-based approach, LA scoring tends to favour parses that are accurate at the upper levels of the tree, so an increase in precision and recall without a corresponding increase in LA score would be consistent with this kind of oddity.

5.3 Parse selection

An alternative approach to integrating the outputs of a parser ensemble is whole parse selection on a per-sentence basis, which has the potential added advantage over hybridisation methods like constituent voting that the gaps in trees described above cannot occur. The most obvious way to guess the best candidate parse for a sentence is to assume that the true parse lies close to the centroid of all the candidates in parse space, and then, using some similarity or distance measure between the candidates, pick the candidate that is most similar to (or least distant from) all the other parses.

We implemented three parse switchers, two based on constituent overlap, and one based on lineage similarity between pairs of parses. Similarity and distance switching (Henderson, 1999) take the

	Ensemble scores on GENIA, all sentences parsed successfully					
Ensemble	LA score	Precision	Recall	F-measure	Mean X	% perfect
Collins-Charniak fallback	90.74	80.51	76.44	78.42	2.16	14.00
Bikel-Charniak fallback	91.08	81.31	76.96	79.08	2.05	14.11

Table 5: Ensemble scores on GENIA for Collins(3, 2, 1)→Charniak and Bikel(0.9.9, 0.9.8)→Charniak fallback cascades.

	Ensemble scores on GENIA, all sentences parsed successfully					
Algorithm	LA score	Precision	Recall	F-measure	Mean X	% perfect
Majority vote ensemble	90.21	83.41	77.50	80.35	1.71	14.68

Table 6: Ensemble scores on GENIA for parse combination by majority constituent voting.

number of constituents that each parse has in common, and the number that are proposed by either one but not both parsers, as measures of similarity and distance respectively. Levenshtein switching, a novel method, uses the sentence-mean LA scores between parses as the similarity measure. In all methods, the parse with the maximum total pairwise similarity (minimum total pairwise distance) to the set of rival parses for a sentence is chosen. In no case were POS tags taken into account when calculating similarity, as they would have made the Collins and Bikel parsers artificially similar.

The results of these experiments are shown in Table 7. All three methods achieved comparable improvements overall, with the similarity and distance switching routines favouring recall and precision respectively (both differences significant at $p < 0.0001$). Note however that the winning LA score for Levenshtein switching is not a statistically significant improvement over the other switching methods.

6 Error analysis

All of the parser integration methods discussed above make the assumption that the parsers in an ensemble will suffer from independently-distributed errors, to a greater or lesser extent. Simple fallback cascades rely on their individual members failing on different sentences, but the more sophisticated methods in Section 5.2 and Section 5.3 are all ultimately based on the principle that agreement between parsers indicates convergence on the true parse. Although the performance gains we achieved with such methods are statistically significant, they are nonetheless somewhat unimpressive compared to the 30% reduction of recall errors and 6% reduction of precision errors reported for the best ensemble techniques in Henderson and Brill (1999) on the PTB.

This led us to suspect that the parsers in the ensembles were making similar kinds of errors on GENIA, perhaps not across the board, but certainly often enough that consensus methods pick incorrect constituents, and centroid methods converge on incorrect parses, with a significant frequency. To investigate this phenomenon, and more generally to tease apart the reasons for each parser's performance drop on GENIA, we measured the precision and recall for each parser on each production rule over GENIA and the PTB. We then gathered the 25 most common production rules in GENIA and compared the scores achieved by each parser on each rule to the same rule in PTB, thus drawing attention to parser-specific issues and more widespread systematic errors. We also collected closest-match data on each missed production in GENIA, for each parser, and calculated substitution frequencies for each production rule. This enabled us to identify both the sources of performance problems, and to a certain extent their causes and connotations. These data tables have been omitted for space reasons, since the discussion below covers the important lessons learnt from them, but they are available as supplementary materials on our website.

	Ensemble scores on GENIA, all sentences parsed successfully					
Algorithm	LA score	Precision	Recall	F-measure	Mean X	% perfect
Similarity switching	91.34	81.73	78.01	79.83	1.97	14.85
Distance switching	91.35	82.10	77.72	79.85	1.92	15.08
Levenshtein switching	91.39	81.83	77.51	79.61	1.95	14.74

Table 7: Ensemble scores on GENIA for parse selection by three centroid-distance algorithms

6.1 Bikel parser errors

Despite the similar overall LA score and F-measure for the two versions on parseable sentences only, there are signs that the differences between them run deeper than failure rates. The newer version's higher precision and lower crossing brackets per sentences, along with lower recall, indicates that it is generating slightly more conservatively than the older version, on GENIA at least; these scores are much closer on the PTB. Also, the production rule scores show one unexpected phenomenon – the older version is actually considerably better at labelling noun phrases of the form `( NP )` as parenthetical expressions in GENIA ($F = 81.07$) than in PTB ($F = 61.29$), as are the Collins and Charniak parsers, while the newer version is much worse at this task in GENIA ($F = 42.36$). On closer inspection, however, 84% of the occurrences of `PRN` $\rightarrow$ `( NP )` mislabelled by the newer version are instead marked as `PRN` $\rightarrow$ `( NN )` productions of the same width – in other words, an intermediate `NP` constituent covering just a single noun has in these cases been removed, and the noun 'promoted' to a direct daughter of the `PRN` constituent. Although this demonstrates a difference in the modelling of noun phrases between the two versions, it is unlikely that such a difference would alter the meaning of a sentence. Furthermore, it must be noted that `PRN` $\rightarrow$ `( NP )` is much more common in GENIA than in PTB, so the improvements achieved by the other parsers may be partially accidental; even if they simply assumed that every phrase in parentheses is a noun phrase, they would do better on this production in GENIA as a result.

6.2 Charniak parser errors

The Charniak parser goes from state-of-the-art on PTB to comparatively poor on GENIA. It ranks lowest in both LA score and F-measure when only successfully parsed sentences are taken into account, and still only achieves mediocre performance when the other parsers' scores cover failed sentences too, despite not failing on any sentences itself. This discrepancy can be explained by a lack of biomedical vocabulary available to its built-in POS tagger. Although it tags GENIA with an accuracy of 85% across all word classes, it achieves only 63% on the `NN` (singular/mass noun) class. This is the most numerous single class in GENIA, and that which many domain-specific single-word terms and components of multi-word phrases belong to.

The knock-on syntactic effects of this disability can be traced in the leaf-ancestor metrics, where the parser scores an impressive mean of 91.50 for correctly-tagged words, compared to just 80.71 for incorrectly-tagged words. A similar effect can be seen in the statistics for productions with `NN` tags on the right hand side. `NP` $\rightarrow$ `NN` and `NP` $\rightarrow$ `NN NN` are identified with respective recalls of only 33% and 19% in GENIA, for example, as opposed to 90% and 82% in the PTB. More than 40% of mislabelled `NP` $\rightarrow$ `NN` productions in GENIA were identified instead as `NP` $\rightarrow$ `NNP` (proper noun) or `NP` $\rightarrow$ `NNS` (plural noun) productions by the parser, and the implications of these mistakes for information extraction tasks do not seem great, especially since the majority of single-word noun phrases of particular interest in this domain are likely to be genes, proteins etc. that can be tagged independently by a dedicated named-entity recognizer. The story is different for mislabelled `NP` $\rightarrow$ `NN NN` productions, where 29% are mistaken for `NP` $\rightarrow$ `JJ NN` productions, a substitution that one can imagine causing greater semantic confusion. On the other hand, the Charniak parser goes from being the worst at identifying ADVP $\longrightarrow$ RB productions (single-word adverb phrases) on the PTB ($F = 87.68$) to being the best at this task on GENIA ($F = 87.06$).

Accuracy issues notwithstanding, Charniak's is still the most robust of all the parsers, failing on none of the supplied sentences in either corpus. This may reflect a strategy of 'making-do' when an exact parse cannot be derived; it deployed more general-purposes FRAG (fragment) and X (unknown/unparseable) constituents combined than any other parser, and even a handful of INTJ (interjection) phrases that no other parser used in GENIA. Of course, such productions are not always correct – there are actually no interjections in GENIA – but from an information extraction point of view, a rough parse may be better than no parse at all, especially if the inclusion of such inexact labels can be reflected in a reduced level of confidence or trustworthiness for the sentence.

6.3 Collins parser errors

We noted in Section 4.2 that the Collins models achieved successively better performance on GENIA once parse failures were discounted. Considering individual production rules, however, the trends are not so clear-cut. There are rules that do follow this overall pattern, such as S $\longrightarrow$ NP VP, which model 3 actually assigns more effectively on GENIA ($F = 88.98$) than it does on the PTB ($F = 88.79$). However, there are several common productions where model 3's accuracy degrades more than model 2's, most of which begin with NP $\longrightarrow$ .... Most of these are found in the PTB more effectively by model 3 than model 2, which suggests over-fitting; these specific increases in performance have apparently come at the expense of portability. Note, however, the caveat regarding noun phrases below.

6.4 Common trends

In addition to these parser-specific observations, there are various phenomena that are common to all or most of the parsers. Due to slight differences in the annotation of co-ordinated structures between the GENIA and PTB guidelines, the correct generation of noun-phrase conjunctions (NP $\longrightarrow$ NP CC NP) proved much harder on GENIA, with all parsers having problems with identification of the boundaries of the conjunction in the text, and often with correct labelling of the constituents involved too. Cases where each NP is a single word were handled relatively well, with the essentially equivalent NN CC NN construction often being proposed instead, but more complex cases caused widespread difficulty.

More surprisingly, the labelling of single-word noun and adjective phrases (NP $\longrightarrow$ NN and AJDP $\longrightarrow$ JJ), both of which are significantly more frequent in GENIA, seemed challenging across the board. The most commonly-occurring error involved subsumption by a wider constituent with the same label, apart from the errors of vocabulary for the Charniak parser as described above. However, for correctly-tagged adjectives and nouns, there are many situations where this will not make any difference to the sense of a sentence. For example, in a production like NP $\longrightarrow$ DT JJ NN, the adjective still has its modificatory effect on the noun without needing to be placed within an ADJP phrase of its own, and the noun is entirely capable of acting as the head of the phrase without being nested within an NP sub-phrase.

Similar effects occurred frequently with longer phrases containing nouns, such as NP $\longrightarrow$ DT NN or NP $\longrightarrow$ NN NN, where the most common errors were also subsumptions by wider noun phrases. Although the GENIA annotators warn that "when noun phrase consists with sequence of nouns *[sic]*, the internal structure is not neccessarily shown,"[6] which must account for some of the noun handling problems, such subsumptions suggest that the opposite may be occurring too – that there are cases where the parsers are failing to generate internal structure within noun phrases.

Prepositions were involved in many of the problematic cases, both on the left-hand and right-hand sides of productions, and understandably so. The disambiguation of prepositional attachment is a continuing problem in parser design, and methods that take into account lexical dependencies between head words will be less effective when the words in question are out-of-domain and thus un-

[6]http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/topics/Corpus/manual-for-bracketing.html

seen in training. The production `NP` $\rightarrow$ `NP PP PP` is a good example. The most common error for all parsers was to produce `NP` $\rightarrow$ `NP PP` at the same span of words in the sentence, indicating that one of the prepositional phrases is frequently attached at the wrong level. The second most common error was to substitute a shorter `NP`, suggesting that one or both of the `PP`s were excluded. Such errors are potentially of more serious semantic importance than differences of opinion about how to mark up the internal structure of noun-only phrases.

7 Discussion

Although the performance gains achieved by our parser integration methods are statistically significant, and illustrative of some important points about parser behaviour and syntactic evaluation methodologies, it is doubtful that the results are good enough to justify deploying these techniques on large amounts of text, at least in their current form. The small increases in accuracy are probably outweighed by the additional computational costs. The fallback cascades provided the same protection from parse failure, with better performance than the widest-coverage parser alone, and in a production system most sentences would only need to be parsed by the first parser in the cascade.

However, the parser integration idea as a whole is not without its merits; we determined that an oracle picking the best sentences on GENIA would achieve an LA score of 93.56, so there is still room for improvement if the algorithms can be made smarter. Although our analysis of the parsers' mistakes on GENIA indicated that the ideal of independently-distributed errors which underpins these integration methods does not hold true, the very fact that we can analyse their behaviour patterns in such detail suggests that a sufficiently well-designed ensemble could in principle learn the circumstances under which each parser could be trusted on a given corpus.

Furthermore, there are additional ways in which an ensemble might assist with practical NLP issues. While analysing the data from the parse selection algorithms, we discovered that the centroid distances for the *winning* parses, scaled by sentence size where necessary, correlate fairly well ($|r| \approx 0.5$) with those parses' true LA scores and F-measures (see Section 7.2). We developed a similar measurement for constituent voting based on the level of consensus among the ensemble members – the number of constituents winning a majority vote divided by the number of distinct candidate constituents – which showed a comparable degree of correlation. This provides an answer to our initial question about the extent to which parser agreement signals parse correctness. Presumably the major limiting factor on these correlations is the presence of widespread systematic errors like those described in Section 6.4.

7.1 Engineering issues

As noted previously, the Charniak parser takes raw text and performs tokenisation and POS tagging internally. While this may seem like an advantageous convenience, in practice it is the source of considerable extra work, besides being the cause of avoidable parse errors. The tokenisation standards encoded by Charniak did not match those assumed by either the GENIA corpus, or indeed the PTB extract, although problems were much more widespread in the GENIA. Words containing embedded punctuation were frequently split into multiple tokens, so these word-internal symbols had to be converted into textual placeholders before parsing and converted back afterwards. The somewhat idiosyncratic conventions of the GENIA corpus did not help (*differentiation/activation* being tagged as one token for example) but the fact that similar issues occurred on the newspaper corpus (e.g. with *US$* or *81-year-old*) suggests that making assumptions about the 'correct' way to tokenise text is a bad policy in any domain.

Even when working on in-domain data, it seems like a bad design decision to assume the parser will be able to match the performance of a state-of-the-art POS tagger on unseen text. The Bikel parser can operate in either mode, which is a much more flexible policy. In all fairness, however, it would probably be fairly trivial for an interested C++ developer to bypass the Charniak parser's tokeniser and tagger and recompile it.

A different kind of engineering issue is that of computation time. Parsing is a slow process in any case, and ensemble methods compound this problem. However, parsing is a canonically easy task to perform in parallel, since (at this level of understanding at least) each sentence has no dependencies on the previous, so even the parse integration step can be split across multiple pipelines. We intend to run pilot studies on the scalability of parallel distributed parsing techniques, both on an IBM Blade Center cluster running Beowulf Linux, and a heterogeneous network of Windows PCs in their spare hours, in order to determine the feasibility and comparative attractiveness of each approach.

7.2 Future work

We mentioned above that there is a significant correlation between the distance of a winning parse from the centroid and its accuracy compared to the gold standard. In an IE or other text mining scenario, one could use these values as estimators of trustworthiness for each parse – empirical measures indicating how reliable to consider it. We would like to explore this idea further, as it can provide an extra level of understanding which is missing from many IE techniques, and could be easily employed in the ranking of extracted 'facts' or the resolution of contradictions. Since LA scores can be calculated per word or per any arbitrary region of the sentence, the potential even for rating the trustworthiness of different clauses or relationships individually cannot be ignored.

We have been experimenting with training a neural network to pick the best parse for a sentence based only on the pairwise Levenshtein distances between the candidate parses, in the hope that it can learn what decision to make based on patterns of agreement between the parsers, rather than just picking the parse which is most similar to all the others as the current methods do. So far, however, it has been unable to exceed the performance of the unsupervised methods, which suggests that additional features may need to be considered.

A complementary approach to parse integration would be to merge the PTB with an annotated biomedical corpus and retrain a parser from scratch. This proposition appears more attractive in the light of the production rule frequency differences between GENIA and the PTB, and will become more effective as the GENIA treebank grows and the Mining the Bibliome project begins posting official releases. In the meantime, we have begun investigating the potential for using biological named-entity and ontological-class information to help rule out unlikely parses, for example in cases where an entity name is bisected by a constituent boundary.

Acknowledgements

This work was funded by the Biotechnology and Biological Sciences Research Council and AstraZeneca PLC. We would like to thank the reviewers for their insights and suggestions, and Mark McLauchlan for his help with prepositions.

References

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the Human Language Technology Conference 2002 (HLT2002)*. San Diego.

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*. ACL, Barcelona.

Eugene Charniak. 1999. A maximum-entropy-inspired parser. Technical report, Brown University.

Kevin B. Cohen and Lawrence Hunter. 2004. Natural language processing and systems biology. In Werner Dubitzky and Francisco Azuaje, editors, *Artificial Intelligence Methods and Tools for Systems Biology*. Springer Verlag, Heidelberg.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD, University of Pennsylvania.

Richard Crouch, Ronald Kaplan, Tracy King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad coverage parser. In *Proceedings of the LREC-2002 workshop "Beyond PARSEVAL: Towards Improved Evaluation Measures for Parsing Systems"*. Las Palmas, Spain.

Carol Friedman, Pauline Kra, and Andrey Rzhetsky. 2002. Two biomedical sublanguages: a description based on the theories of Zellig Harris. *Journal of Biomedical Informatics*, 35(4):222–235.

John C. Henderson. 1999. *Exploiting Diversity for Natural Language Parsing*. PhD, Johns Hopkins University.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing (EMNLP99)*. University of Maryland.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182.

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In Lynette Hirschman and James Pustejovsky, editors, *HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, pages 61–68. Association for Computational Linguistics, Boston, Massachusetts, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 89–96. Association for Computational Linguistics, Boston, Massachusetts, USA.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*. University of Pennsylvania.

Eric Ringger, Robert C. Moore, Eugene Charniak, Lucy Vanderwende, and Hisami Suzuki. 2004. Using the Penn Treebank to evaluate non-treebank parsers. In *Proceedings of 4th International Conference on Language Resource and Evaluation (LREC2004)*, volume IV. ELDA.

Brian Roark. 2002. Evaluating parser accuracy using edit distance. In *Proceedings of the LREC-2002 workshop "Beyond PARSEVAL: Towards Improved Evaluation Measures for Parsing Systems"*. Las Palmas, Spain.

Geoffrey Sampson and Anna Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, 9(4):365–380.

Hagit Shatkay and Ronen Feldman. 2003. Mining the biomedical literature in the genomic era: An overview. *Journal of Computational Biology*, 10(6):821–856.

Larry H. Smith, Thomas Rindflesch, and W. John Wilbur. 2004. MedPost: a part-of-speech tagger for biomedical text. *Bioinformatics*, 20(14):2320–2321.

Mark Steedman, Steven Baker, Stephen Clark, Jay Crim, Julia Hockenmaier, Rebecca Hwa, Miles Osborne, Paul Ruhlen, and Anoop Sarkar. 2003. CLSP WS-02 final report: Semi-supervised training for statistical parsing. Technical report, Johns Hopkins University.

Cornelis J. van Rijsbergen. 1979. *Information Retrieval, 2nd edition*. Butterworths, London.

Juan Xiao, Jian Su, GuoDong Zhou, and ChewLim Tan. 2005. Protein-protein interaction: A supervised learning approach. In *Proceedings of the First International Symposium on Semantic Mining in Biomedicine*. Hinxton, UK.