Revisiting Supertagging for Faster HPSG Parsing

Olga Zamaraeva and Carlos G´omez-Rodr´ıguez Universidade da Coru~na, CITIC Departamento de Ciencia

February 21, 2026

Abstract

We present new supertaggers trained on English grammar-based treebanks and test the effects of the best tagger on parsing speed and accuracy. The treebanks are produced automatically by large manually built grammars and feature high-quality annotation based on a well-developed linguistic theory (HPSG). The English Resource Grammar treebanks include diverse and challenging test datasets, beyond the usual WSJ section 23 and Wikipedia data. HPSG supertagging has previously relied on MaxEnt-based models. We use SVM and neural CRF- and BERT-based methods and show that both SVM and neural supertaggers achieve considerably higher accuracy compared to the baseline and lead to an increase not only in the parsing speed but also the parser accuracy with respect to gold dependency structures. Our fine-tuned BERT-based tagger achieves 97.26

1 Introduction

We present new supertaggers for English and use them to improve parsing efficiency for Head-driven Phrase Structure Grammars (HPSG). Grammars have been gaining relevance in the natural language processing (NLP) landscape (Someya et al., 2024), since it is hard to interpret and evaluate the output of NLP systems without robust theories. Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994, HPSG) is a theory of syntax that has been applied in computational linguistic research (see Bender and Emerson 2021 §3–§4). At a high level, an HPSG grammar is a set of declarative rules and lexical entries, so that every parse is a structure which follows these rules and where every node has rich feature annotation. An HPSG grammar can have broad coverage (see for example the English Resource Grammar (Flickinger, 2000, ERG) and the Alpino grammar (van Noord et al., 2006, for Dutch)). These broad-coverage grammars encode a large amount of linguistic information, and their parsers can produce full dependency structures and semantic representations.

The broad-coverage HPSG grammars are used to create high-quality treebanks automatically, i.e. by parsing large corpora and selecting the best parses (Oepen et al., 2004; Flickinger et al., 2012). The resulting treebanks are used in grammar coaching (Flickinger and Yu, 2013; Morgado da Costa et al., 2016, 2020), natural language generation (Hajdik et al., 2019), and as training data for high precision semantic parsers (Lin et al., 2022; Chen et al., 2018; Buys and Blunsom, 2017). Assuming a good parse ranking model, a treebank is produced automatically by parsing text with the grammar, and any updates are encoded systematically in the grammar, with no need of manual treebank annotation.1

HPSG parsing, which is typically bottom-up chart parsing, is both relatively slow and RAM-hungry. Often, more than a second is required to parse a sentence (see Table 7), and sometimes the performance is prohibitively bad for long sentences, with a typical user machine requiring unreasonable amounts of RAM to finish parsing with a large parse chart (Marimon et al., 2014; Oepen and Carroll, 2002). It is important to emphasize that this is the state of the art in HPSG parsing, and its speed is one of the reasons why the true potential of HPSG parsing in NLP remains not fully realized despite the evidence that it helps create highly precise training data automatically. Approaches to speed up HPSG parsing include local ambiguity packing (Tomita, 1985; Malouf et al., 2000; Oepen and Carroll, 2002), on the one hand, and forgoing exact search and reducing the parser search space, on the

other (Dridan et al., 2008; Dridan, 2009, 2013).

Here we contribute to the second line of research, aka supertagging, a technique to discard unlikely interpretations of tokens. Dridan et al. (2008) and Dridan (2009, 2013) used maximum entropy-based models trained on a combination of gold and automatically labeled data from English, requiring large-scale computation. They report an efficiency improvement of a factor of 3 for the parser they worked with (Callmeier, 2000) and accuracy improvements with respect to the ParsEval metric.

We present new models for HPSG supertagging, an SVM-based one, a neural CRF-based one, and a fine-tuned-BERT one, and compare their tagging accuracy with a MaxEnt baseline. We now have more English gold training data thanks to the HPSG grammar engineering consortium's treebanking efforts (Flickinger, 2000; Oepen et al., 2004; Flickinger, 2011; Flickinger et al., 2012).2 It makes sense to train modern models on this wealth of gold data. Then we use the supertags to filter the parse chart at the lexical analysis stage, so that the parser has fewer possibilities to consider. We observe improvements both in speed and accuracy with the best supertagger integrated into the state-of-the-art HPSG parser: a speedup by a factor of 3 with respect to no supertagging and an improvement in dependency extraction accuracy.

2 Background

We define lexical types and the ERG treebanks, and show why lexical types are useful for speeding up parsing.

2.1 Lexical types

An HPSG grammar includes a large type hierarchy, which defines a system of types for linguistic objects. Among others, these objects include lexical entries (words). The lexical entries are associated with lexical types. The lexical types are shared among words and define their behavior. For example, the lexical type for a transitive verb includes a specification that the verb has a subject and an object. In HPSG, words are often associated with several lexical types, which correspond to different possible interpretations, for example a noun and a verb interpretation. The parser considers all possible lexical types for each token, and this can lead to a combinatorial explosion.

IMAGE NOT PROVIDED

Figure 1: Part of the HPSG type hierarchy (simplified; adapted from ERG). NB: This is not a derivation.

2.2 The ERG treebanks

The English Resource Grammar (ERG) is a broad-coverage HPSG grammar for English (Flickinger, 2000). The ERG is part of the DELPH-IN consortium and is used to parse text and produce semantic representations. The ERG has a large hand-built lexicon and a complex set of rules.

The ERG treebanks (also known as the Redwoods treebanks) are produced automatically by parsing corpora with the ERG and selecting the best parse for each sentence. This yields a large, high-quality set of parses. The ERG treebanks include Wall Street Journal (WSJ) data, Wikipedia data, conversational data, and other domains. They also include an out-of-domain technical essay, The Cathedral and the Bazaar (Raymond, 1999).

2.3 HPSG parsing

HPSG parsing is typically done using a chart parser. The chart parser constructs a parse chart bottom-up, combining lexical entries and applying grammar rules. The complexity of parsing depends on the number of possible lexical types per token and the number of possible rule applications. The worst-case parsing time for HPSG feature structures is proportional to $C2n\rho + 1$ where ρ is the maximum number of children in a phrase structure rule and C is a constant factor (Malouf et al., 2000). As a result, reducing the number of lexical types considered for each token can lead to substantial speed improvements.

3 Methodology

We train and evaluate supertagging models for predicting lexical types, and integrate the best model into a state-of-the-art HPSG parser. Supertagging is simi-

IMAGE NOT PROVIDED

Figure 2: Two interpretations of the sentence The dog barks. The second one is an unlikely noun phrase fragment, which would be discarded with the supertagging technique. (Trees provided by the English Resource Grammar Delphin-viz online demo.)

lar to POS tagging, but the tagset is much larger and the tags encode rich syntactic information.

3.1 Previous and related work

Bangalore and Joshi (1999) introduced the concept of supertagging. Clark and Curran (2003) showed mathematically that supertagging improves parsing efficiency for a lexicalized formalism (CCG). They used a maximum entropy model; Xu et al. (2015) introduced a neural supertagger for CCG. Vaswani et al. (2016) and Tian et al. (2020) further improved the accuracy of neural-based CCG supertagging achieving an accuracy of 96.25

Supertagging experiments with HPSG parsing speed using hand-engineered grammars are summarized in Table 1. In addition, there were experiments on the use of supertagging for parse ranking with statistically derived HPSG-like grammars (Ninomiya et al., 2007; Matsuzaki et al., 2007; Miyao and Tsujii, 2008; Zhang et al., 2009, 2010; Zhang and Krieger, 2011; Zhang et al., 2012). These statistically derived systems are principally different from the ERG as they do not represent HPSG theory as understood by syntacticians. In the context of the ERG, Dridan et al. 2008 represents our baseline SOTA for the tagger accuracy. Dridan 2013 is a related work on "ubertagging", which includes multi-word expressions. Specifically, an ubertagger considers various multi-word spans, whereas a supertagger relies on a standard tokenizer. We use the ubertagger that was implemented for the ACE parser for the parsing speed experiments, as the baseline (§4.2). Dridan's (2013) parsing accuracy results, however, are not comparable to ours; she used a different dataset, a different parser, and a different accuracy metric.

model	grammar	training tok
Alpino (Dutch)	24 mln	1,365
113K	615	8.5 MEMM (Dridan, 2009, p. 169)
676	12 height	

Table 1: Supertagging effects on HPSG parsing speed.

dataset	description	sent
technical essay	713	17,244
1,088	11,550	24,934
34,098	147,166	90.45
1,578	92.88	95.31
93.57	94.29	95.62
92.02	93.66	95.59
96.05	97.26	[MISSING] all
96.02	[MISSING] all	average
[MISSING] speed (sen/sec)	average	7,918

Table 2: Baseline (MaxEnt) and experimental supertaggers' accuracy and speed on test data; tagset size is 1,299.

3.2 Data

We train and evaluate our taggers, both for the baseline (§4.1.1) and for the experiment (§3.3), on gold lexical types from the ERG 2023 release (§2.2). We use the train-dev-test split recommended in the release.7 There are 84,894 sentences in the training data, 2,045 in dev, and 7,918 in test. WSJ section 23 is used as test data, as is traditional, but so are a number of other corpora, notably The Cathedral and the Bazaar (Raymond, 1999), a technical essay which serves as the out-of-domain test data. See Table 2 for the details about the test data. The column titled "training tokens" shows the number of tokens for the training dataset which is from the same domain as the test dataset in the row. For example, WSJ23 has 23K tokens and WSJ1-22 have 960K tokens in the ERG treebanks.

3.3 SVM, LSTM+CRF, and fine-tuned BERT

We train a liblinear SVM model with default parameters (L2 Squared Hinge loss, C=1, one-v-rest, up to 1,000 training iterations) using the scikit-learn library (Pedregosa et al., 2011). We use a set of features inspired by Dridan (2009) including word form, prefixes/suffixes, capitalization patterns, and surrounding

context.

We train a neural LSTM+CRF model using NCRF++ (Yang and Zhang, 2018). The neural model uses pre-trained GloVe embeddings (Pennington et al., 2014) and character embeddings, followed by a BiLSTM and a CRF decoding layer.

Finally, we fine-tune BERT (Devlin et al., 2019) for token classification. We experiment with different learning rates, cased/uncased models, and select the best model based on dev accuracy.

3.4 The ACE HPSG Parser

We use the ACE parser (Callmeier, 2000) for HPSG parsing experiments. ACE is a high-performance HPSG parser used with the ERG. It supports different parsing options and has an in-built "ubertagger" (Dridan, 2013) which prunes lexical analyses based on a statistical model.

3.5 Exceptions for supertagging

A key issue in supertagging for grammar-based parsing is that each tagging mistake can have a large impact: selecting a wrong lexical type even for one word means the entire sentence will likely not be parsed correctly. Thus the accuracy of the tagger is crucial. Related to this is the matter of how many possibilities to consider for supertags: the more are considered, the slower the parsing, but the higher the accuracy. In this paper, we experiment with a single, highest-scored tag for each token. However, we combine this strategy (which prioritizes parsing speed) with a list of tokens exempt from supertagging (which increases accuracy).

4 Results

We first present tagging accuracy and speed results for the baseline and experimental models. Then we present parsing speed and parsing accuracy results.

4.1 Tagger accuracy and tagging speed

4.1.1 Tagging accuracy baseline

We use a MaxEnt baseline model trained using scikit-learn (Pedregosa et al., 2011). We train and evaluate the baseline on the same training data split as the experimental models.

model	top mistaken token (all)	top mistaken token (not cl
to	to	$\mathrm{n}_{np}n_t emp_n t$
to	$\mathrm{n}_{np}n_t emp_n t$	$\mathrm{n}_{np}n\ MaxEnt$
$\mathrm{n}_{np}n_t emp_n t$	$\mathrm{n}_{np}n$ height	

Table 3: [ILLEGIBLE]

4.1.2 Tagger accuracy results

Table 2 presents tagging accuracies and decoding speeds for the baseline and experimental models across test datasets. The BERT-based model achieves the best accuracy across datasets, with 97.26

We also analyze the most common mistakes of the models. Table 3 shows the most frequent mistaken tokens and the most under- and over-predicted tags.

4.2 Results: Parsing Speed and Accuracy

4.2.1 Baseline

We compare our system with two systems: ACE with no tagging at all and ACE with the in-built "ubertagger". The system with no tagging at all is the baseline for parsing speed and, theoretically, the upper boundary for the parsing accuracy (as the parser could have access to the full lexical chart). However, in practice it is difficult to obtain this upper bound because it requires at least 54GB of RAM (see §A.5) and the parsing takes unreasonably long (up to several minutes per sentence). With realistic settings, the system with no tagging fails to parse some of the longer sentences because the lexical chart exceeds the RAM limit. It is precisely the problem that ubertagging/supertagging is supposed to solve: reduce the size of the lexical chart so that the parsing can be done with realistic RAM allocation and in reasonable time.

The ubertagger is a MEMM tagger based on Dridan 2013. It was trained on millions of sentences using large computational resources (the Titan system at University of Oslo) and as such is not easily reproducible. In contrast, our BERT-based model is fairly easy to fine-tune and reproduce on an individual machine. For the purposes of parsing accuracy and speed, rather than comparing our system to other experimental taggers presented in §4.1, we compare it

IMAGE NOT PROVIDED

Figure 3: Pareto Frontier (Speed and F-score)

dataset	description	sent
technical essay	713	17,244
1,088	11,550	0.55
34,098	2.40	0.13
1.93	0.10	0.23 vm32
0.13	0.27 ws213-214	Wikipedia
1.39 wsj23	Wall Street J.	950

Table 4: [ILLEGIBLE]

to the ubertagger because the ubertagger is integrated into the ACE parser for production and as such is a more challenging baseline.

4.2.2 Default parsing

Tables 4, 5, and 6 present the results for the ACE parser default RAM limit setting (1200MB). On the ubertagger and the supertagger side, we use all the predictions and do not exclude any tags from the pruning process. The results show that while we can parse faster with tagging (the ubertagger being the fastest), both the ubertagger and the supertagger suffer from the high cost of each tagging mistake: while the new BERT-based supertagger is more accurate, its accuracy is still not 100

4.2.3 Parsing with exceptions lists

Tables 7-9 present the results for parsing with ubertagging and supertagging with exceptions. The no-tagging system's results are the same as before; we repeat them for convenience.

We have looked at the most common mistakes in the supertags in the training data and have compiled a list of 15 tags which BERT tends to predict wrong.14 On the ubertagger side, there was already a list of exceptions. The ubertagger's exception list is a list of 1715 lexical entries (words, e.g. "my"), whereas ours is a list of 15 lexical types (tags, e.g. "d-poss-my", which is a supertype for "my" in the grammar). The ubertagger's list includes some of the same phenomena as ours.

dataset	description	sent	tok	No tagging P	No
technical essay	713	17,244	0.89	0.40	
1,088	11,550	0.93	0.84	0.88	
34,098	0.87	0.68	0.76	0.84	
0.85 tok	0.68	No tagging	0.75	Ubertagging 0.78	BERT-based 0.30
0.65 6.15	0.74	0.42	0.81	0.76 ecpr	0.40
0.64 0.05	0.88	0.52 jh*,tg*,ps*, ron*	0.41	travel brochures 0.56	0.90
0.86 0.32 petet	0.53	textual entailment	0.65	0.86 581	0.51
phone conv.		1,000		8,730	
1,470		22,697		4.71	
22,987		4.76		0.37	

Table 5: [ILLEGIBLE]

dataset	description	sent	tok	
technical essay	713	17,244	0.71	
1,088	11,550	0.87	0.86	0.
34,098	0.89	0.85	0.88 petet	
0.92	0.79	0.83 vm32	phone conv.	
0.80	0.80 ws213-214	Wikipedia	1,470	
0.75 wsj23	Wall Street J.	950	22,987	

Table 6: [ILLEGIBLE]

5 Conclusion and future work

We used the advancements in HPSG treebanking to train more accurate supertaggers. The ERG is a major project in syntactic theory and an important resource for creating high quality semantic treebanks. It has the potential to contribute to NLP tasks that require high precision and/or interpretability including probing of the LLMs, and thus making HPSG parsing faster is strategic for NLP. We tested the new supertagging models with the state-of-the-art HPSG parser and saw improvements in parsing speed as well as accuracy. We consider the results on multiple domains, well beyond the WSJ Section 23. We show promising results but also confirm that domain remains important, and purely statistical systems are brittle and often require rule-based additions in real-life scenarios. We contribute the ERG datasets converted to huggingface transformers format intended for token classification, along with the code which can be adapted for other purposes.

6 Limitations

Our paper is concerned with training supertagging models on an English HPSG treebank. The limitations therefore are associated mainly with the training

dataset	description	sent	tok
technical essay	713	17,244	6.15
1,088	11,550	0.55	8.05
34,098	2.40	0.13	0.27
1.93	0.10	0.17	vm32
0.13	0.22	ws213-214	Wikipedia
0.97	wsj23	Wall Street J.	950

Table 7: [ILLEGIBLE]

dataset	description	sent	tok	No tagging	No tagging R	No tagging F1	Ubertagging
technical essay	713	17,244	0.89	2	0.40	0.55	0.86
1,088	11,550	0.93	0.84	tuned	0.88	0.92	100–1200
34,098	0.87	0.68	0.76	word emb. dim.	300	0.43	0.57
0.85	0.68	0.75	0.78	300	0.50	0.43	0.87
0.65	0.74	0.81	0.27	0.5	0.40	0.87	0.46
0.64	0.88	0.41	0.56	[MISSING]	1.89	0.50	0.64
0.86	0.53	0.65	0.87		0.55	0.68	height

Table 8: [ILLEGIBLE]

dataset	description	sent	tok	No tagging	No tagging R	No tagging F1	Ubertagging
technical essay	713	17,244	6.15	2	0.42	0.71	0.86
1,088	11,550	8.05	0.55	jhk	0.89	0.87	travel brochures
34,098	2.40	0.13	0.27	petet	textual entailment	0.85	0.87
1.93	0.10	0.17	vm32	phone conv.	0.92	0.79	1,000
0.13	0.22	ws213-214	Wikipedia	0.80	1,470	0.90	29,297
0.97	wsj23	Wall Street J.	950	0.23	22,987	4.76	wsj23

Table 9: [ILLEGIBLE]

dataset	description	sent	tok	No tagging	No tagging R	No tagging F1	Ubertagging
technical essay	713	17,244	0.89	2	0.40	0.55	0.86
1,088	11,550	0.93	0.84	tuned	0.88	0.92	100–1200
34,098	0.87	0.68	0.76	word emb. dim.	300	0.43	0.57
0.85	0.68	0.75	0.78	300	0.50	0.43	0.87
0.65	0.74	0.81	0.27	0.5	0.40	0.87	0.46
0.64	0.88	0.41	0.56	[MISSING]	1.89	0.50	0.64
0.86	0.53	0.65	0.87		0.55	0.68	height

Table 10: NCRF++ model parameters

of the models including neural networks, and with the building of broad-coverage grammars such as the English Resource Grammar. Crucially, while our method does not require industry-scale computational resources, training a neural classifier such as ours still requires a certain amount of training data, and this means that our method assumes that a large HPSG treebank is available for training. The availability of such a treebank, in turn, depends directly on the availability of a broad-coverage grammar.

Acknowledgments

We are grateful to the anonymous reviewers for their constructive feedback. We are also grateful to the ERG and DELPH-IN communities for making their resources available.

A Appendix A

A.1 Tuning ranges

BERT (Devlin et al., 2019) was fine-tuned using transformers (Wolf et al., 2019) and pytorch (Paszke et al., 2017) using 4 learning rates: 1e-5, 2e-5, 3e-5, and 5e-6. Cased and uncased pretrained BERT models were tried.

A.2 Computational resources

We trained the neural models with a single NVIDIA GeForce RTX 2080 GPU, CUDA version 11.2. The SVM model and the MaxEnt baseline were trained using Intel Core i7-9700K 3,60Hz CPU (using single core processing for each model). We have experimented with Stochastic Gradient Descent (SGD) optimizer along with AdaGrad, Adam, and AdaDelta. The ranges for parameter values can be found in Table 10. The decoding time (sentences per second) for the models can be found in Table 2. The training times are presented in this Appendix in Table 11. The energy costs as estimated by the Python library carbontracker (Anthony et al., 2020) 15 are in Table 12.

A.3 Development accuracies

The development (validation set) accuracies are presented in Tables 13, 14, and 15. The best models are bolded. NCRF++ has nondeterministic components, and the average dev accuracy of the best (bolded) model in Table 14; the average accuracy is 95.15

A.4 MaxEnt model

Below we describe in detail how we trained the baseline MaxEnt models.

Model type	models trained for tuning	total models for all models in this row (approx.)	height
1	3664	MaxEnt Scikit-learn	
106,922	NCRF++	31	
5	100,000 (approx.)	height	

Table 11: Training times for models used to choose the best baseline and best experimental models

Measurement	Value	Measurement	Value
5.55 kWh	3.5 kWh	Carbon emissions	1.63 kg CO2
5.5 kg CO2	Equivalent km driven	13 km	5.5 km

Table 12: Energy cost estimate for training the final NCRF++ model in 38 epochs (31 were trained in total, number of epochs varied) and for BERT 50 epochs

A.4.1 MaxEnt model selection

Rather than comparing our experimental numbers with numbers obtained by Dridan (2009),16 we create our own baseline because we want to be able to compare classic models with neural models with the same amount of training data. We experimented with autoregressive and nonautoregressive MaxEnt models and in the end chose one MaxEnt as the baseline.

A.4.2 MaxEnt classifiers

We use scikit learn Python library (Pedregosa et al., 2011) to train the baseline MaxEnt classifiers.17 The scikit learn classifiers are optimized for processing a large number of observations. For that reason, we organized our evaluation data (dev and test) so as to maximize the number of observations passed to the classifier at each step. Dridan's (2009) models were autoregressive; we also implemented autoregressive baseline models, and in order to make them faster at test time, we organized the evaluation data by the word's position in the sentence. So the classifier would first process all the first words in all sentences, then all the second words, etc. For nonautoregressive models, which we also tried in order to find the best-performing baseline model, we just pass the classifier the entire list of observations in their original order.

Model type	models trained for tuning	total models for all models in this row (approx.)	height
[MISSING]	SVM	93.02	[MISSING]
95.15	[MISSING]	1258	[MISSING]
[MISSING]	height		

Table 13: [ILLEGIBLE]

Model	[ILLEGIBLE]	[ILLEGIBLE]	height
[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]

Table 14: [ILLEGIBLE]

References

[1] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. ArXiv:2007.03051.

[2] Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. Computational linguistics, 25(2):237–265.

[3] Emily M Bender and Guy Emerson. 2021. Computational linguistics and grammar engineering. In Stephan M¨uller, Anne Abeill´e, Robert D. Borsley, and Jean-Pierre Koenig, editors, Head-Driven Phrase Structure Grammar: The handbook, pages 1–46. Language Science Press.

[4] Thorsten Brants. 2000. TnT: A statistical part-of-speech tagger. In Proceedings of the sixth conference on Applied natural language processing, pages 224–231.

[5] Jacob Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1215–1226.

[6] Ulrich Callmeier. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. Natural Language Engineering, 6(1):99–108.

[7] Yen-Chun Chen, Yu-Ying Chiu, and Jyun-Sheng Huang. 2018. Improving semantic parsing using semantic dependency parsing for building a semantic graph. In Proceedings of the 27th International Conference on Computational Linguistics, pages 4321–4332.

Model	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	height

Table 15: [ILLEGIBLE]

Model	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	height

Table 16: [ILLEGIBLE]

Model	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	[ILLEGIBLE]	height

Table 17: [ILLEGIBLE]

Notagging		[ILLEGIBLE]	Utagging		BERTsupertagging		heightda
coverage			F-score	speed			cov
F-score	speed		cb				0.86
59.3			0.58	0.58			
0.63	0.64		8.58				ecpr
0.95			0.68	0.96			
0.06	0.97		0.93				0.68
0.98			0.88	8.89			
0.75	0.22		0.91				0.87
petet			0.99	0.92			
0.79	0.79		0.12				0.85
0.32			vm32	0.99			
0.99	0.87		0.86				0.05
0.90			0.10	wsj23			
0.79	52.2		0.64				0.69
-			-	*			[MISS
[MISSING]	height						

Table 18: Parsing with 54GB RAM

[8] Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pages 97–104.

[9] Mariana Morgado da Costa, Dan Flickinger, and Christopher Manning. 2016. The redwoods treebank and HPSG parse selection. In Proceedings of [ILLEGIBLE].

[10] Mariana Morgado da Costa, Dan Flickinger, and Christopher Manning. 2020. Grammar development and treebanking for [ILLEGIBLE].

[11] A. M. Dridan, Stephan Oepen, and Kristina Toutanova. 2008. Supertagging for efficient deep grammatical processing. In Proceedings of ACL-08: HLT, pages 845–853.

[12] Rachel Dridan. 2009. Accurate and efficient linguistic processing using a Supertagging approach. Ph.D. thesis, University of Sydney.

[13] Rachel Dridan. 2013. A multiword expression supertagger for efficient HPSG parsing. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1175–1184.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019, pages 4171–4186.

[15] Dan Flickinger. 2000. On building a more efficient grammar by exploiting existing resources. In Proceedings of the Workshop on Efficient Processing with HPSG, pages 1–11.

[16] Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In Proceedings of the 2011 conference on [ILLEGIBLE].

[17] Dan Flickinger and Jiye Yu. 2013. Toward more precise grammar coaching. In Proceedings of [ILLEGIBLE].

[18] Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories.

[19] Bj¨orn Hajdik, Michael White, and [ILLEGIBLE]. 2019. High-precision generation from deep grammar semantics. In Proceedings of [ILLEGIBLE].

[20] Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In Proceedings of ACL 2015, pages 483–488.

[21] Maximilian Kogkalidis and Michael Moortgat. 2023. Supertagging improves robustness to rare words. In Proceedings of [ILLEGIBLE].

[22] Xiaoqiang Lin, [ILLEGIBLE]. 2022. Training high precision semantic parsers from ERG treebanks. In Proceedings of [ILLEGIBLE].

[23] Bing Liu, [ILLEGIBLE]. 2021. Fine-grained CCG supertagging. In Proceedings of [ILLEGIBLE].

[24] Robert Malouf, John Carroll, and Ann Copestake. 2000. Efficient feature structure processing for unification-based grammars. Natural Language Engineering, 6(1):1–22.

[25] Montserrat Marimon, [ILLEGIBLE]. 2014. Speeding up deep parsing. In Proceedings of [ILLEGIBLE].

[26] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and [ILLEGIBLE]. In Proceedings of [ILLEGIBLE].

[27] Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. Computational Linguistics, 34(1):35–80.

[28] Stephan Oepen and John Carroll. 2002. Deep grammatical processing of large corpora. In Proceedings of COLING 2002, pages 1–7.

[29] Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit. 2004. LinGO redwoods: A rich and dynamic treebank for HPSG. Research on Language and Computation, 2(4):575–596.

[30] Chris Packard. 2015. Treebanking and grammar engineering: Semi-automatic parse selection. In Proceedings of [ILLEGIBLE].

[31] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Automatic differentiation in PyTorch. NIPS-W.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

[33] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of EMNLP 2014, pages 1532–1543.

[34] Carl Pollard and Ivan A. Sag. 1994. Head-Driven Phrase Structure Grammar. University of Chicago Press.

[35] Maja Prange, [ILLEGIBLE]. 2021. Long-tail phenomena in supertagging. In Proceedings of [ILLEGIBLE].

[36] Thijs Prins and Gertjan van Noord. 2004. Unsupervised POS tagging for deep grammar parsing. In Proceedings of [ILLEGIBLE].

[37] Eric S. Raymond. 1999. The Cathedral and the Bazaar. O'Reilly Media.

[38] Yusuke Someya, [ILLEGIBLE]. 2024. [ILLEGIBLE]. In Proceedings of [ILLEGIBLE].

[39] David A. Wolff, [ILLEGIBLE]. 2020. [ILLEGIBLE]. In Proceedings of [ILLEGIBLE].

[40] Hiroyuki Tomita. 1985. Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems. Kluwer.

[41] Ashish Vaswani, [ILLEGIBLE]. 2016. Supertagging with neural networks. In Proceedings of [ILLEGIBLE].

[42] Gertjan van Noord, [ILLEGIBLE]. 2006. The Alpino grammar for Dutch. In Proceedings of [ILLEGIBLE].

[43] Yi Zhang, [ILLEGIBLE]. 2009. [ILLEGIBLE]. In Proceedings of [ILLEGIBLE].

[44] Yi Zhang, [ILLEGIBLE]. 2010. [ILLEGIBLE]. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 645–648.

[45] Yi Zhang and Hans-Ulrich Krieger. 2011. Large-scale corpus-driven PCFG approximation of an HPSG. In Proceedings of the 12th international conference on parsing technologies, pages 198–208.

[46] Shuangzhi Wu, Thomas Wolf, and [ILLEGIBLE]. 2019. Transformers: State-of-the-art natural language processing. In Proceedings of [ILLEGIBLE].

[47] Wei Xu, [ILLEGIBLE]. 2015. Neural CCG supertagging. In Proceedings of [ILLEGIBLE].

[48] Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In Proceedings of ACL 2018 (System Demonstrations), pages 74–79.