Research on Language and Computation **00:** 2–44, 2011.
© 2011 *Kluwer Academic Publishers. Printed in the Netherlands.*

Cross-Domain Effects on Parse Selection for Precision Grammars

ANDREW MACKINLAY (amack@csse.unimelb.edu.au)
University of Melbourne/NICTA

REBECCA DRIDAN (rdridan@csse.unimelb.edu.au)
University of Melbourne/NICTA

DAN FLICKINGER (danf@stanford.edu)
Stanford University

TIMOTHY BALDWIN (tb@ldwin.net)
University of Melbourne/NICTA

Received: . . . . . . ; Accepted: . . . . . .

Abstract.

We examine the impact of domain on parse selection accuracy, in the context of precision HPSG parsing using the English Resource Grammar, using two training corpora and four test corpora and evaluating using exact tree matches as well as dependency F-scores. In addition to determining the relative impact of in- vs. cross-domain parse selection training on parser performance, we propose strategies to avoid cross-domain performance penalty when limited in-domain data is available. Our work supports previous research showing that in-domain training data significantly improves parse selection accuracy, and that it provides greater parser accuracy than an out-of-domain training corpus of the same size, but we verify experimentally that this holds for a handcrafted grammar, observing a 10–16% improvement in exact match and 5–6% improvement in dependency F-score by using a domain-matched training corpus. We also find it is possible to considerably improve parse selection accuracy through construction of even small-scale in-domain treebanks, and learning of parse selection models over in-domain and out-of-domain data. Naively adding an 11000-token in-domain training corpus boosts dependency F-score by 2–3% over using solely out-of-domain data. We investigate more sophisticated strategies for combining data from these sources to train models: weighted linear interpolation between the single-domain models, and training a model from the combined data, optionally duplicating the smaller corpus to give it a higher weighting. The most successful strategy is training a monolithic model after duplicating the smaller corpus, which gives an improvement over a range of weightings, but we also show that the optimal value for these parameters can be estimated on a case-by-case basis using a cross-validation strategy. This domain-tuning strategy provides a further performance improvement of up to 2.3% for exact match and 0.9% for dependency F-score compared to the naive combination strategy using the same data.

1. Introduction

In recent years there has been a growing awareness of the domain-brittleness of parsers (Gildea, 2001), and a variety of methods for adapting parsers to new domains have been developed (Roark and Bacchiani, 2003; Blitzer et al., 2006; Hara et al., 2007; Rimell and Clark, 2009; Finkel and Manning, 2009). Much of this previous work has looked at parsing with treebank-derived grammars, and it is not clear how applicable the results are to the scenario of parsing with hand-crafted precision grammars. This paper explores domain adaptation in the context of parse selection for HPSG-based precision grammars, based on the English Resource Grammar (Flickinger, 2000).

Parsing with precision grammars is generally a two-stage process: (1) the full parse yield of the precision grammar is calculated for a given item, often in the form of a packed forest for efficiency (Oepen and Carroll, 2000; Zhang et al., 2007); and (2) the individual analyses in the parse forest are ranked using a discriminative statistical model ("parse selection"). In the domain of treebank parsing, the Charniak and Johnson (2005) reranking parser adopts an analogous strategy, except that ranking and pruning are incorporated into the first stage, and the second stage is based on only the top-ranked parses from the first stage. Some differences in this precision parsing process suggest that domain-specificity could be a less pressing issue than has been seen in previous work. For one thing, a hand-written precision grammar based on linguistic theory will often encode more generally applicable facts about language than what can be learnt from a specific treebank. Furthermore, McClosky et al. (2006) show that their discriminative re-ranker is less susceptible to domain effects than their generative parser, and hence we might expect the precision grammar parsing, which uses a discriminative model, to show less domain bias. However, neither of these suppositions have been tested in any systematic way, and so one of the goals of this work is to explore the scope of domain effects in an HPSG precision grammar framework.

Another aspect of this work that differs from previous domain adaptation work is that our interest in parse selection accuracy is two-fold. Like previous work, we are interested in the top-ranked parse being as accurate as possible, but we are also concerned with how parse ranking feeds into the treebanking process. The Redwoods treebanks associated with the HPSG-based English Resource Grammar, as described in Section 2.1, are built by parsing sentences, and then having an annotator select the correct tree from amongst the n-best analyses. Annotation efficiency can be improved by using a lower n, which is only possible if the parse selection is accurate enough to reliably rank the correct parse within that top n. Hence, in order to build new treebanks in new domains, we need a statistical model that works in different domains, or a method to easily adapt a model for the new domain, without requiring a large treebank for the particular domain.

The primary contributions of this paper are: (a) exploration of the relative effects of domain on parse selection accuracy for a precision grammar of HPSG; (b) investigation of the most effective and robust methods for using small amounts of in-domain training data. This has been made possible by the recent creation of multiple ERG treebanks of sufficient size, and spanning complex and variable domains enabling us to explore these issues systematically for the first time for precision HPSG parsing. We additionally introduce a methodology for quantitatively and qualitatively comparing different corpora for lexico-syntactic divergences.

2. Background

2.1. REDWOODS-STYLE TREEBANKING

The annotated corpora used in this study form the Redwoods treebank (Oepen et al., 2004), which has been updated for the October 2010 version of the English Resource Grammar (ERG: Copestake and Flickinger (2000), Flickinger (2000), Flickinger (2011)). In brief, the treebank is constructed by parsing each sentence in the corpus using the ERG, automatically identifying (up to) the 500 most likely parses, then manually identifying the correct parse using *discriminants* (Carter, 1997; Oepen et al., 2002) to allow or disallow constructions postulated by the grammar for a particular sentence. The discriminants used in the Redwoods treebank consist of the 200 unary and binary syntactic and lexical rules of the ERG, along with the lexical entries themselves, which together form the relevant sources of ambiguity in each parse forest. When a new version of the grammar is released, the recorded discriminant-level annotations enable semi-automatic updating of the treebank, requiring manual intervention only for new sources of ambiguity introduced by changes to the grammar. Parse ranking is determined using a maximum entropy model trained on the previous version of the treebank. Similar discriminant-based treebank annotation has also been employed in the development of the Alpino treebank for Dutch (Bouma et al., 2001), and in treebanks built using the LFG Parsebanker (Rosén et al., 2009).

This grammar-centric method of treebank construction differs in a couple of fundamental aspects to that used for other widely used treebanks such as the Penn Treebank for English (Marcus et al., 1993), or the TIGER treebank for German (Brants et al., 2002). The dynamic nature of the treebank means that as linguistic analysis improves and matures, the treebank can be updated to reflect modern analyses, without expensive full re-annotation. Furthermore, the treebanking process supplies a wealth of negative training data (from rejected alternative analyses licensed by the grammar) which is used to build the discriminative parse selection models.

2.2. DOMAIN ADAPTATION FOR PARSE SELECTION

It is relatively easy to motivate the need for domain adaptation: if we wish to utilise the outputs of a given parser in some application, it is often the case that our target domain differs from that for which the parser was originally developed, leading to a higher error rate in the target domain.

English-language parsers are often trained on the Penn Treebank, but unsurprisingly the domain of financial newswire text is not appropriate for many NLP applications. Gildea (2001) found that training a parser on the WSJ corpus rather than the Brown corpus resulted in significantly worse performance over Brown corpus test data, reporting a 3.5% drop[⋆] in F-score over labelled constituents[⋆⋆] from 84.1%. Gildea uses Model 1 of Collins (1997) to measure changes in parser performance, but other work finds similar penalties with alternative parsers and domains as described below.

Some work goes further, also investigating strategies for avoiding these performance penalties when moving across domains. Roark and Bacchiani (2003) show that using a technique known as maximum *a posteriori* estimation on the productions in a probabilistic context-free grammar, it is possible to make more efficient use of in-domain and out-of-domain training data, giving labelled constituent F-score improvements of up to 2.5% over using only in-domain data when the amount of in-domain training is very limited (from a baseline of 80.5%), arguing that the conclusion of Gildea (2001) that out-of-domain data has very little value, was premature. Honnibal et al. (2009) found that the C&C parser (Clark and Curran, 2007b) trained on WSJ text gives a 4.3% lower F-score (based on CCG dependencies) when tested on Wikipedia data compared to held-out WSJ data (which had an F-score of 85.1%), but that self-training the super-tagging component on parsed in-domain data reduced this penalty to 3.8%.

Plank and van Noord (2008) investigate domain adaptation of a parser trained on the Alpino Dutch treebank (Van der Beek et al., 2002) using *auxiliary distributions*, by augmenting the model trained from a small quantity of in-domain data, with a real-valued feature which takes the value of the negative logarithm of the conditional probability of the sentence according to the larger out-of-domain model. This approach achieves performance between 1% worse and 4% better than a model trained by simply combining the in-domain and out-of-domain data, improving the performance over a purely in-domain model by up to 1%, although over most test corpora there is only a small increase or decrease, indicating that integrating out-of-domain training data is difficult in this case. An alternative strategy of creating a model with only two features – the conditional probabilities from the in-domain and out-of-domain models – yields more modest improvements of around 0.6%, but more reliably.

[⋆] All percentage changes quoted in this section are absolute.

[⋆⋆] The precision, recall and F-score figures in this work and that described below can be broadly grouped into two categories: (1) constituent-based evaluation, loosely or exactly following PARSEVAL (Black et al., 1991), which requires parse tree constituents to have the same label and token span as the gold-standard; and (2) dependency-based evaluation (discussed more in Section 3.3), which requires matching dependency tuples between tokens of the sentence. These metrics produce different results, and even the relative changes should not be considered directly comparable. Results are often reported for only one of the two, however.

For adapting WSJ-trained parsers into the biomedical domain, Clegg and Shepherd (2005) investigate the performance of three treebank parsers (Collins, 1999; Charniak, 2000; Bikel, 2002) over the GENIA treebank (Ohta et al., 2002), finding that labelled constituent-based F-scores are 8–9% lower than those obtained over WSJ data, and that these errors can be slightly ameliorated by combining the outputs of different parsers in various ways. Lease and Charniak (2005) observe that PARSEVAL F-score from the Charniak parser trained on the WSJ is lower by 13% over GENIA and 4% over the Brown corpus, compared with parsing in-domain data which gets an F-score of 89.5%. They show that using shallow domain-specific resources such as a domain-specific POS tagger and named entities from a medical thesaurus avoids some of the cross-domain training performance penalty, increasing the GENIA F-score by 3.3%. A somewhat similar conclusion is found by Rimell and Clark (2009) using the C&C parser, then mapping the parser output to the grammatical relations (somewhat like dependencies) of the BioInfer corpus (Pyysalo et al., 2007) to calculate F-score. Using a domain-specific POS-tagger, and to a lesser extent a domain-tuned supertagger for the CCG lexical categories, improves F-score by 5.5% from the baseline of 76.0%. McClosky and Charniak (2008) show that it is possible to adapt the Charniak parser to the biomedical domain without a domain-specific treebank using self-training, i.e. learning from automatically created parser output, thus improving the baseline F-score of 80.4% over the GENIA corpus by 3.9%. In the HPSG space, Hara et al. (2005), also working on the GENIA corpus, show that it is possible to augment a larger log-linear model trained on the WSJ with carefully selected features derived from a smaller in-domain treebank. They report a 1.6% improvement in constituent F-score compared to a baseline of 85.1% using a WSJ model only, and a 0.5% improvement over simply retraining a new model from the combined WSJ and GENIA training data, while greatly reducing the training time. In later work, Hara et al. (2007) show that simply retraining the lexical entry features (rather than the grammatical ones) could yield further improvements of around 2% over this method.

3. Resources

3.1. CORPORA

At present, there are two corpora available for the ERG which are large enough individually for training a parse selection model.

First, we use the LOGON corpus (Oepen et al., 2004), a collection of English translations of Norwegian hiking texts from the LOGON project. It is freely available for download, and contains 8,550 sentences which have exactly one gold standard tree annotated in the treebank.[*] We divided these sentences into training and development data. The translations in LOGON include alternative English

[*] This omits six small sections ‘jhu’, ‘jhk’, ‘psu’, ‘psk’, ‘tgu’ and ‘tgk’.

translations of each source sentence from up to three different translators, although direct repetitions are very rare. To ensure that the similarities between these translations did not interfere with the results, we placed equivalent translations in the same section, so that two translations of the same sentence would never occur between both training and test corpora. The ERG was extensively developed for the LOGON project (particularly in terms of lexicon) so the grammar may show a slight bias towards this particular corpus as it was implicitly tuned for it, and as such, we would expect the corpus to be easier for the ERG to parse.

Second, we use the more recent WeScience corpus (Ytrestøl et al., 2009), a set of Wikipedia articles related to computational linguistics, which is, again, freely downloadable. With 9,167 sentences with a single gold-standard tree, it is slightly larger than the LOGON corpus in the number of sentences, and has somewhat longer average sentence length. Crucially for our purposes, it is in a very different domain, and beyond both corpora exhibiting fairly formal written prose, there is little similarity in content (we examine this in more detail in Section 3.1.1). We use only 7,631 of those sentences, again dividing them into a training set and a development set, preserving the remainder for future work.

We are able to use LOGON and WeScience as both training and test datasets to explore in- and cross-domain effects. We additionally experiment with a smaller-size corpus as test data only: the Cathedral and Bazaar corpus,* on open-source software development. This treebanked corpus is also included with the ERG distribution, and as a single-author text, could be expected to be slightly more homogeneous than the other corpora. Finally, we use one more corpus which is not yet publicly available, labelled ‘robot1’, which consists of transcripts of several spoken dialogues between two humans, one of whom is simulating a robot collaborating in a coloured-block-hunting task (Flickinger et al., 2009). As natural dialogue, the utterances in this corpus are on average relatively short and full of disfluencies, including many which are not full sentences. This makes it quite different to the carefully edited written prose that comprises the other three corpora we use. The various corpora are described in Table I.

While we are framing our work here as a problem of “domain”-adaptation, we have not actually defined what the term means. It is surprisingly difficult to find a satisfying definition for this somewhat nebulous concept. Informally, a domain can be considered to be a homogeneous topic paired with a particular register of language. The focused content of a domain corpus tends to lead to skewed lexical and constructional distributions, which have a direct impact on parsing as the skew in one domain tends to differ significantly from that in another. We return to explore the question of just how different our corpora are in Section 3.1.2, but in general we consider each corpus we use to consist of a single domain.

* http://www.catb.org/esr/writings/cathedral-bazaar (authored by Eric Raymond).

Table I. Corpora we use for our experiments and example sentences from each.

Corpus	Description	Example
WeScience (WESC)	Wikipedia articles	*There are a number of competitions and prizes to promote research in artificial intelligence.*
LOGON (LOG)	Hiking brochures	*The bird life is typical of an area of this kind.*
Cathedral and Bazaar (C&B)	Essay on Linux development	*I had been preaching the Unix gospel of small tools, rapid prototyping and evolutionary programming for years.*
Robot1 (ROBOT1)	Transcribed interactive dialogue	*okay I walked through the hallway um turned right*

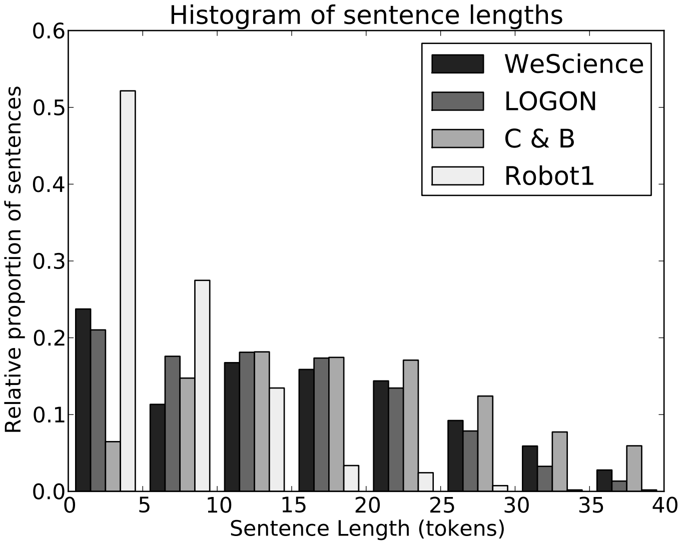


Figure 1. Histogram of sentence lengths for the corpora

3.1.1. *Corpus Characteristics*

In Table II we give broad statistics such as counts of tokens (using the tokenisation in the gold-standard ERG output) and types, including those which are outside the ERG's lexicon. WESCIENCE stands out as having a large vocabulary, as well as a high proportion of unknown types, with 43% of types outside the lexicon, which is fairly unsurprising given the nature of its content. LOGON has a lower proportion at only 9%, which may also reflect the history of the grammar – as

Table II. Corpora we use for our experiments showing numbers of "validated" sentences which have a single gold parse (which are the only sentences used in these experiments), average sentence length (in tokens), average ambiguity on test set (number of parses produced per sentence with each sentence capped at 500 parses, approximating, but underestimating, the difficulty of the parse selection problem), and the top-500 random baseline calculated as the average probability of randomly selecting the correct parse for each sentence from this top 500 according to the parse selection model used in treebanking (which is not truly random). To show the effects of the top-500 cutoff, we also show the number of validated sentences and average ambiguity over only sentences with < 500 parses. Also shown are statistics for various interesting tokens and constituents, both per-sentence and per-token (in square brackets). O.O.V. denotes 'out-of-vocabulary' for the lexicon of the '1010' version of the ERG which we use here.

	WeScience	LOGON	C&B	ROBOT1
Validated Sentences	7631	8550	567	1303
– training	6149	6823	0	768
– test	1482	1727	567	535
Parses/Sent	260.5	229.5	304.0	83.5
Top-500 random (exact)	13.3%	13.7%	7.8%	28.0%
Top-500 random (top 10)	35.3%	37.5%	24.5%	69.2%
Valid'd Sent, < 500 parses	4173	5338	265	1169
Parses/Sent, < 500 parses	62.1	66.8	80.7	35.8
Word Types	13662	7888	2744	664
O.O.V. Word Types	5931	744	306	15
Tokens	15.02 [1.000]	13.63 [1.000]	18.67 [1.000]	5.82 [1.000]
O.O.V. Tokens	2.02 [0.070]	1.49 [0.025]	1.66 [0.039]	1.11 [0.003]
Construction Rules	20.74 [1.380]	18.85 [1.383]	25.72 [1.378]	8.28 [1.421]
Lexical Rules	5.78 [0.382]	5.13 [0.372]	6.12 [0.322]	2.34 [0.245]
Noun Compounds	1.10 [0.073]	0.58 [0.043]	1.10 [0.059]	0.12 [0.021]
Co-ordination	0.65 [0.043]	0.67 [0.049]	0.56 [0.030]	0.09 [0.015]
Passives	0.54 [0.036]	0.27 [0.020]	0.28 [0.015]	0.02 [0.004]

noted above, it was extensively expanded for the LOGON project, so the lexicon is somewhat tuned to match that corpus. C&B, at only 11%, also shows a fairly small percentage of unknown types, which is perhaps influenced by having only a single author. The very small vocabulary of ROBOT1 has, perhaps unsurprisingly, an even lower proportion unknown-word proportion of 2.3%, although this could also be partly due to grammar tuning. As we would expect, on a per-token basis, the number of unknown words is much smaller, and the relative differences between the corpora also decrease, ranging from 0.3% of tokens for ROBOT1 to 7.0% for WeScience.

We also list the ambiguity of the sentences in terms of the number of parses postulated by the ERG (counting at most 500 parses per sentence, and hence underestimating), giving an idea of how difficult the parse selection task is beyond the rough estimates we can make on the basis of sentence length. ROBOT1 again stands out as being particularly different in this regard, with fewer than 100 per sentence, while LOGON is intermediate in difficulty compared to the progressively more ambiguous sentences of WESCIENCE and C&B, in line with the increased sentence lengths in these corpora.

We show some more details of sentence length in the histogram in Figure 1. The distributions are mostly unsurprising, with the longer-tailed distribution for C&B that we would expect for the longer average sentence length. The most noticeable value is the unexpectedly large value in the 0–5 range for WeScience, which is likely a reflection of the frequent occurrence of short article titles in this corpus.

Table II also shows statistics on the applications of ERG rules to give an idea of the relative complexities of the parse trees, separately to the sentence length. We show statistics on a per-sentence as well as a per-token basis, as the latter attempts to factor out the effects of sentence length on the complexity. The ERG rules can be divided up into "construction rules", which correspond to unary or binary rules accounting for syntactic phenomena, and "lexical rules", which can be roughly equated to morphological and morphosyntactic phenomena such as inflection.[⋆] We show a crude comparison of the number of applications of these rules in the gold-standard parse trees in the corpora. Construction rules are not meaningfully different on a per-token level, suggesting that perhaps after scaling for sentence length, the corpora have similar levels of syntactic complexity. However, there are substantially more applications of lexical rules in WESCIENCE and LOGON than in C&B and particularly ROBOT1. Some explanations for this are discussed in Section 3.1.2.

We also show some finer-grained statistics, intended to provide more insight into the distribution of some selected interesting syntactic phenomena which show different distributions between the corpora. WESCIENCE has a much higher incidence of the passive voice,[⋆⋆] with 0.036 instances per token indicating almost twice as many occurrences per token than the nearest competitor LOGON with 0.020. This is probably in line with what we would expect given the nature of the different domains, with WESCIENCE containing more formal academic-style prose, rather than the slightly more conversational style of LOGON. C&B has a lower incidence still – again the style is more conversational, and the essay is written from a first-person perspective. ROBOT1 has by far the lowest, but this is just what we would expect for spoken interactive dialogue.

[⋆] A third class of rules relates to bookkeeping for punctuation, which is linguistically less interesting and ignored here.

[⋆⋆] This was determined by counting instances of grammar rules which are subtypes of BASIC_PASSIVE_VERB_LR.

The relative differences between the corpora of frequency of noun compounds and coordination[‡] are smaller, but still noticeable. The technical subject matter of WESCIENCE may partially account for the greater frequency of noun compounds. The coordination differences are not as easy to explain, except for their infrequent occurrence in the simple ROBOT1 sentences.

3.1.2. *Inter-corpus comparisons*

It is clear that the corpora have different characteristics in terms of broad statistical counts, but it may also be informative to directly compare pairs of corpora to measure how alike they are in a more statistically-grounded way. While we have only briefly addressed the question of exactly what constitutes a domain, a statistical measure of corpus similarity should partially serve as a proxy for this. Counting how many words occur in only one of the corpora gives us some idea of the difference; however, this discards most of the distributional information. To make a more thorough comparison, we follow the technique of Verspoor et al. (2009) in using relative entropy to compare pairs of corpora, which we briefly describe here.

We choose some vocabulary V which is a subset of the union of the vocabularies of the two corpora, and we then construct probability distributions P_1 and P_2 using maximum likelihood estimation and add-one smoothing for each corpus. We can calculate the relative entropy over the words in that vocabulary using corpus 1 against corpus 2 as follows:

$$D(P_1||P_2) = \sum_{w \in V} P_1(w) \log_2 \frac{P_1(w)}{P_2(w)}$$

This gives us a way of quantifying how different the distributions of words are between corpora. Also following Verspoor et al., we show values for different frequency cutoffs after sorting the vocabulary by combined frequency between the two corpora.

However, we may also be interested in the words which most strongly characterise the differences between the corpora. Rayson and Garside (2000) outline a way to achieve this using log-likelihood. Given two corpora with total number of tokens T_1 and T_2, we can calculate the expected frequency count for a particular word with observed counts t_1 and t_2 in each corpus as follows:

$$E_1 = T_1 \frac{(t_1 + t_2)}{T_1 + T_2} \quad E_2 = T_2 \frac{(t_1 + t_2)}{T_1 + T_2}$$

From this, we can calculate the log-likelihood for the particular word:

$$L = 2 \left(t_1 \log_2 \frac{t_1}{E_1} + t_2 \log_2 \frac{t_2}{E_2} \right)$$

[‡] These are subtypes of BASIC_N_N_CMPND_PHR and BASIC_COORD_PHR respectively.

If we sort the log-likelihood values for each word by decreasing log-likelihood values, those items at the top of the list are those which are most different between the corpora and thus in one view characterise the differences between the corpora.

In our work, we are also interested in the distributions of syntactic constructions between the corpora to see whether the differences extend beyond the vocabularies. We can achieve a coarse version of this using a variant of the word-based procedures described above. We take advantage of the fact that the rules in the ERG are named according to their syntactic function. The grammar has some 200 construction rules and lexical rules to account for different phenomena. For example, three subject-head rules account for the subject of a sentence attaching to the verb phrase in different circumstances, while twelve different rules are used for different kinds of noun-compounding. Like lexical items, these rules have widely different relative frequencies of use from one another, and we also expect these rules may have different frequencies of usage between different corpora. By replacing words in the procedures described above with rule names, so that the vocabulary is the complete inventory of available rules, we can calculate relative entropy figures and log-likelihood figures across lexical rules, construction rules, or a combination of both. In fact Rayson and Garside (2000) note that their procedure could be applied to other entities such as POS-tags, so applying it to syntactic construction names is within the spirit of the original method.

We could possibly improve over aggregating solely by rule name – some very general rules such as the head-complement rules can apply to a wide range of lexical heads with complement slots, including prepositions, verbs, adjectives and nouns, and we are ignoring this potentially interesting information. Nonetheless, this fairly simple procedure provides a concise indication of the relative syntactic similarity between the corpora.

We show the results for log-likelihood and relative entropy against subsets of each training corpus in Table III, also showing the relative entropy in graphical format in Figure 2. First, examining the words with the greatest log-likelihood differences, we can see some instructive differences between the corpora (noting that items prefixed with '¬' (e.g. '¬language_n1') are more frequent in the reference corpus, not the test corpus). C&B and ROBOT1 are characterised by much more frequent use of the pronoun *I* than the other corpora, which is predictable as C&B is a conversational first-person essay and ROBOT1 is a transcription of dialog. The second-person pronoun *you* is somewhat characteristic of all the corpora (less strongly for C&B) compared to WESCIENCE, again indicative of the styles we would expect. The other more specific vocabulary items are also clearly characteristic of the domain – it is unsurprising that C&B talks about *bazaars* and *software*, LOGON talks about *trails* and *routes* while WESCIENCE talks about *language*.

The relative entropy differences between the in-domain comparison versus out-of-domain comparison are striking. The in-domain relative entropy is 0.03 or less for the top-100 words and 0.25 or less for the top-1000, versus 0.36 or more and 0.96 or more for the respective out-of-domain figures. It is perhaps not so surprising

Table III. Test corpora compared with reference training corpora (1833-sentence subset of WeScience; 1710-sentence subset of LOGON). The in-domain test set (i.e. a different subset of the same source corpus) is labelled in bold. Relative entropy is shown for the reference corpus against test corpus and then vice versa, and uses add-one smoothing, only examining items with at least 5 occurrences (10 for grammatical rules) in the combination of the corpus pair. Additionally there is a cutoff by rank in combined frequency list for lexical items (100 or 1000 words). Div(ergen)t rules and words are those with the highest log-likelihood (which is shown) against the combined distribution, with the item prefixed with '¬' if the highest frequency is in the reference distribution.

Ref: WeSc (train)	**WeScience**	LOGON	C&B	robot1
Types in ref	50.9%	32.6%	48.7%	51.7%
Tokens in ref	86.7%	63.7%	78.6%	63.3%
Rel-Ent (Top-100)	0.00/0.03	0.52/1.14	0.38/0.36	1.04/4.14
Rel-Ent (Top-1k)	0.11/0.14	2.09/2.70	0.96/1.61	1.71/7.45
Div't lexemes		457.0: ¬language_n1 354.7: you 251.6: trail_n1 231.5: trip_n1 199.3: route_n1	463.2: i 185.7: ¬language_n1 128.5: bazaar_n1 89.7: project_n1 88.5: you	836.4: okay_s_adv 794.4: box_n1 511.4: you 367.5: i 363.0: yeah_root_pre
Rel-Ent (all rules)	0.01/0.01	0.33/0.34	0.27/0.28	1.82/2.94
Rel-Ent (constr.)	0.01/0.01	0.29/0.31	0.28/0.28	1.99/3.23
Rel-Ent (lex rules)	0.00/0.00	0.46/0.42	0.19/0.23	0.94/0.95
Ref: LOG (train)	WeScience	**LOGON**	C&B	robot1
Types in ref	22.6%	54.3%	31.7%	57.1%
Tokens in ref	59.2%	87.6%	65.6%	69.9%
Rel-Ent (Top-100)	1.25/0.51	0.02/0.03	1.08/0.49	1.06/3.25
Rel-Ent (Top-1k)	2.72/2.01	0.25/0.24	2.11/2.11	1.76/6.28
Div't lexemes	356.9: language_n1 279.4: ¬you 234.0: ¬trail_n1 228.2: ¬trip_n1 208.7: ¬route_n1		466.6: i 168.3: project_n1 155.1: linux_n1 151.8: software_n1 132.8: ¬trail_n1	820.9: okay_s_adv 779.6: box_n1 373.2: i 356.2: yeah_root_pre 300.5: room_n1
Rel-Ent (all rules)	0.35/0.34	0.01/0.02	0.43/0.40	1.60/2.68
Rel-Ent (constr.)	0.32/0.29	0.01/0.02	0.40/0.35	1.85/2.98
Rel-Ent (lex rules)	0.47/0.50	0.01/0.01	0.53/0.59	0.45/0.62

that we should see low relative entropy on a per-word basis within a domain, and high relative entropy on a word-by-word basis compared to other corpora with different subject matter and register, where we expect a widely different vocabulary (as the figures for token-overlap with the reference corpus indicate).

It is more interesting that the corpora have widely different distributions of syntactic constructions and lexical rule applications, even though these differences are much less dramatic in terms of bits of entropy, with out-of-domain corpora having as few as 0.19 bits of relative entropy compared to in-domain. It is not necessarily clear that this should be the case. While we might suspect that a dialogue-based

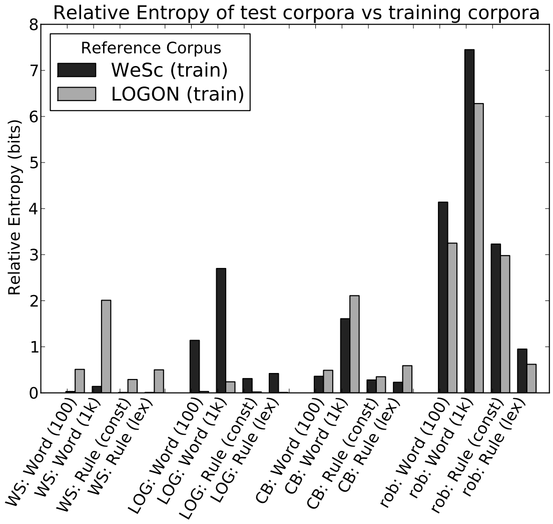


Figure 2. Graphical representation of relative entropy of the test corpora against the two reference corpora (subset of the information in Table III)

corpus like ROBOT1 would have a higher percentage of imperative constructions for example, we could not know whether this would have a noticeable impact on the overall distribution. In fact, ROBOT1 stands out as being very different syntactically, but we can also see some other noticeable differences between the corpora. C&B seems to be most similar to WESCIENCE in terms of syntax as well as vocabulary.

Examining the details of the rules that characterise the inter-corpus differences, they are less easy to explain than the lexical items, even with knowledge of the meanings of the rules within the ERG internals. WESCIENCE has a disproportionately high number of occurrences of HDN_BNP_C, corresponding to noun phrases headed by common nouns with no determiners such as *algorithms*, possibly due to article titles and factual, general subject matter. Meanwhile LOGON has a preponderance of HDN_BNP-PN_C instances, denoting bare noun phrases with proper nouns, like *Jotunheimen* (a region in Norway), which is probably related to the large number of place names mentioned. Another unusually frequent construction in LOGON is NP_ADV_C, for adverbial noun phrases like *here* in *They came here*. The discrepancy between proper noun usage may also partially explain the much higher prevalence of HDN-OPTCMP_C in WESCIENCE, which is used for common nouns with optional complements. A cursory examination of the parse trees suggests that another contributing factor is the fact that nouns with optional complements are often nominalisations like *implementation*, and such constructions seem particularly frequent in the technical subject matter of WESCIENCE.

From all of this, we can see some reasons why a parse selection model trained in one domain may perform poorly over a second domain. Not only are there differences in distributions of lexical items to take into account, there are also widely different distributions of rule applications, both lexical and syntactic. As the parse selection models we use take into account rule names of nodes and their ancestors as well as individual lexical entries, all of this could be important.

3.2. EXACT MATCH EVALUATION

As we mentioned in the introduction, one of the differences between this work and previous work is the different motivation for domain adaptation: in addition to wanting the best top-ranked parse possible, we also intend to use the parser to build *treebanks* for new domains.

Previous work using the ERG and related DELPH-IN grammars has generally reported results on the basis of exact match of the top parse or exact match within the top-10 parses (Zhang et al., 2007). One reason for the usage of exact match accuracy is to reflect the utility of a ranking model for treebanking.

In Redwoods-style treebanking (Oepen et al., 2004), the parse selection model is crucial for two reasons. Firstly, a correct parse close to or at the top of the ranking enables the treebanker to quickly select it as the gold-standard parse tree. Secondly, the treebanking process requires the selection of some ad-hoc cutoff for the number of highest-ranked parses to present to the treebanker, which is usually set to 500. This number decides the balance between tractability in the treebanking process for the treebanker (and, to a lesser extent, the underlying machinery), and completeness in terms of not excluding correct but low-ranked parses. Inevitably, there will be a small amount of 'leakage' — correct parses which are not ranked high enough to be considered — but we can reduce this while holding the number of parses constant by providing a better parse selection model. So, better parse selection models enable treebanks to be built more quickly and with greater coverage.

We use notation Acc_N to denote the exact gold-standard tree being found somewhere in the top-N parses. In terms of treebanking utility, Acc_{500} tells us whether the correct analysis is available to the treebanker in the top-500 analyses; conversely, it is possible to calculate the average N required to achieve, say, 80% exact match parsing accuracy. However, these values are expensive to calculate for dozens of different parser configurations. Following Zhang et al. (2007), we use less-expensive Acc_1 (primarily) or Acc_{10}, which we argue act as a as a proxy for Acc_{500}; this is evaluated in Section 5.1. Both of the figures also tell us important information about the treebanking utility of the parse selection model: whether the target parse is ranked highest, or occurs within a highly-ranked manageable group of trees.

3.3. EDM EVALUATION

Granular evaluation metrics of the type generally reported in other parsing research are complementary to the exact match metric, in that rather than evaluating whether the top parse is completely correct, they indicate how close the top parse is to the gold standard. There is currently no standard granular evaluation metric suitable for evaluating the detailed output of the PET parser. While it may seem that existing metrics could be used, hence allowing direct comparison, recent work from Clark and Curran (2007a) and Miyao et al. (2007) has shown that mapping between different parser representations is a far from trivial task. Moreover, since our main purpose in this work is to measure effects of domain on parse selection within one formalism, we are more interested in seeing how close we can get to the analysis that our parser aims to produce than in directly comparing with parsers with different output. For that reason, building on the work of Kaplan et al. (2004), Briscoe and Carroll (2006), and Plank and van Noord (2008), *inter alia*, we adopt a dependency tuple-based evaluation method, but in a formalism that is native to the ERG and also an interestingly large family of DELPH-IN grammars couched in the HPSG formalism.

The goal of the PET parser is to extract meaning from text, and we consider three classes of information that contribute to meaning:

class 1: core argument structure, either surface or deep structure

class 2: predicate information, such as the lemma, POS and sense

class 3: properties of events and entities, such as tense, number and gender

Many existing dependency-based evaluation metrics use grammatical relations to describe **class 1** information. That is, the dependencies are usually labels like SUBJ, OBJ and MOD (or ADJUNCT). While these grammatical functions allow us to describe the surface level linguistic structure, they do not make explicit the underlying deep structure of any utterance. This deep structure describes semantic rather than syntactic arguments and can be seen in resources such as the Prague Dependency Treebank (Böhmová et al., 2003), the Redwoods treebank (Oepen et al., 2004) and PropBank (Kingsbury et al., 2002). Using this semantic argument structure for parser evaluation not only gets closer to the actual sentence meaning that we are trying to extract, but is potentially more general, as there is generally wider agreement on semantic arguments than on, for example, whether the main verb depends on the auxiliary, or vice versa. Minimal Recursion Semantics (MRS: Copestake et al. (2005)) is one formalism (and the one which is native to the ERG) that allows us to describe this semantic argument structure, and our granular evaluation metric is based on this formalism.

3.3.1. *Minimal Recursion Semantics*

Minimal Recursion Semantics (MRS) is a flat semantic formalism that represents semantics with a bag of *elementary predications* and a list of scopal constraints. An

$$
\begin{array}{l}
\langle\, h_1, \\
\quad h_3\text{:pron}{<}0{:}2{>}(\text{ARG0 } x_4\{\text{PERS } \mathit{3}, \text{NUM } \mathit{sg}, \text{GEND } \mathit{m}, \text{PRONTYPE } \mathit{std\_pron}\}), \\
\quad h_5\text{:pronoun\_q}{<}0{:}2{>}(\text{ARG0 } x_4,\ \text{RSTR } h_6,\ \text{BODY } h_7), \\
\quad h_8\text{:\_persuade\_v\_of}{<}3{:}12{>}(\text{ARG0 } e_2\{\text{SF } \mathit{prop}, \text{TENSE } \mathit{past}, \text{MOOD } \mathit{indicative}\}, \\
\qquad\qquad\qquad\qquad\qquad\quad \text{ARG1 } x_4,\ \text{ARG2 } x_{10},\ \text{ARG3 } h_9) \\
\quad h_{11}\text{:proper\_q}{<}13{:}16{>}(\text{ARG0 } x_{10}\{\text{PERS } \mathit{3}, \text{NUM } \mathit{sg}\},\ \text{RSTR } h_{12},\ \text{BODY } h_{13}), \\
\quad h_{14}\text{:named}{<}13{:}16{>}(\text{ARG0 } x_{10},\ \text{CARG } \mathit{Kim}), \\
\quad h_{15}\text{:\_leave\_v\_1}{<}20{:}26{>}(\text{ARG0 } e_{16}\{\text{SF } \mathit{p\text{-}or\text{-}q}, \text{TENSE } \mathit{untensed}, \text{MOOD } \mathit{indicative}\} \\
\qquad\qquad\qquad\qquad\qquad \text{ARG1 } x_{10},\ \text{ARG2 } p_{17}) \\
\{\, h_{12} =_q h_{14},\ h_9 =_q h_{15},\ h_6 =_q h_3 \,\}\,\rangle
\end{array}
$$

Figure 3. MRS representation of *He persuaded Kim to leave.*

elementary predication can be directly related to words in the text, or can reflect a grammatical construction, such as compounding. Each elementary predication has a relation name, a label and an index (designated ARG0). Arguments of a predication are represented by 'bleached' ARG*n* roles (which are to be semantically interpreted for classes of predicates). Figure 3 shows the MRS representing the semantic analysis of *He persuaded Kim to leave*. Here we see six elementary predications, four with text referents and two as construction-specific covert quantifiers. The ARG1, ARG2 and ARG3 roles of the verbal predicates describe the predicate–argument relations and demonstrate co-indexation between the ARG2 of *persuade* and the ARG1 of *leave*. Entities and events carry properties such as gender or tense which are attached to their index variables. An evaluation scheme based on MRS therefore allows us to evaluate **class 1** information using the roles, **class 2** information through predicate names and **class 3** information from the properties of the index variables.

3.3.2. *Elementary Dependencies*

The metric we use is Elementary Dependency Match (EDM: (Dridan, 2009)), based on elements that Oepen and Lønning (2006) defined as Elementary Dependencies (EDs), a variable-free reduction of MRS which excludes scopal information. We differ from their definition by using sub-string character spans (e.g. <3:12>) instead of predicate names (_persuade_v_of) to represent nodes in the dependency graph. In keeping with our information classes, this allows us to separate the evaluation of **class 2** information from **class 1**. Our EDM metric hence consists of three types of triples which align with the three information classes:

ARGS:	$span_i$	$role_j$	$span_k$
NAMES:	$span_i$	NAME	$relation_i$
PROPS:	$span_i$	$property_j$	$value_j$

In these forms, $relation$ is the predicate name of an elementary predication from the MRS, $role$ is an argument label such as ARG1, $property$ refers to an attribute such as TENSE or GEND and $value$ is an appropriate instantiation for the respective property. Figure 4 shows the triples produced for the MRS in Figure 3.

The text segment associated with each character span is shown for illustrative purposes, and is not part of the triple.

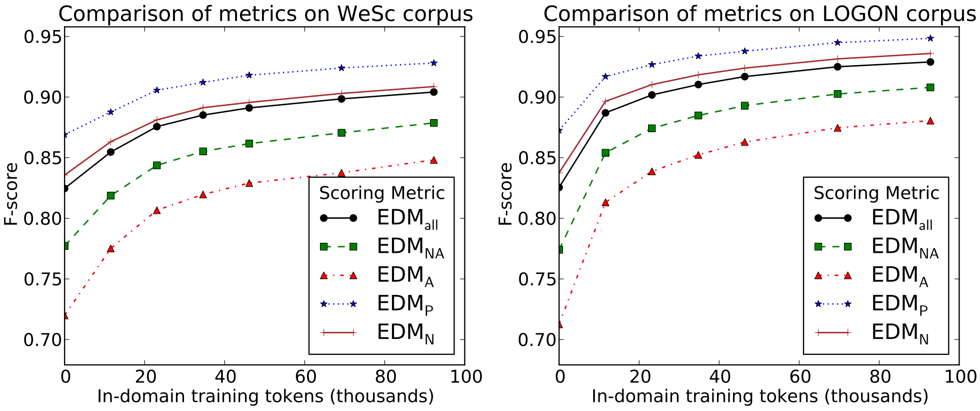
“He”	<0:2>	ARG0	<0:2>	“He”
“persuaded”	<3:12>	ARG1	<0:2>	“He”
“persuaded”	<3:12>	ARG2	<13:16>	“Kim”
“persuaded”	<3:12>	ARG3	<20:26>	“leave.”
“Kim”	<13:16>	ARG0	<13:16>	“Kim”
“leave.”	<20:26>	ARG1	<13:16>	“Kim”
“He”	<0:2>	NAME	pronoun_q	
“He”	<0:2>	NAME	pron	
“persuaded”	<3:12>	NAME	_persuade_v_of	
“Kim”	<13:16>	NAME	proper_q	
“Kim”	<13:16>	NAME	named	
“leave.”	<20:26>	NAME	_leave_v_1	
“He”	<0:2>	GEND	*m*	
“He”	<0:2>	NUM	*sg*	
“He”	<0:2>	PERS	*3*	
“He”	<0:2>	PRONTYPE	*std_pron*	
“persuaded”	<3:12>	MOOD	*indicative*	
“persuaded”	<3:12>	SF	*prop*	
“persuaded”	<3:12>	TENSE	*past*	
“Kim”	<13:16>	NUM	*sg*	
“Kim”	<13:16>	PERS	*3*	
“leave.”	<20:26>	MOOD	*indicative*	
“leave.”	<20:26>	SF	*p-or-q*	
“leave.”	<20:26>	TENSE	*untensed*	

Figure 4. Gold triples for *He persuaded Kim to leave.*

During evaluation, we compare the triples from the gold standard analysis with those ranked top by the parser, and calculate precision, recall and F_1-score across all triples, as well as across the three separate triple types (NAME, ARG and PROP).

3.3.3. *Analysis*

The full EDM metric weights each triple equally, which may not be ideal for all scenarios. The division by triple type gives one alternative view that provides a more complete picture of what sort of mistakes are being made by the parser, but other weightings are also possible. To get some idea of the numeric range of the different EDM configurations, we parsed sections of the WESCIENCE and LOGON corpora using the English Resource Grammar (ERG: Flickinger (2000)), ranking against



(a) WeScience (b) LOGON

Figure 5. Comparison of variants of the EDM metric

parse selection models trained on different quantities of in-domain data (this idea is developed further in Section 5.1). Figure 5 shows the relative differences between five configurations: all triples together (EDM$_{\text{all}}$), the NAME, ARG and PROP triple types separately (EDM$_{\text{N}}$, EDM$_{\text{A}}$ and EDM$_{\text{P}}$, respectively) and measuring just the NAME and ARG types together (EDM$_{\text{NA}}$). This last configuration produces a metric broadly similar to the CCG dependencies used by Clark and Curran (2007b) and also to the predicate argument structures produced by the ENJU parser (Miyao and Tsujii, 2008), in that, in all cases, the same classes (**class 1** and **class 2**) of information are being evaluated[⋆].

Here and below, we create the zero-training data points on the y-axes for EDM by randomly selecting a parse from the (at most) top-500 parses included in the treebanks distributed with the ERG, and choosing one of those at random to create the EDM triples. We repeat the process over the whole test corpus 10 times to get an average, since we cannot simply use the raw number of parse trees to calculate the probability of randomly selecting the correct one, as is possible with exact match. Because the ranking of the top-500 parse trees is based on a parse selection model, this is not truly random as it uses training data to create the 500-tree parse forest (the same is also true for exact match). However, a truly random baseline is very difficult to create, as it is not possible to create a genuinely random parse forest.

We can see that all EDM configurations show approximately the same trends, and maintain their relative order. EDM$_{\text{P}}$ is consistently higher, which follows from the fact that many of the properties are inter-dependent, and that the parser enforces agreement, so that getting one unambiguous property correct via the lexicon or the morphology will lead to getting others correct. The most difficult type of triple

[⋆] While predicate information could be mostly aligned between the three metrics, the actual predicate arguments that represent **class 1** are different enough in each case due to different analyses of, for example, conjunctions and infinitival *to*, that direct comparison is not practical.

to correctly identify is the ARG type, which represents the core semantic arguments. We will primarily report results using $\mathrm{EDM}_{\mathrm{NA}}$ as the configuration most like previous work. All of the scores are fairly high – even the no-training data baseline (subject to the caveat above about this actually using a parse selection model) for the most difficult $\mathrm{EDM}_{\mathrm{A}}$ relationship has an F-score above 0.7, and the most compatible mode $\mathrm{EDM}_{\mathrm{NA}}$ starts at 0.77 for the baseline, and rises to between 0.87 and 0.89 where there is abundant in-domain training data. The scores for $\mathrm{EDM}_{\mathrm{NA}}$ are calculated from 2.8–2.9 tuples per token, depending on the corpus.

We emphasise that we do not expect these metrics to produce results that are directly compatible with dependency evaluations from other formalisms. As Dridan (2009) notes, even with simple dependency systems, there is a large range of differences reflecting the different assumptions made in the grammar, and mapping between them is an object of research in its own right (e.g. Clark and Curran (2007a)). However, our $\mathrm{EDM}_{\mathrm{NA}}$ metric is broadly comparable to other dependency-based metrics, and it provides an evaluation of parse quality which complements the exact match metric. In particular, it may provide a better indication of utility to downstream applications than exact match, since many 'near-match' trees that score zero in the Acc_1 metric may in fact get most dependencies correct. Software to produce the EDM results will be made available as part of the DELPH-IN code distribution.[⋆]

3.4. COMPARISON OF EVALUATION METRICS

It is also worth considering at this point whether the differences we could observe between the behaviours of exact match and dependency-based evaluation are purely because the EDM metric is far less sensitive to effects of sentence length. Correcting Acc_1 for sentence length is difficult, but we show in Table IV the various results aggregated by sentence length trained on solely in-domain data. As the sentences get longer, the number of parsing decisions needed increases exponentially, so it is unsurprising that exact match accuracy decreases as the sentence lengths increase, but it is not necessarily clear that this should be be the case for $\mathrm{EDM}_{\mathrm{NA}}$, since we may be getting most dependencies correct within the sentence. The table suggests that this is indeed the case – the longer sentences are harder to parse even relative to their sentence length, as the relatively length-independent EDM metric does indeed decrease. The question of whether EDM gives us more than we could get from correcting Acc for sentence length is difficult to answer. However, we can see that the well-populated 30–34 range has a lower Acc_1 and Acc_{10} than the 25–29 (unsurprisingly), but a higher EDM F-score, which lends some support to this notion.

Relatedly, we can see that LOGON gets higher exact match performance over a similar quantity of training data than WESCIENCE – we would like to know if this is purely because of the slightly higher sentence length and average ambiguity

[⋆] http://www.delph-in.net

Table IV. WeScience and LOGON test corpora using models trained on all in-domain tokens aggregated by sentence length, scored against exact match (Acc_1), top-10 match (Acc_{10}) and EDM_{NA}. EDM_{NA} figures come from an average of 2.81 gold tuples per token for WeScience and 2.87 per token for LOGON.

	WeScience				LOGON			
Length	Sents	Acc_1	Acc_{10}	EDM_{NA}	Sents	Acc_1	Acc_{10}	EDM_{NA}
				P / R / F				P / R / F
0–4	350	87.4	99.7	93.7 / 93.7 / 93.7	361	93.1	100.0	97.1 / 97.0 / 97.1
5–9	167	62.3	94.6	90.5 / 89.8 / 90.1	302	71.2	96.0	93.4 / 93.1 / 93.2
10–14	247	46.6	88.7	91.4 / 89.6 / 90.5	311	58.5	90.0	93.4 / 93.6 / 93.5
15–19	234	26.5	65.4	88.8 / 87.1 / 88.0	298	34.6	71.8	91.7 / 89.2 / 90.5
20–24	212	20.3	61.8	89.1 / 87.2 / 88.1	231	20.3	58.9	89.9 / 88.6 / 89.2
25–29	136	12.5	36.8	87.5 / 83.5 / 85.4	135	11.1	31.1	89.2 / 88.9 / 89.0
30–34	87	6.9	27.6	88.4 / 87.2 / 87.8	56	10.7	32.1	90.2 / 86.2 / 88.2
35–39	36	2.8	16.7	86.7 / 84.8 / 85.8	21	4.8	14.3	90.4 / 81.9 / 85.9
40–44	11	0.0	9.1	85.1 / 68.9 / 76.1	7	0.0	0.0	88.3 / 77.5 / 82.5
45+	2	0.0	50.0	98.5 / 35.7 / 52.4	5	20.0	20.0	96.1 / 97.0 / 96.5
ALL	1482	44.1	73.7	89.2 / 86.6 / 87.8	1727	52.5	77.9	91.5 / 90.0 / 90.8

of WeScience. The lower EDM score for in-domain WeScience compared to in-domain LOGON in Figure 5 makes us suspect that WeScience is intrinsically a ‘harder’ corpus to parse with the ERG (which would be unsurprising, given that it has been more extensively tuned for LOGON), but we would like to evaluate this against Acc as well. The results in Table IV lend some support to the greater difficulty of WeScience, even for sentences of similar length: all ranges with more than 20 sentences have higher performance for LOGON.

Splitting results by sentence length also allows us to more closely examine the EDM numbers. For both corpora in Table IV, we see a preference for precision over recall when measured over all sentences, a trend that is repeated in all EDM results reported here. When broken down by sentence length, we see that this trend is fairly stable, but more pronounced as the sentences get longer. Error analysis at the individual sentence level has shown us that this imbalance reflects a slight but genuine tendency of the parser towards simpler analyses with fewer dependencies, which is perhaps an artefact of the parse selection models being designed to produce the correct syntactic analysis, rather then being optimised to produce semantic dependencies.

4. Experiments

The test and training corpora are all originally parsed with PET (Callmeier, 2000) and the most recent “1010” version of the ERG. The text was pre-POS-tagged with TnT (Brants, 2000) to enable handling of unknown words, and other preprocessing is done using the default settings for the corpus as configured in the [incr tsdb()] performance and competence profiler (Oepen, 2001).

We evaluate over corpora in several different domains, training discriminative parse selection models based on manually annotated treebank data drawn from the corresponding training corpus.

4.1. METHODOLOGY

In our experiments we are interested first in the effects of domain on parse selection accuracy, as evaluated using our various evaluation metrics (exact match at varying top-N beam widths, and EDM F-score). The second stage of our experiments is designed to investigate the domain adaptation problem, by evaluating different methods of combining in- and out-of-domain data, and to explore how much in-domain data is required to make an appreciable difference to parsing accuracy in a particular domain. Both stages of the experiment drew from the same training and test data sections.

To generate training and test data from the two larger corpora (LOGON and WeScience), we first shuffled the items in each corpus with no regard for section boundaries,[⋆] and then selected roughly equal-sized training and test item sets from each (see Table II for the exact item numbers). Finally, we randomly divided the respective training sections into fixed subsections to use in generating learning curves, ensuring there were approximately equal numbers of tokens in the sections from each training corpus (Table II shows this also corresponds to a similar number of rule applications, since WESCIENCE and LOGON both have 1.38 construction rules per token and around 0.38 lexical rules). The same combinations of training data subsections for a given amount of training data were used in all experiments.

We train the discriminative parse selection models in the framework of Velldal (2007), along the lines of previous work such as Zhang et al. (2007), which involves feeding in both correct and incorrect parses licensed by the grammar to the TADM toolkit (Malouf, 2002), and learning a maximum entropy model. In all experiments, we use the default feature function sets from previous work, with training parameters selected from the grid search we conducted.

[⋆] LOGON, however, contains either 2 or 3 English translations of each original Norwegian sentence (as noted in Section 3.1), so as part of this, we ensured that sentences translated from the same source sentence were placed together in the partitioning, to limit the chance of having very similar sentences in multiple sections.

4.1.1. *Grid searching to optimise training parameters*

To train our models, it is necessary to select in advance a number of parameters. Firstly, we need to set the standard maximum entropy parameters required by TADM: absolute tolerance, relative tolerance and variance. Additionally, Velldal (2007) notes the importance of other significant thresholds related to the features themselves rather than the learner. *Grandparenting level* is the maximum depth above a target node we will use to extract features when training the model. A level of 2 means that when we examine a particular target node, we create features based on the immediate parent node (one level) as well as the grandparent (two levels). This can improve performance (Zhang et al., 2007), but can also cause data-sparseness for small training sets, which we will often be dealing with here.

Another important feature-extraction parameter is the *relevance count threshold*. Following Malouf and Van Noord (2004), a feature is relevant if it takes different values between known correct and incorrect parses, and thus is useful for discriminating between different outputs. The relevance count of a given feature is the number of input strings for which the feature is relevant, so by demanding a minimum value for this over the whole corpus, we affect which features are included in the model.

While we could have selected these parameters on the basis of solid performers in prior work such as Zhang et al. (2007), most of this work assumes a relatively large training set, which is not always the case for our work here; for example, learning curves by their nature require a smaller data set. Velldal (2007) details performing a grid search to optimise these parameters. Essentially we can enumerate all possible values we wish to test for each parameter, and evaluate the performance of each possible combination of those parameters. We can achieve this using cross-validation over our existing corpus, repeatedly holding out a different subset to test on, and training on the remainder then aggregating the results over all the folds. The maxent training parameters and the model selection parameters can be optimised simultaneously, since there may be interactions between them. For training, we make the reasonable approximation that relevance counts over $\frac{7}{8}$ of the corpus that we use in 8-fold cross-validation are approximately the same as those over the whole corpus, to avoid the need for multiple expensive feature extraction operations for each fold. The optimal parameter combinations are then simply those with the highest exact match accuracy over the training data.

Here, we would like to ideally find the set of parameters which performs optimally over each different set of training data – which is dozens of different training sets. To perform grid searches over all of these, and have different parameters configured for all of these would be prohibitively expensive and complex. Instead, we aimed to find a single set of parameters which performed close to optimally in a range of configurations, from a small single corpus training set to a larger training set encompassing both corpora.

We performed a grid search over the following training corpus combinations, to make sure that we were testing over a representative range of corpus sizes and combinations reflecting the diverse corpus sizes we would be using for training data:

- 11,000-token subsets of both WeScience and LOGON (750–850 sentences)
- 23,000-token subsets of WeScience, LOGON and a 50/50 split of the two
- 46,000-token subsets of WeScience, LOGON and a 50/50 split of the two
- 92,000 tokens of a 50/50 WeScience/LOGON split
- Full WeScience and LOGON training corpora (185,000 tokens/13,000 sentences)

For each corpus combination, we tested 288 different parameter combinations (all possible combinations from those listed below), trying a range of values for each parameter comfortably spanning those which produced near state-of-the-art performance in (Zhang et al., 2007). We aggregated accuracy over the cross-validation folds to give an overall accuracy figure for each parameter combination, and considered only combinations that ranked in the top 100 out of 288 for all of the corpus combinations (suggesting they are robust across different training data sizes), giving 22 parameter combinations. From these, we selected the combination with the highest mean rank, although all 22 combinations had exact match accuracy within 0.8% of the best for the corpus, so in practice any would perform well. For tractability, we followed the standard practice of Velldal (2007) in using a pre-parsed corpus of the top 500 parse trees for each sentence according to some previous model. In each iteration we reranked these parse trees using the new model rather than exhaustively reparsing, which saves a considerable amount of computation time and produces approximately equivalent accuracy figures.

We tested the following parameters in different combinations (the parameters we ultimately selected using the described procedure are highlighted in bold):

- Relevance count threshold: 1, 2, 3, $\mathbf{5}$
- Grandparenting level: 1, 2, $\mathbf{3}$, 4
- Maxent relative tolerance: 10^{-6}, 10^{-8}, $\mathbf{10^{-10}}$
- Maxent variance: 10^{4}, 10^{2}, 10^{0}, 10^{-2}, $\mathbf{10^{-4}}$, 10^{-6}

The parameters selected differ from Zhang et al. (2007) in using only 3 levels of grandparenting rather than 4, which is unsurprising given the comments above about data-sparseness in smaller training corpora.

4.1.2. *Single domain models*

To evaluate the cross-domain effect, we use the parameters found during the grid search and train parse selection models over differing amounts of training data from LOGON and WeScience. We apply them both in-domain (to the corresponding held-out test data) and cross-domain (to the test datasets for the other corpora) and look at the relative differences between domains, according to our various evaluation metrics. We also created learning curves using different-sized subsets of both

WeScience and LOGON as the training data against each test corpus, including the in-domain corpus.

4.1.3. *Models from unmodified concatenated data*

Next we looked at some simple strategies for alleviating the cross-domain penalty. From the work described in Section 2.2, there are various strategies for achieving this. Domain-adapted POS-taggers or supertaggers (Lease and Charniak, 2005; Rimell and Clark, 2009), have been successful in the biomedical domain, in the latter case for helping to constrain the parser search space. This is not something we investigate here since domain-adapted tagging models are not readily available; additionally, in the parsing configuration we use, POS tagging is only used for unknown words, but investigating the utility of POS-tagging for domain adaptation is a possible area for future work.

Given that we already have relatively large training corpora in separate domains, a logical first step is to see how much manually-treebanked data we need in order to improve accuracy in a new domain, whether we should combine this with existing data, and how best to do so.

First, we evaluated a relatively naive method for combining training data from the LOGON and WeScience domains into one model. In this strategy, denoted CONCAT, we simply concatenate the training data sets and train a MaxEnt model from this, in the same way as the “combined” method of Hara et al. (2005). To simulate the effects of differing amounts of treebanking effort, we varied the amount of in-domain training data used in each model.

4.1.4. *Linear combinations of trained models*

An alternate approach to CONCAT, which we denote COMBIN, is to use MaxEnt models derived from simple arithmetic combinations of other pre-trained models, similar to McClosky et al. (2010). A MaxEnt model is simply a set of feature weights. If we have two trained MaxEnt models A and B, and we wish to use weighting parameters α and β to combine these models into a new model C, the weight λ for a given feature in C will be given by $\lambda_C = \alpha\lambda_A + \beta\lambda_B$ where λ_A and λ_B have values of zero if the given feature is not explicitly encoded in the model. Since parse ranking uses the unnormalized MaxEnt score, only the ratio $\frac{\alpha}{\beta}$ matters, we constrain α and β to sum to one, preventing multiple equivalent models. For example, assume we have the following weighting parameters and feature/weight pairs (i.e. vectors of λ values) for each model:

$$\alpha = 0.3 \qquad \beta = 0.7$$
$$m_A = \{1 : 1.5, 2 : 2.0\}$$
$$m_B = \{1 : 0.5, 3 : -1.0\}$$

The resulting model is simply the weighted linear combination:

$$m_C = \{1 : 0.8, 2 : 0.6, 3 : -0.7\}$$

For our experiments here, the pre-trained models come from exactly one single-corpus treebank before they are arithmetically combined.

This is a potentially useful strategy as it allows flexible and fast tuning of parsing models with the possibility of improving performance in a number of cases. Using this strategy, we might as a starting point create a combined model by setting the weighting of each corpus in proportion to the number of sentences in the treebank from which the model was trained. (We may expect to produce somewhat similar results to the aforementioned strategy of training a model from the concatenated treebanks, but there is no guarantee that this is the case.) However, often we might expect better results by biasing this weighting somewhat – in general, we probably wish to give more weighting to the model trained on a more similar treebank. It is clear that this could be useful in the situation where we have a small treebank and trained model in a new domain that we wish to use most effectively, alongside an existing model from a larger treebank – by modifying the weightings, and probably boosting the weighting for the corpus in the new domain, we have a way to make maximal use of this small amount of training data, without needing to discard anything from the larger established treebank. A related situation is where we have two models trained from treebanks in two domains, and wish to parse a third domain for which we have no treebanked data. Intuitively, we should make use of all the data we have, but it may improve performance if we give a higher weighting to the domain that is more "similar" to the target domain – i.e. providing a higher accuracy parse selection model. McClosky et al. (2010) shows some *a priori* ways to determine this for a different parsing framework. We do not investigate this in detail here, but as described below, we do examine some techniques to optimise the parameters when we know that the target domain very closely matches some small training corpus.

4.1.5. *Monolithic models from duplicated data*

We also investigate another method for weighting two training corpora differently when they have different sizes and probably different levels of appropriateness for the test data. In strategy DUPLIC, we simply duplicate one of the corpora some number of times, as if the training corpus consisted of multiple copies of every sentence, then concatenate the data to the other corpus and extract training features and train a model in the same way. As noted in Section 4.1.1 we pay attention to certain 'counts' associated with each maximum entropy feature – in particular the relevance count threshold, for how many times a feature has been used to distinguish between good and bad parses. These relevance counts are incremented in the duplicate corpora as if the sentences are genuinely distinct. This is obviously a generalisation of CONCAT, but we treat it separately, as CONCAT is the usual default approach for combining such corpora.

In comparison to COMBIN, this is a more expensive approach to optimise over different corpus combination parameters, since we need to retrain the maximum entropy model (and, currently, recreate the feature cache) for each different weighting of the corpora, rather than building at most two models and performing a rapid linear combination. Nonetheless, this might lead to a more accurate model than COMBIN, justifying the extra cost, although in a system which is trying to optimise between several different corpora (rather than just two as we are doing here), this extra cost may be prohibitive. If the parameters are known in advance however, and we are merely trying to build a model, the difference in training time between the two approaches is minimal.

4.1.6. *Optimising combination parameters*

One of the potential disadvantages of the more nuanced combination strategies is the need to determine effective parameters. Even if COMBIN or DUPLIC can provide superior results with some set of parameters, this does not address the question of determining the optimal set of parameters without evaluating over the test data (which would not be realistic for a real-world application where the gold-standard annotations presumably don't exist). In the same way that we can tune the grid-search parameters using cross-validation over the training data, we can also use a similar approach to optimise the weights or multiplication factors for each domain. Specifically, we can use our small in-domain training set, divide that corpus into n cross-validation folds, and combine the training data from each fold with the complete data-set from the larger out-of-domain corpus using COMBIN or DUPLIC with various sets of parameters, then test over the test fold.

For 8-fold cross-validation, which we use here, this means we must train and test 8 models per set of parameters. For 7 different DUPLIC and 10 different COMBIN parameters, this means 136 different models are required per test-run. As in the grid-search discussed above, we rerank rather than reparse these sentences.

By aggregating these test fold results, we can select the optimal parameters. For tractability, we only calculate Acc_1 (using reranking of the parse forest) for the cross-validation results, and simply select the highest accuracy combination as 'optimal'. A more sophisticated approach could also take into account factors such as consistency, as measured by a low variance across the cross-validation folds, and also pay attention to the other scoring metrics.

Of course, we cannot guarantee that the parameters will indeed be optimal over the test data. For this reason, as a post hoc analysis, we evaluate the various parameter combinations both using cross-validation and over the test data, to determine how close the best performer in cross-validation is to the optimal parameter values for unseen data.

4.1.7. *Software Tools*

To perform the tasks, including grid search, optimisation over COMBIN and DUPLIC parameters described in Section 4.1.6, and reranking of existing parse forests against a new model, we customised a C++ implementation of the feature extraction code and model training code, known as MUPS (duplicating functionality of

Table V. Results using different metrics (exact match on highest-ranked tree [Acc_1], exact match within top-10 [Acc_{10}], and Precision, Recall and F-score over EDM_{NA} and F-score on EDM_{all}), for each of the test corpora, training on pure WeScience or pure LOGON data, or for a baseline randomly selected from the top-500 parses ranked according to a combined WeScience/LOGON model. As in all results, WeSc and LOG test data is 2 sections of held-out data not overlapping with training data. Bold indicates the training data has the same domain as the test data. Figures in square brackets are standard deviations, calculated over 10 random partitions of the data

Test	Train	Acc_1	Acc_{10}	EDM_{NA}
				P / R / F
LOG	–	13.7	37.5	79.2 / 75.7 / 77.4
	WeSc	36.7[4.5]	66.7[3.1]	86.2[0.9] / 83.7[1.1] / 84.9[0.6]
	LOG	52.5[2.6]	77.9[3.1]	91.6[0.7] / 90.0[1.2] / 90.8[0.7]
WeSc	–	13.3	35.3	79.6 / 76.0 / 77.7
	WeSc	44.1[2.9]	73.7[3.6]	89.2[1.1] / 86.5[2.1] / 87.9[1.5]
	LOG	33.6[4.4]	62.4[3.7]	84.3[0.8] / 80.8[1.7] / 82.5[1.2]
C&B	–	7.8	24.5	80.5 / 76.5 / 78.4
	WeSc	27.2[6.0]	61.9[5.0]	88.3[1.2] / 84.5[2.8] / 86.4[1.7]
	LOG	27.7[3.7]	57.7[7.4]	87.2[0.9] / 82.9[2.9] / 85.0[1.7]
robot1	–	28.0	69.2	74.0 / 66.1 / 69.8
	WeSc	43.4[6.7]	86.4[3.3]	82.3[3.0] / 75.3[5.2] / 78.6[4.1]
	LOG	45.6[5.7]	89.2[2.3]	82.6[3.9] / 70.1[4.4] / 75.9[3.7]

[incr tsdb()] in some cases). We plan to make this code available to the community as part of the previously-mentioned DELPH-IN code distribution.

5. Results

5.1. Evaluating the cross-domain performance penalty

In Table V we give an indication of the size of cross-domain performance penalty using the ERG for these four domains, against the baseline performance for reference (giving us an idea of how difficult the parse selection problem is for that corpus). It shows results using several metrics: exact match on the highest-ranked parse, exact match within the top 10, and precision, recall and F-score for EDM_{all} and EDM_{NA}.

Of primary interest here is how the performance over WeScience data drops when the training data is purely LOGON versus purely in-domain WeScience data, and vice versa. A related question is whether for a given new target domain (e.g. C&B or ROBOT1) we would expect each alternative training corpus to give equally useful accuracy figures. So, the results shown are over all test corpora using all WeScience data or all LOGON data as training data.

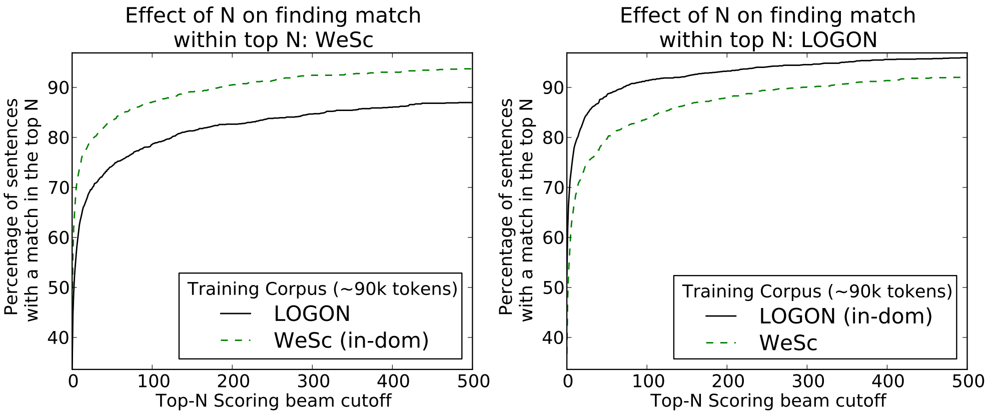
We see here that in our two major test sets, a domain penalty can be seen across all evaluation metrics. For EDM_{NA}, which is the most comparable to dependency evaluations in other formalisms, the out-of-domain drop is 5–6%, which is in line with domain effects seen elsewhere. For the Acc_1 and Acc_{10} figures, there is a larger 10.5–15.8% drop, which is not unexpected for a stricter metric.

The results also show standard deviations, calculated by comparing results from 10 random sub-partitions of the test data. As we would expect, the standard deviation is larger for the exact match metric, where it ranges from 2.5 to 7%, than for the less-variable EDM F-score, where it is between 0.6 and 4.1%. We also see much greater variation for all metrics on the ROBOT1 corpus, and smaller standard deviation for exact match when the training corpus is from the same domain, although this does not hold for other metrics, and may not be meaningful. For the two test corpora where in-domain data is available, we see that the difference between accuracies from in-domain and out-of-domain training data is substantially larger than the standard deviations, but for the other two test corpora the differences caused by using the different training corpora are much smaller than the standard deviation for a single model, so the difference is too small to draw conclusions.

Comparing these results with the relative entropy figures in Figure 2, there is generally a correlation between a lower relative entropy and higher parse selection score using the corresponding model. The WESCIENCE and LOGON test corpora unsurprisingly show this effect most strongly, where there are the largest relative entropy differences against one training corpus versus the other (close to zero for the in-domain corpus). Both C&B and ROBOT1 generally show differences according to the metrics which agree with the differences in relative entropy against the training corpora (lower relative entropy corresponding to higher scores), but as we noted, these differences are not large enough to be clearly significant.

In Section 3.2, we argued that Acc_1 and Acc_{10} are easy-to-calculate representatives for a range of figures denoting treebanking utility. To illustrate this, in Figure 6 we show the Acc_N that would be obtained for values of N from one to 500 for some of the same in-domain and cross-domain training/test combinations that are shown in Table V. The full graphs more directly show the effects of interest for treebanking, but they are expensive to create. Comparing between these graphs and Table V, it appears that Acc_1 and Acc_{10} are representing some of this picture. In Figure 6a, the two parse selection models for the WeScience data set result in a fairly consistent difference of around 11%, a consistency that can be seen just from the Acc_1 and Acc_{10} values in the table.[⋆] For Figure 6b, the gap narrows as the beam widens, a fact again reflected by comparing Acc_1 and Acc_{10} for the relevant

[⋆] The WESC treebank is parsed and treebanked against the top-500 parses according to a model trained on the WESC data itself, so we might expect the accuracy to reach 100% in Figure 6a, but the



(a) WESCIENCE (b) LOGON

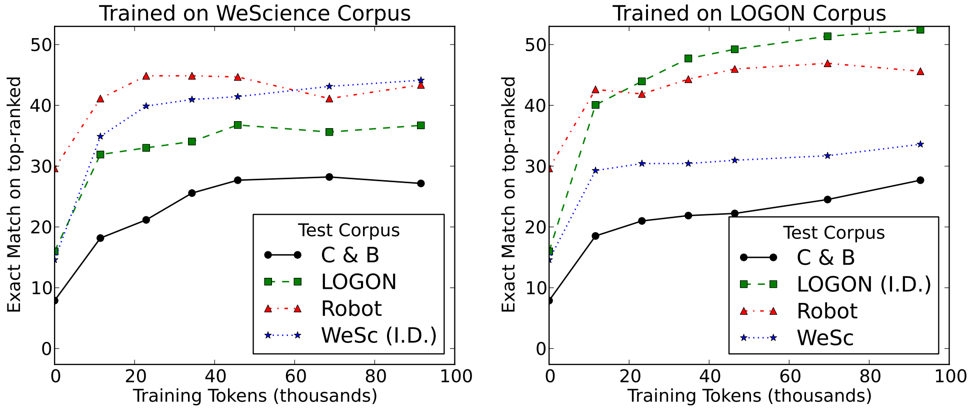
Figure 6. Accuracy for gold-standard parse occurring within the top-N (i.e. Acc_N) against N

LOGON data set results. This narrowing gap shows that the impact of in- vs. out-of-domain training data is reduced when we are looking at more parses – probably because the less ambiguous sentences of LOGON are more likely to have less than N possible parses for higher values of N (i.e. all parses are within the beam), in which case the parse selection model does not matter.

For the test sets for which we have no in-domain training data, there is little difference between the two parse selection models. They do, however, provide an interesting comparison between the exact match based metrics and the F-scores. In terms of EDM, the WESC trained model gives almost the same results over C&B (within 2%) as it does for the in-domain test set; however, the Acc_1 results are much lower. For the ROBOT1 data set, the EDM results are substantially lower than on any of the other test sets, but Acc_1 and Acc_{10} are high. To partially explain this, we can look back to Table II. We saw there that the ambiguity level of the ROBOT1 corpus, measured in parses per sentence, was much lower than that of the other data sets. This simplifies the parse selection task, since there are fewer analyses to consider. Conversely, the longer and more ambiguous sentences in the C&B corpus make it much more difficult to get every aspect of an analysis right. The relatively high F-scores for this corpus suggest that both parse selection models are doing a good job of returning a top parse with most of the dependencies and constituents correct. The different behaviours of the Acc_N and EDM metrics on different corpora show why we should consider both together in selecting the most appropriate parse selection model for a particular corpus. For the rest of this work, we primarily report results on exact match (Acc_1) and EDM_{NA}, with brief results for Acc_{10} where appropriate.

model we use here does not use the complete WeScience data set and has slightly different training parameters.

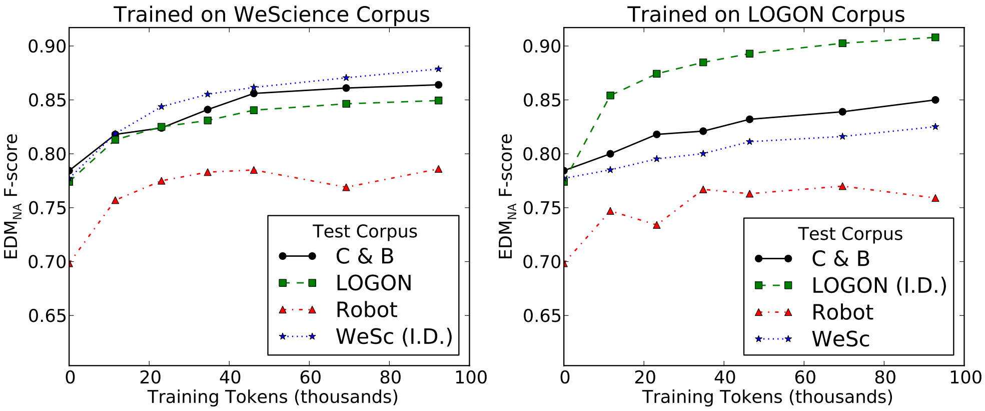
The results in Table V were produced by using all the available training data for each domain. We were also interested in how the results changed with the amount of training data and so learning curves were produced for the same training and test sets. The learning curves obtained using the two different training corpora of approximately 8000 sentences each are shown in Figures 7 and 8 using the exact match metric as well as EDM F-score over four different test-domains. In each case, the in-domain corpus is marked as such.



(a) WESC

(b) LOG

Figure 7. Learning Curves – exact match. ‘I.D.’ denotes in-domain corpus



(a) WESC

(b) LOG

Figure 8. Learning Curves – $\mathrm{EDM}_{\mathrm{NA}}$ F-score

The basic shape of the curves is unsurprising. Generally, the curves are monotonically increasing, so more training data of any type produces better results. This effect continues even when relatively large amounts of training data were already being used, but as we would expect, there is some flattening off in these curves, as the more training data we have, the less incrementally valuable it is. This levelling off is more noticeable when the training data is entirely out-of-domain – suggesting that there is a limit to the amount of out-of-domain data which can usefully improve parsing performance, at least in isolation (i.e. when not in combination with in-domain data, which is discussed in more detail below). Indeed, it is possible that too much (solely) out-of-domain data could have a detrimental effect. There are small drops in some of the learning curves as out-of-domain data is added, particularly in the Acc_1 evaluation, for C&B, ROBOT1 and LOGON at various points, although this is at most a very minor effect.

Again, we can clearly see that Acc_1 and EDM give a different picture of performance on the C&B and ROBOT1 corpora. Comparing these figures tells us something else about the different metrics: in some situations, Acc_1 may be more useful in differentiating the success of a given model on multiple domains. The exact match metric shows more noticeable relative changes than EDM when we make subtle changes in the model and the domain, emphasising the importance of domain for treebanking or other applications where we demand an exact tree match. The EDM results cluster more closely in absolute terms for most data sets regardless of the amount of training data or the domain, but there are still reliable, albeit small, changes as the amount of training data is altered. This follows from the more 'forgiving' nature of the EDM-based evaluation, but it also tells us something about the grammar: given a very small amount of training data from any domain, the top-ranked parse will have most of the dependencies correct.

For maximum sensitivity in parse selection experiments (as well as tractability in experiments with many successive runs such as grid searches), we would argue that the exact match metric is undoubtedly useful, and provides a complementary perspective to EDM.

If the EDM metric, as intended, more closely reflects the performance we could expect in downstream applications, it may appear that these are more robust to changes in domain. However, it is possible that for these applications using the parser output, it is the error rate which is more important. From this perspective it seems EDM can be more sensitive to choice of training domain than exact match. From Table V, we can see that over WESCIENCE, for Acc_1, the error rate goes from 66.4% to 55.9%, a 16% relative reduction, when moving from out-of-domain to in-domain training data, while the relative reduction in EDM F-score error rate (from 17.5% to 12.1%) is 31%. Similarly for LOGON by using in-domain data, we get a 25% relative error rate reduction for Acc_1, and 39% for EDM.

It is also instructive to compare error rate reduction relative to the random baseline (although it is not truly random as it incorporates the top-500 parses according to a model which has training data from WESCIENCE and LOGON). For EDM the relative reduction from using out-of-domain data is 21% for WESCIENCE, and 33% for LOGON. This is smaller than the reduction in error rate when we move from out-of-domain to in-domain data (31% and 39% respectively, as quoted above), suggesting that a tuned parse-selection model is quite important – it can give more of a performance boost over an informed but unmatched training corpus than that unmatched corpus does over a random selection. However, further experimentation would be required to determine whether the error rate reduction is more meaningful in terms of downstream utility.

Additionally, it seems that not all corpora are equal in terms of cross-domain applicability. From Figures 7 and 8, we can see that WeScience as the only training corpus gives slightly better results for C&B (apart from the slightly higher $\mathrm{Acc}_1$from the full 93000 token training sets when using LOGON), as we would predict from the relative entropy figures. On the other hand, the best training corpus for ROBOT1 is less obvious. Indeed, it seems that training data beyond the first 11000 tokens does very little, and sometimes decreases performance. LOGON gives slightly higher exact match performance, although the figures are so variable that the differences may not be meaningful. In general, training data does little for ROBOT1 probably due to the very different nature of the corpus compared to the data we have available, with the smallest absolute improvements and error rate reductions over the baseline of any of the test corpora.

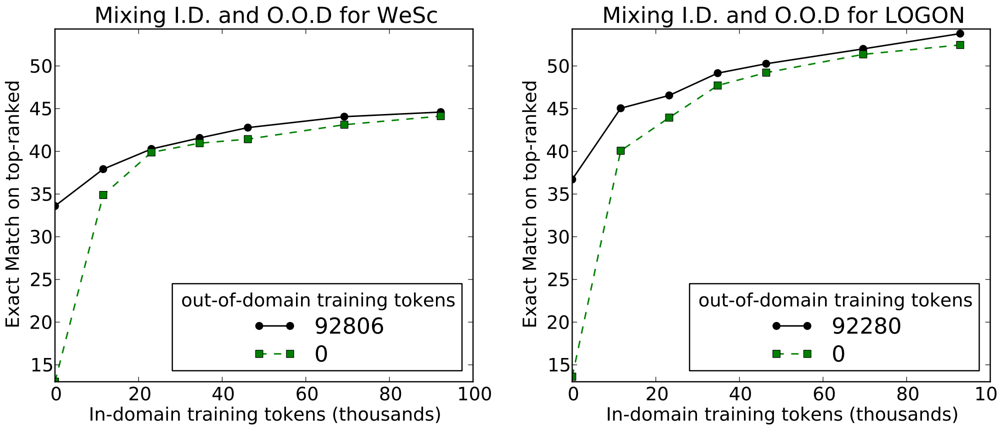
5.2. CONCAT: NAIVE CONCATENATION

Having measured how much our performance is affected by using only out-of-domain data, we now look at the results from the simple concatenation of the two corpora of training data (CONCAT). There are likely to be two significant factors here – the amount of in-domain data, and the amount of out-of-domain data.

One common scenario might be that we have a fixed volume of training data, and wish to know how much in-domain data we need to treebank to substantially improve performance when combining it using CONCAT with the out-of-domain data, or alternatively how much improvement we can expect from some volume of in-domain data. Secondarily, it is interesting to investigate whether it is possible to add too much out-of-domain data compared to in-domain – is there ever a situation where more data does not improve performance?

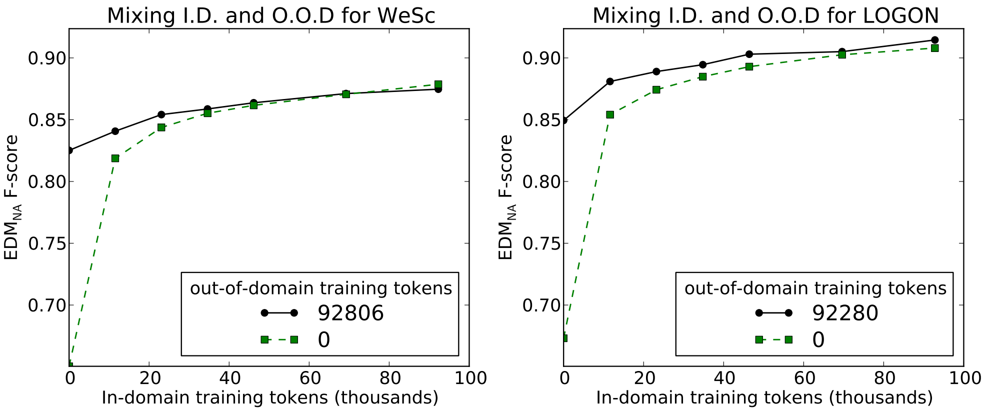
In Figures 9 and 10 we show some indicative results for these questions, using no out-of-domain data or all of it, and evaluate how they interact with different-sized in-domain training corpora.

One interesting result from these figures is the effect that even a small amount of in-domain training data can have. Using a model built only on approximately 11000 tokens on in-domain data, we get better results than the large out-of-domain trained model. Over the WESCIENCE corpus, once we have around 23000 tokens of in-domain data, the out-of-domain data is having almost no effect. There does not appear to be a negative effect from including the out-of-domain data, but the benefit is almost non-existent for WESCIENCE and very slight for the LOGON corpus. We also see that the additional benefit of adding more in-domain data tails off once we have a reasonable quantity (i.e. a few thousand sentences), which is a similar effect to the learning curves.



(a) WESCIENCE (b) LOGON

Figure 9. Combining in-domain and out-of-domain training data using CONCAT: training a model from concatenated training corpora: Exact match

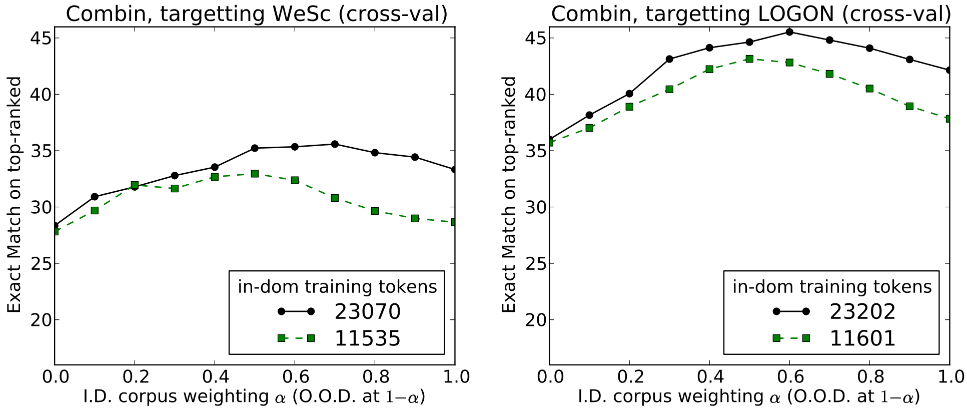


(a) WESCIENCE (b) LOGON

Figure 10. Combining in-domain and out-of-domain training data using CONCAT: training a model from concatenated training corpora: EDM

5.3. OPTIMISING PARAMETERS FOR CORPUS COMBINATION

While CONCAT is the most obvious strategy for making use of limited quantities of in-domain training data when we already have an abundant corpus of out-of-domain data, it is possible that some scheme which gives heavier weight to the in-domain training data would provide superior performance. The COMBIN strategy of Section 4.1.4 (linearly interpolating between two single-domain models) is one way to address this, and the DUPLIC strategy from Section 4.1.5 (multiplying the in-domain corpus by some integer) is another. However, as noted in Section 4.1.6, both strategies are parameterised, so ideally we need a method to



(a) WESCIENCE (b) LOGON

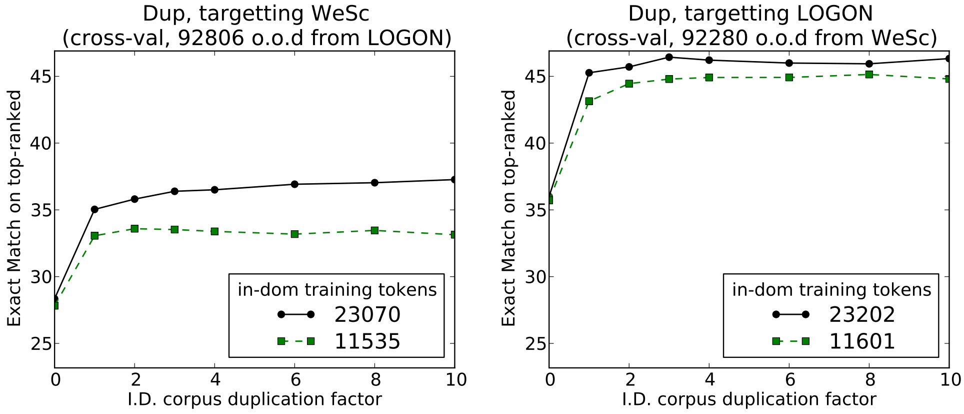
Figure 11. Aggregated accuracy using COMBIN over the eight cross-validation folds of the in-domain corpus, where the in-domain training data is the other seven folds and the out-of-domain data is the entire other corpus

optimise the respective parameters in advance for unseen data. Cross-validation over the in-domain training corpus may be able to achieve this with a relatively simple implementation, and we investigate this possibility here.

The graphs in Figures 11 show COMBIN results for two different-sized in-domain models, and how the performance varies as the weighting between the in- and out-of-domain models is varied, with exact match accuracy calculated by reranking rather than exhaustively reparsing. Clearly, the choice of weighting makes a relatively large difference: the worst performing weight combination is substantially worse than simply using the CONCAT strategy. Both curves show a similar parabolic shape, with the optimal mixture point being further to the right for the larger in-domain models in each case, meaning that it is better to weight the in-domain model even more heavily when it is larger, presumably because it is more reliable (compared to a model from a smaller treebank) as well as being closely matched to the domain.

Figure 12 shows the results for DUPLIC, which duplicates the smaller in-domain corpus some integral number of times and combines it with the larger out-of-domain corpus before training a model. Again, we show graphs for two different sized in-domain training sets, this time varying how many times the in-domain set was duplicated before training the model.

For handling real-world data with an unbalanced in-domain corpus, we might generally pick the best-performing parameter set from cross-validation and apply that parameter combination to new data. We evaluate this approach over the test data from previous sections (which has not been used for any of the parameter tuning). In Table VI, we show the results over unseen test data, using the parameters which performed best in training set cross-validation using DUPLIC (results for



(a) WESCIENCE

(b) LOGON

Figure 12. Aggregated accuracy using DUPLIC over the eight cross-validation folds of the in-domain corpus, where the in-domain training data is the other seven folds and the out-of-domain data is the entire other corpus

COMBIN did not reliably outperform the CONCAT benchmark). For another point of comparison, we also applied the same technique using cross-validation over the ROBOT1 development set (which has not yet been used until this point) combined with different combinations of the WESCIENCE and LOGON training sets and tested using the same test data as used in previous sections.

We calculate statistical significance using the "compute-intensive" shuffling procedure of Yeh (2000). In a given iteration, for all results which differ between the new method and the benchmark, we swap each result pair with probability 0.5 and test whether the new synthesised results differ by more than was observed between the actual results of the new method and the benchmark (which would suggest the difference is due to chance and the null hypothesis could be true), incrementing count c if this is the case. Repeating for some large number of iterations t, the p-value can be estimated to be at most $\frac{c+1}{t+1}$.

Our end-to-end procedure selects a parameter with cross-validation, and uses that parameter to build a model to apply to the test corpus. To test whether this technique gave a consistently significant boost over the benchmark, we aggregate all benchmark results for the different test and training corpus combinations, and all those to which we have applied parameter selection over DUPLIC. This aggregation avoids multiple runs (using different training and/or test data) falsely showing significance by chance. DUPLIC in these conditions produces statistically significant improvements over the benchmark on unseen data for Acc_1, Acc_{10} and EDM_{NA} at $p < 0.001$ – even though the improvements are modest in some individual cases, as we can see from the unaggregated results shown in Table VI. COMBIN (not shown in the table) does not provide an improvement – all the metrics decrease, Acc_1 significantly so. But if we restrict ourselves to DUPLIC, it seems that this

Table VI. Comparison of results for Duplic, with weighting parameter selected by cross-validation over training data (the first usage of the robot1 training data), and evaluated over standard test sets and metrics (Acc$_1$, Acc$_{10}$and EDM$_{\mathrm{NA}}$) used previously. Aggregating the Duplic results with CV-parameter selection indicates results are significantly improved over the benchmark at $p < 0.0001$ for Acc$_1$, Acc$_{10}$and EDM$_{\mathrm{NA}}$

Test	Train Tokens		O.O.D Corp		Weights	A$_1$	A$_{10}$	EDM
	I.D.	O.O.D						
LOG	11.6k	92.3k	WeSc	benchmark		45.0	72.4	88.1
				Duplic	(8, 1)	45.9	73.5	88.7
LOG	23.2k	92.3k	WeSc	benchmark		46.5	73.9	88.9
				Duplic	(3, 1)	47.7	74.6	89.1
robot1	4.5k	92.8k	LOG	benchmark		74.0	93.5	89.0
				Duplic	(3, 1)	75.1	93.6	88.8
robot1	4.5k	92.3k	WeSc	benchmark		75.1	93.3	88.8
				Duplic	(4, 1)	75.5	93.3	89.2
robot1	4.5k	92.8k	W+L	benchmark		75.5	93.1	89.1
				Duplic	(10, 1)	77.8	93.5	90.0
WeSc	11.5k	92.8k	LOG	benchmark		37.9	66.9	84.1
				Duplic	(2, 1)	38.5	67.8	84.3
WeSc	23.1k	92.8k	LOG	benchmark		40.3	69.0	85.4
				Duplic	(10, 1)	42.0	71.3	86.3

simplistic and imperfect cross-validation technique to tune parameters can produce statistically significant improvements in accuracy and F-score at the cost of only some CPU-cycles, with no observed performance drops against the benchmark. For 23000 tokens of WeScience, we get a relative reduction in the exact match error rate of 2.9% and 6.1% in the EDM F-score error rate, almost as much as we would get from treebanking an extra 23000 tokens.

All of this is dependent on being able to closely match the test and training corpora – we must be sure they are from the same domain, which may not be as easy in practice as it is with curated data sets. The 'shuffling' we used to divide up the corpora may mean that the match between the different corpus sections is artificially close compared to what we would see in real-world data, so this could have slightly inflated the performance improvements from upweighting the in-domain data and parameter-selection using the training corpus. The relative entropy comparison we discussed in Section 3.1.2 suggests one possible strategy for matching training and test corpora by similarity automatically, and although the comparison of lexical rules depended on hand-annotated gold standard data, the best parse from an automatically-created parse tree may be a reasonable substitute. Relatedly, we might be able to apply a more informed approach along the lines of McClosky et al. (2010), which suggests a model on the basis of how closely a test corpus is predicted to match each possible training corpus.

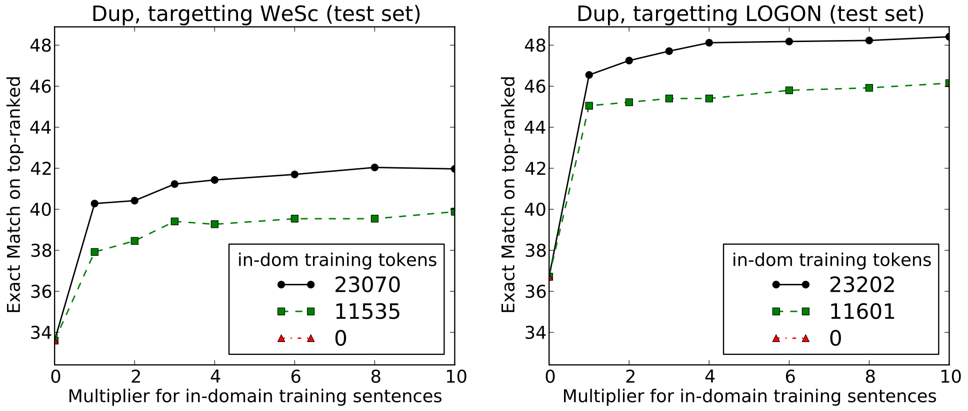
5.4. APPLYING OTHER PARAMETERS TO TEST DATA

We have established that cross-validation over training data to select parameters can give slightly improved parse selection models, but we have no guarantee that the parameters we obtained were optimal. As an analysis phase, it is instructive to evaluate a range of parameters over the test data, to see what the oracle performance would be, and whether a more advanced parameter tuning technique could improve over the relatively simple cross-validation approach, and to simply see how well the cross-validation scores correlate with test scores.

We show the test set results using over a range of DUPLIC parameters in Figures 13, the analogue of Figure 12, and the EDM test set results in Figure 14. From these, we can see that in some cases cross-validation provided a reasonable estimate of test set performance and the optimal parameter. Over the smaller set of WeScience training data, however, it was not helpful, performing best in cross-validation with a $(2, 1)$ weighting for in-domain against out-of-domain, while over the test set, a $(10, 1)$ weighting is substantially better, for both EDM and exact match. We have again omitted the COMBIN results as the performance did not reliably improve.

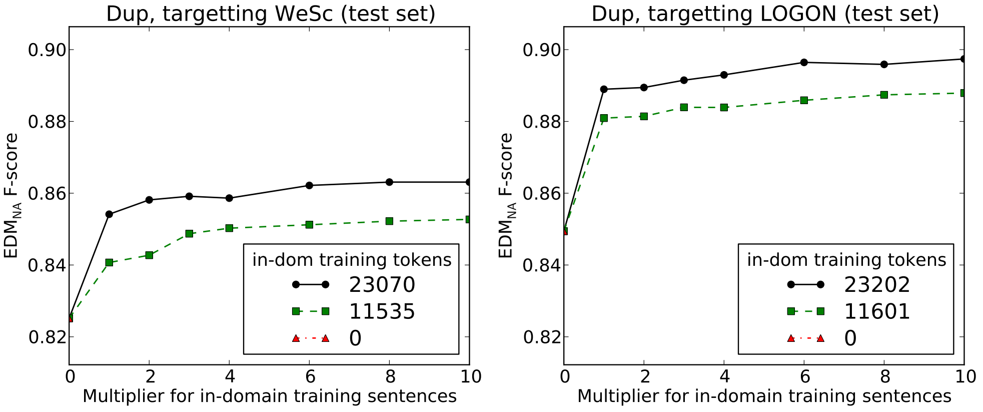
These results indicate that DUPLIC is tolerant of suboptimal parameter selection (this is not the case for COMBIN– the worst parameters produce substantially worse performance than CONCAT). The exact match accuracy over the full corpus usually increases monotonically, so it is almost always of some benefit to weight the smaller in-domain corpus more heavily (although, presumably, at some point these benefits vanish). The same is generally true for EDM, although the increase is less convincing in this case.

While the cross-validation results seem to provide a reasonable estimator of performance over unseen test data, we can evaluate this more rigorously by measuring the Pearson correlation between the cross-validation results for Acc_1 and the results using the same parameters on the held-out test data for both Acc_1 and EDM. This figure is usually 0.99 or more for DUPLIC, with the only exception being for WESCIENCE with 769 sentences of training data, which gives 0.95 for Acc_1 and 0.88 for EDM. Thus, the cross-validation estimates are fairly predictive of the results we can expect over held-out data in the same closely-matched domain using DUPLIC. This suggests that selecting the best parameter combination from cross-validation is a potentially useful strategy, at least in terms of predicting relative changes in accuracy over unseen data. However, as noted above, this correspondence is almost certainly helped by our data selection method of using random corpus partitions causing a close match between the training and test corpora. Additionally, this



(a) WeScience (b) LOGON

Figure 13. Exact match scores for Duplic: duplicating the in-domain data set multiple times



(a) WeScience (b) LOGON

Figure 14. EDM_{NA} F-scores for Duplic: duplicating the in-domain data set multiple times

parameter selection does not always produce optimal results over new data, so there is some room for improvement. On the basis of the test data we have looked at here, it seems that a weighting around $(8, 1)$ would be a robust performer, so the cross-validation parameter selection may not be necessary — although it may not be as applicable for all data.

6. Discussion

6.1. DOMAIN ADAPTATION STRATEGIES

One of the important aspects of this work to other users of precision grammars is to suggest a strategy for achieving optimal performance over data in some new domain, and make a decision about how much effort to expend treebanking, and how best to make use of treebanked data. On the basis of this work, we would make the following recommendations:

- Unsurprisingly, out-of-domain data is much better than no data. If there are insufficient resources to treebank *any* in-domain data, you can expect tolerable parsing accuracy from using only out-of-domain data – the domain adaptation performance penalty does not make the outputs from the ERG unusable (and we might expect this to hold for similarly constructed grammars).
- However, the effort required to treebank around 11000 tokens (750–850 sentences for the corpora here) gives substantial gains in accuracy compared to the benchmark of using only out-of-domain data – these 750 sentences are extremely valuable. The time requirements for this are modest: using Redwoods machinery, Zhang and Kordoni (2010) found that it was possible to treebank a curated WSJ subset at 60 sentences per hour, while Tanaka et al. (2005) found that Japanese speakers could treebank 50 sentences of Japanese dictionary definitions per hour. So even with a conservative figure of 40 sentences per hour, 750 sentences would be under 20 hours of annotation time.
- Even simply concatenating 11000 tokens of in-domain data to existing training data gives a good performance boost, but by applying the optimisation strategy using cross-validation we have discussed for DUPLIC, it is possible in some corpora to obtain accuracies close to those you would expect if you had 11000 more training tokens. Without performing the optimisation step, upweighting the in-domain corpus by a factor of 5-10 provides near-optimal performance across the corpora we examined.
- If you have resources to treebank 23000 tokens (roughly 1600 sentences) in total, you can achieve additional boosts in performance, although the value is considerably reduced.
- Beyond 23000 tokens of training data for some domain, the gains in accuracy per treebanked sentence are more modest, so the effort would only be justified for more difficult domains or if maximising accuracy is of utmost concern.

6.2. FUTURE WORK

We have presented only indicative results here for two domains which happen to be available and provide large quantities of training data. Finding out how broadly applicable they are and whether they extend to other domains leaves room for further research. Building a new custom treebank of perhaps 2000 sentences would be tractable and help answer that question. Additionally, self-training has shown small but solid improvements on both biomedical text (McClosky and Charniak,

2008) and Wikipedia (Honnibal et al., 2009) so could be a useful strategy to evaluate, given that it requires no expensive manual annotation. Work in this direction could also incorporate recent work on unsupervised parse selection (Dridan and Baldwin, 2010). Another interesting question is whether we can improve EDM scores, particularly recall, by training a model using semantic dependencies rather than syntactic constituents, and whether these improved scores would be reflected in downstream applications.

Further to this, we have not yet addressed the question of when in the treebanking process it is optimal to build the parse selection model for minimal overall time. A better parse selection model could possibly reduce treebanking time by requiring fewer trees to be examined – e.g. from Figure 6a, a domain-adapted model can give as many correct trees in the top 100 as a non-adapted model gives in the top 500. It could also increase the chance of having the correct tree in the parse forest, and this would reduce the need for rejecting trees, which is particularly expensive in the Redwoods treebanking process as it often requires multiple passes through the data (Tanaka et al., 2005). It is not clear how important this effect would be, but given this information as well as the time taken to build a model (which is on the order of a few CPU-hours, depending on training set size), we could work out the optimal point in the treebanking process to stop and train a new model to use for the remaining sentences in the corpus. In future work, we are interested in determining where this optimal point lies, based on exploration of the impact of parse selection on determinant selection, and in situ treebanking experiments. Our domain adaptation approach to improving annotation efficiency should be complementary to the active-learning strategy proposed by Baldridge and Osborne (2003) and Osborne and Baldridge (2004) for Redwoods-style treebank development, given their focus on identifying which sentences within a corpus will be most useful to annotate, rather than on reducing the cost of annotating any given sentence.

7. Conclusion

This paper examined the impact of domain on parse selection accuracy in the context of precision parsing, evaluated across exact match and dependency-based metrics. Our findings confirm our intuition that parse selection accuracy is significantly improved by in-domain training data, particularly if we are interested in returning a completely correct parse, and in-domain training data is considerably more valuable in terms of accuracy obtained from a given number of training sentences. Additionally, the construction of even small-scale in-domain treebanks, which is fairly tractable, can considerably improve parse selection accuracy, through combining the in-domain with out-of-domain data. We showed that linear combination of models from different domains can provide slightly improved performance compared to training from a monolithic concatenated corpus, although without careful selection of weights, it can also decrease. A better strategy for tuning a model to a domain with a small training corpus was to duplicate this small corpus some integral number of times. A multiplier of 5-10 often produces good results for the data we have shown here, but we have also shown that the optimal value for this parameter can be estimated on a case-by-case basis by using cross-validation over the training corpus, as the values are highly correlated. This finding is highly significant for both treebanking and downstream applications that use the parser output, and it suggests a useful strategy for grammar consumers to use when adapting to a novel domain.

8. Acknowledgments

We wish to thank the two anonymous reviewers for their valuable comments on earlier versions of this paper, and Stephan Oepen for his constructive comments and technical assistance. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. This research was supported in part by a VLSCI research allocation grant and by funding from Microsoft Research Asia.

References

Baldridge, J. and M. Osborne: 2003, 'Active Learning for HPSG Parse Selection'. In: *Proceedings of the Seventh Conference on Natural Language Learning*. Edmonton, Canada, pp. 17–24.

Bikel, D. M.: 2002, 'Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine'. In: *Proceedings of the Second International Conference on Human Language Technology Research*. San Francisco, USA, pp. 178–182.

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski: 1991, 'Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars'. In: *Proceedings of the Workshop on Speech and Natural Language*. Pacific Grove, USA, pp. 306–311.

Blitzer, J., R. McDonald, and F. Pereira: 2006, 'Domain Adaptation with Structural Correspondence Learning'. In: *Proceedings of EMNLP 2006*. Sydney, Australia, pp. 120–128.

Böhmová, A., J. Hajič, E. Hajičová, and B. Hladká: 2003, 'The Prague Dependency Treebank: A Three Level Annotation Scenario'. In: A. Abeillé (ed.): *Treebanks: building and using parsed corpora*. Springer.

Bouma, G., G. van Noord, and R. Malouf: 2001, 'Alpino. Wide-Coverage Computational Analysis of Dutch'. In: W. Daelemans, K. Sima-an, J. Veenstra, and J. Zavrel (eds.): *Computational Linguistics in the Netherlands*. Amsterdam, The Netherlands, pp. 45–59, Rodopi.

Brants, S., S. Dipper, S. Hansen, W. Lezius, and G. Smith: 2002, 'The TIGER treebank.'. In: *Proceedings of the First Workshop on Treebanks and Linguistic Theories*. Sozopol, Bulgaria.

Brants, T.: 2000, 'TnT - A Statistical Part-of-Speech Tagger'. In: *Proceedings of the 6th ACL Conference on Applied Natural Language Processing*. Seattle, USA, pp. 224–231.

Briscoe, T. and J. Carroll: 2006, 'Evaluating the Accuracy of an Unlexicalized Statistical Parser on the PARC DepBank'. In: *Proceedings of the COLING/ACL 2006 Poster Sessions*. Sydney, Australia, pp. 41–48.

Callmeier, U.: 2000, 'PET – A Platform for Experimentation with Efficient HPSG Processing Techniques'. *Natural Language Engineering* **6**(1), 99–107.

Carter, D.: 1997, 'The TreeBanker. A Tool for Supervised Training of Parsed Corpora'. In: *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*. Madrid, Spain, pp. 9–15.

Charniak, E.: 2000, 'A Maximum-entropy-inspired Parser'. In: *Proceedings of NAACL 2000*. Seattle, USA, pp. 132–139.

Charniak, E. and M. Johnson: 2005, 'Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking'. In: *Proceedings of ACL 2005*. Ann Arbor, USA, pp. 173–180.

Clark, S. and J. Curran: 2007a, 'Formalism-Independent Parser Evaluation with CCG and DepBank'. In: *Proceedings of ACL 2007*. Prague, Czech Republic, pp. 248–255.

Clark, S. and J. R. Curran: 2007b, 'Wide-coverage Efficient Statistical Parsing with CCG and Log-linear Models'. *Computational Linguistics* **33**(4), 493–552.

Clegg, A. and A. Shepherd: 2005, 'Evaluating and Integrating Treebank Parsers on a Biomedical Corpus'. In: *Proceedings of the ACL 2005 Workshop on Software*. Ann Arbor, USA, pp. 14–33.

Collins, M.: 1997, 'Three Generative, Lexicalised Models for Statistical Parsing'. In: *Proceedings of ACL 1997*. Madrid, Spain, pp. 16–23.

Collins, M.: 1999, 'Head-Driven Statistical Models for Natural Language Parsing'. Ph.D. thesis, University of Pennsylvania.

Copestake, A. and D. Flickinger: 2000, 'An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG'. In: *International Conference on Language Resources and Evaluation*.

Copestake, A., D. Flickinger, I. A. Sag, and C. Pollard: 2005, 'Minimal Recursion Semantics: An Introduction'. *Research on Language and Computation* pp. 281–332.

Dridan, R.: 2009, 'Using Lexical Statistics to Improve HPSG Parsing'. Ph.D. thesis, Saarland University.

Dridan, R. and T. Baldwin: 2010, 'Unsupervised Parse Selection for HPSG'. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. Boston, USA, pp. 694–704.

Finkel, J. R. and C. D. Manning: 2009, 'Hierarchical Bayesian Domain Adaptation'. In: *Proceedings of HLT-NAACL 2009*. Boulder, USA, pp. 602–610.

Flickinger, D.: 2000, 'On Building a More Efficient Grammar by Exploiting Types'. *Natural Language Engineering* **6 (1)**, 15–28.

Flickinger, D.: 2011, 'Accuracy vs. Robustness in Grammar Engineering'. In: E. M. Bender and J. E. Arnold (eds.): *Language from a Cognitive Perspective: Grammar Usage, and Processing*. Stanford, pp. 31–50, CSLI Publications.

Flickinger, D., R. Bhayani, and S. Peters: 2009, 'Sentence boundary detection in spoken dialogue'. Technical report, Stanford University, TR-09-06.CSLI.

Gildea, D.: 2001, 'Corpus Variation and Parser Performance'. In: *Proceedings of EMNLP 2001*. Pittsburgh, USA, pp. 167–202.

Hara, T., Y. Miyao, and J. Tsujii: 2005, 'Adapting a Probabilistic Disambiguation Model of an HPSG Parser to a New Domain'. In: *Proceedings of IJCNLP 2005*. Jeju Island, Korea, pp. 99–210.

Hara, T., Y. Miyao, and J. Tsujii: 2007, 'Evaluating Impact of Re-training a Lexical Disambiguation Model on Domain Adaptation of an HPSG Parser'. In: *Proceedings of IWPT '07*. Prague, Czech Republic, pp. 11–22.

Honnibal, M., J. Nothman, and J. R. Curran: 2009, 'Evaluating a Statistical CCG Parser on Wikipedia'. In: *People's Web '09: Proceedings of the 2009 Workshop on The People's Web Meets NLP*. Singapore, pp. 38–41.

Kaplan, R., S. Riezler, T. H. King, J. T. Maxwell III, A. Vasserman, and R. Crouch: 2004, 'Speed and Accuracy in Shallow and Deep Stochastic Parsing'. In: *Proceedings of HLT-NAACL 2004*. Boston, USA, pp. 97–104.

Kingsbury, P., M. Palmer, and M. Marcus: 2002, 'Adding Semantic Annotation to the Penn TreeBank'. In: *Proceedings of the Human Language Technology 2002 Conference*. San Diego, USA, pp. 252–256.

Lease, M. and E. Charniak: 2005, 'Parsing Biomedical Literature'. In: *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*. Jeju Island, Korea, pp. 58–69.

Malouf, R.: 2002, 'A Comparison of Algorithms for Maximum Entropy Parameter Estimation'. In: *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*. Taipei, Taiwan, pp. 49–55.

Malouf, R. and G. Van Noord: 2004, 'Wide Coverage Parsing with Stochastic Attribute Value Grammars'. In: *Proceedings of the IJCNLP-04 Workshop: Beyond Shallow Analyses — Formalisms and Statistical Modeling for Deep Analyses*. Hainan, China.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz: 1993, 'Building a Large Annotated Corpus of English. The Penn Treebank'. *Computational Linguistics* **19**, 313–330.

McClosky, D. and E. Charniak: 2008, 'Self-training for Biomedical Parsing'. In: *Proceedings of ACL-08 HLT: Short Papers*. Columbus, USA, pp. 101–104.

McClosky, D., E. Charniak, and M. Johnson: 2006, 'Reranking and Self-Training for Parser Adaptation'. In: *Proceedings of the COLING/ACL 2006*. Sydney, Australia, pp. 337–344.

McClosky, D., E. Charniak, and M. Johnson: 2010, 'Automatic Domain Adaptation for Parsing'. In: *Proceedings of HLT-NAACL 2010*. Los Angeles, USA, pp. 28–36.

Miyao, Y., K. Sagae, and J. Tsujii: 2007, 'Towards Framework-Independent Evaluation of Deep Linguistic Parsers'. In: *Proceedings of the GEAF 2007 Workshop*. Palo Alto, USA.

Miyao, Y. and J. Tsujii: 2008, 'Feature Forest Models for Probabilistic HPSG Parsing'. *Computational Linguistics* **34**(1), 35–80.

Oepen, S.: 2001, '[incr tsdb()] — Competence and Performance Laboratory. User Manual'. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Oepen, S. and J. Carroll: 2000, 'Ambiguity Packing in Constraint-based Parsing – Practical Results'. In: *Proceedings of NAACL 2000*. Seattle, USA, pp. 162–169.

Oepen, S., D. Flickinger, K. Toutanova, and C. D. Manning: 2004, 'LinGO Redwoods: A Rich and Dynamic Treebank for HPSG'. *Research on Language and Computation* **2**(4), 575–596.

Oepen, S. and J. T. Lønning: 2006, 'Discriminant-Based MRS Banking'. In: *Proceedings the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*. Genoa, Italy, pp. 1250–1255.

Oepen, S., K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants: 2002, 'The LinGO Redwoods Treebank: Motivation and Preliminary Applications'. In: *Proceedings of the 19th International Conference on Computational Linguistics – Volume 2*. pp. 1–5.

Ohta, T., Y. Tateisi, and J.-D. Kim: 2002, 'The GENIA Corpus: An Annotated Research Abstract Corpus in Molecular Biology Domain'. In: *Proceedings of the Second International Conference on Human Language Technology Research*. San Francisco, USA, pp. 82–86.

Osborne, M. and J. Baldridge: 2004, 'Ensemble-based Active Learning for Parse Selection'. In: *Proceedings of HLT-NAACL 2004: Main Proceedings*. Boston, USA, pp. 89–96.

Plank, B. and G. van Noord: 2008, 'Exploring An Auxiliary Distribution based approach to Domain Adaptation of a Syntactic Disambiguation Model'. In: *Proceedings of the COLING 2008 Workshop on Cross Framework and Cross Domain Parser Evaluation*. Manchester, UK.

Pyysalo, S., F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski: 2007, 'BioInfer: A Corpus for Information Extraction in the Biomedical Domain'. *BMC Bioinformatics* **8**(1), 50.

Rayson, P. and R. Garside: 2000, 'Comparing Corpora using Frequency Profiling'. In: *The Workshop on Comparing Corpora*. Hong Kong, China, pp. 1–6, Association for Computational Linguistics.

Rimell, L. and S. Clark: 2009, 'Porting a Lexicalized-grammar Parser to the Biomedical Domain'. *Biomedical Informatics* **42**(5), 852 – 865.

Roark, B. and M. Bacchiani: 2003, ‘Supervised and Unsupervised PCFG Adaptation to Novel Domains’. In: *Proceedings of HLT-NAACL 2003*. Edmonton, Canada, pp. 126–133.

Rosén, V., P. Meurer, and K. D. Smedt: 2009, ‘LFG Parsebanker: A Toolkit for Building and Searching a Treebank as a Parsed Corpus’. In: *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*. Utrecht, The Netherlands, pp. 127–133, LOT.

Tanaka, T., F. Bond, S. Oepen, and S. Fujita: 2005, ‘High Precision Treebanking—Blazing Useful Trees Using POS Information’. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*. Ann Arbor, USA, pp. 330–337, Association for Computational Linguistics.

Van der Beek, L., G. Bouma, R. Malouf, and G. Van Noord: 2002, ‘The Alpino Dependency Treebank’. *Computational Linguistics in the Netherlands* **45**(1), 8–22.

Velldal, E.: 2007, ‘Empirical Realization Ranking’. Ph.D. thesis, University of Oslo Department of Informatics.

Verspoor, K., K. B. Cohen, and L. Hunter: 2009, ‘The Textual Characteristics of Traditional and Open Access Scientific Journals are Similar’. *BMC Bioinformatics* **10**(1), 183.

Yeh, A.: 2000, ‘More Accurate Tests for the Statistical Significance of Result Differences’. In: *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*. pp. 947–953.

Ytrestøl, G., D. Flickinger, and S. Oepen: 2009, ‘Extracting and Annotating Wikipedia Sub-Domains – Towards a New eScience Community Resourc’. In: *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories*. Groeningen, The Netherlands.

Zhang, Y. and V. Kordoni: 2010, ‘Discriminant Ranking for Efficient Treebanking’. In: *Coling 2010: Posters*. Beijing, China, pp. 1453–1461, Coling 2010 Organizing Committee.

Zhang, Y., S. Oepen, and J. Carroll: 2007, ‘Efficiency in Unification-based N-best parsing’. In: *Proceedings of IWPT 2007*. Prague, Czech Republic, pp. 48–59.