On the Derivation Perplexity of Treebanks

Anders Søgaard

Center for Language Technology
University of Copenhagen
Njalsgade 142
DK-2300 Copenhagen S
Email: soegaard@hum.ku.dk

Martin Haulrich

ISV Computational Linguistics Group
Copenhagen Business School
Dalgas Have 6
DK-2000 Frederiksberg
Email: mwh.isv@cbs.dk

Abstract

Parsing performance is typically assumed to correlate with treebank size and morphological complexity [6, 13]. This paper shows that there is a strong correlation between derivation perplexity and performance across morphologically rich and poor languages. Since perplexity is orthogonal to morphological complexity, this questions the importance of morphological complexity. We also show that derivation perplexity can be used to evaluate parsers. The main advantage of derivation perplexity as an evaluation metric is that it measures global aspects of parsers (like counting exact matches), but is still fine-grained enough to derive significant results on small standard test sets (like attachment scores).

1 Introduction

State-of-the-art accuracy on a particular parsing dataset is typically assumed to correlate with treebank size and morphological complexity of the language in question [6, 13]. Out-of-domain evaluation also shows that parsers are typically very domain-sensitive. For example, as shown in the CoNLL 2007 shared task, parsers trained on the Penn Treebank are much better at parsing the Wall Street Journal than at parsing biomedical articles or transcribed speech. This of course relates to perplexity, since out-of-domain text is much less predictable for language models.

However, no one has to the best of our knowledge used perplexity as a metric to better evaluate parsers and treebanks. This paper suggests some ways to do so and presents some preliminary experiments.

The easiest way to explain perplexity is perhaps by considering the case of a fair k-sided die. The perplexity of such a die is k, meaning that we are k-ways perplexed about the outcome of the die. If the die is unfair and always end with the same side up, the perplexity is 1, meaning that we are certain about the outcome of the die.

Given a language model **lm** trained on a corpus (of sentences or derivation orders), we are interested in how well it predicts a sample of test instances $x_1, \ldots, x_N$. The perplexity is defined as:

$$2^{-\Sigma_{i=1}^{N} \frac{1}{N} \log_2 \mathbf{lm}(x_i)}$$

Better language models will tend to assign higher probabilities to the instances in our sample and will thus have lower perplexity. In our experiments in this paper, we use standard trigram language models with modified Kneser-Ney smoothing and interpolation.

Dependency treebanks are collections of dependency trees. A dependency tree is a tree that represents a syntactic analysis such that words are vertices with various labels and grammatical functions label the directed edges (dependencies). Each word thus has a single incoming edge, except one called the root of the tree. Dependency parsing is thus a structured prediction problem with trees as structured variables. Each sentence has exponentially many possible dependency trees. The observed variables are typically sentences with words labeled with part-of-speech tags. The parsing task for each sentence is to find the dependency tree that maximizes an objective function which is typically learned from a dependency treebank.

The standard metrics used in dependency parsing are labeled attachment score (LAS), i.e. the ratio of words with correct syntactic heads and grammatical functions, unlabeled attachment score (UAS), i.e. the ratio of words with correct syntactic heads, and exact matches (EM), i.e. the ratio of sentences in which all words are assigned correct syntactic heads. The disadvantage of LAS and UAS is that attachment scores do not reflect global properties of the predicted syntactic analyses, while EM has the disadvantage that differences are seldom statistically significant on small evaluation data sets. This paper suggests that perplexity of derivation order may also be a useful metric for parser evaluation. It is not a stand-alone metric, but used in conjunction with LAS or UAS it may supply the global information that EM is supposed to reflect.

Perplexity of derivation order may also be used to predict state-of-the-art accuracy on treebanks and thereby indirectly for treebank evaluation. State-of-the-art parsing performance is typically assumed to correlate with treebank size and morphological complexity, but in this paper we show that there is a strong correlation between perplexity of derivation order and parsing performance across morphologically rich and poor languages.

1.1 Related Work

Nivre [6] presents an analysis of the CoNLL 2007 shared task, building on [9], and draws the conclusion that state-of-the-art parsing accuracy primarily depends on morphological complexity.

> The ten languages involved in the multilingual track can be grouped into three classes with respect to the best parsing accuracy achieved:
>
> - Low (LAS = 76.3-76.9): Arabic, Basque, Greek
> - Medium (LAS = 79.2-80.2): Czech, Hungarian, Turkish
> - High (LAS = 84.4-89.6): Catalan, Chinese, English, Italian
>
> To a large extent, these classes appear to be definable from typological properties. The class with the highest top scores contains languages with a rather impoverished morphology. Medium scores are reached by the two agglutinative languages, Hungarian and Turkish, as well as by Czech. The most difficult languages are those that combine a relatively free word order with a high degree of inflection. Based on these characteristics, one would expect to find Czech in the last class. However, the Czech training set is four times the size of the training set for Arabic, which is the language with the largest training set of the difficult languages. On the whole, however, training set size alone is a poor predictor of parsing accuracy, which can be seen from the fact that the Italian training set is only about half the size of the Arabic one and only one sixth of Czech one. Thus, there seems to be a need for parsing methods that can cope better with richly inflected languages.

The same conclusion was the motivation for a workshop in Statistical Parsing of Morphologically Rich Languages at NAACL'10 in Los Angeles, California [13]. In this paper, we show that, not surprisingly, there is a strong correlation between treebank size and state-of-the-art accuracy, but also that a stronger correlation exists between relative derivation perplexity and state-of-the-art accuracy, even across morphologically rich and poor languages.

Evaluation of parsers has been widely debated in recent years. Caroll et al. [2] review the parsing evaluation metrics that were available at the time and propose a new one. They first discuss a number of metrics that can be used with unannotated corpora, incl. coverage, average ambiguity and perplexity. The problem with coverage and average ambiguity is that the metrics do not say anything about accuracy. The perplexity of a parsing model on a corpus may under certain assumptions tell us about the accuracy of the model or about the "degree to which a model captures regularities in the corpus by minimising unpredictability and ambiguity". The advantages of this metric are that it has a clear probabilistic interpretation, allows meaningful comparison and can be used "as a method for scaling results obtained

using other corpus-dependent measures to allow for some degree of cross-corpus comparison and evaluation." The disadvantages are that the metric is "expensive to compute", "only applicable to probabilistic models" and that it "only provides a weak measure of accuracy." The notion of derivation perplexity introduced here differs from perplexity of a probabilistic parsing model in that it can be read off structures immediately. We can therefore talk about the derivation perplexity of parsers as well as treebanks. Consequently, it is *not* expensive to compute, it is applicable to non-probabilistic models (such as transition-based dependency parsers), and we also show that it correlates strongly with stronger measures of accuracy.

Rimell et al. [11] suggest a new evaluation scenario for dependency parsing of English text. They construct a corpus of 700 English unbounded dependency constructions. They argue:

> These are interesting for parser evaluation for the following reasons: one, they provide a strong test of the parser's knowledge of the grammar of the language, since many instances of unbounded dependencies are difficult to recover using shallow techniques in which the grammar is only superficially represented; and two, recovering these dependencies is necessary to completely represent the underlying predicate-argument structure of a sentence, useful for applications such as Question Answering and Information Extraction.

One problem with this approach, noted already by [2], is that the usefulness of such a corpus depends heavily on what constructions are included. It is certainly not trivial to distinguish between important and less important constructions. It would be interesting to see how retrieval of unbounded dependencies in this corpus correlates with other metrics and pipeline evaluations. Another problem is of course that the metric is language-dependent. The advantage, however, is that retrieval of unbounded dependencies in most parsers depends heavily on the rest of the analysis and thus can be said to capture global aspects of the syntactic analysis.

Finally, we note that many researchers have proposed pipeline evaluation of parsers where parsers are evaluated in terms of their contribution to a particular application, e.g. machine translation [4] or textual entailment [14].

2 Dependency Treebanks

Dependency treebanks have become increasingly popular over the last five years. With the development of fast, reliable dependency parsers [5, 10], theoretically motivated dependency treebanks such as the 1M word Prague Dependency Treebank and two competitive shared tasks in dependency parsing at the Conferences on Natural Language Learning (CoNLL) in 2006–7, large scale evaluation of dependency parsers and pipeline evaluation in natural language processing applications have become possible. Dependency parsers have among other things been used for summarization and machine translation [4].

More formally, a dependency tree for a sentence $x = w_1, \ldots, w_n$ is a tree $T = \langle \{0, 1, \ldots, n\}, A \rangle$ with $A \subseteq V \times V$ the set of dependency arcs. Each vertex corresponds to a word in the sentence, except 0 which is the root vertex, i.e. for any $i \leq n$ $\langle i, 0 \rangle \notin A$. Since a dependency tree is a tree it is acyclic. A tree is projective if every vertex has a continuous projection, i.e. if and only if for every arc $\langle i, j \rangle \in A$ and node $k \in V$, if $i < k < j$ or $j < k < i$ then there is a subset of arcs $\{\langle i, i_1 \rangle, \langle i_1, i_2 \rangle, \ldots, \langle i_{k-1}, i_k \rangle\} \in A$ such that $i_k = k$.

Characteristics of a large portion of the available dependency treebanks can be found in the CoNLL shared task organizers' papers [1, 9], but several other dependency treebanks now exist, incl. treebanks for Ancient Greek, Latin, Romanian and Thai. The treebanks differ in size and domain dependence, and they adopt different annotation guidelines. Different annotation guidelines sometimes complicate translation-oriented applications, and parallel dependency treebanks are therefore also being developed.

In our experiments, we use the treebanks from the CoNLL-X and CoNLL 2007 shared tasks.

3 Derivation Perplexity

A transition-based dependency parser p's derivation perplexity on a text T is defined as the perplexity of the derivation language of $p(T)$, where $p(T)$ is the 1-best parse trees of the sentences in T: The *derivation language* of $p(T)$ is the set of strings $\sigma : w_1 \ldots w_n$ such that for any w_i, w_j with dependency structure d if $w_i \prec w_j$ then w_i was attached to d in $p(T)$ prior to the attachment of w_j.

The derivation perplexity of a treebank R over a text T is the derivation perplexity of $f(T)$ where f is a function from the strings in T into the canonical parse of the corresponding trees in R given some parsing algorithm. In this paper, the parsing algorithm used to obtain canonical parses will be the so-called Swap-Lazy algorithm introduced in [8].

The Swap-Lazy algorithm was chosen because it is a non-projective dependency parsing algorithm (and many of the treebanks used in our experiments contain non-projective dependencies) and because, as documented in [8], it has higher accuracy in terms of exact matches than other state-of-the-art transition-based dependency parsing algorithms.

The algorithm works as follows: We begin with a configuration (Stack, Buffer, Arcs) where Stack is a stack that initially only contains an artificial root element, Buffer is the string to be read, and Arcs is the dependency structure to be build (a set of dependency arcs of the form (w_i, w_j)). The initial configuration is thus $([w_0]_S, [w_1, \ldots, w_n]_B, \{\}_A)$ with $w_1 \ldots w_n$ the input sentence and where w_0 is the artificial root note in the dependency tree. The transition algorithm halts when a final configuration is reached of the form $([w_0]_S, []_B, A)$, i.e. when all words are read and removed from the stack. Below we only consider unlabeled parsing. Otherwise different Right-Arc and Left-Arc transitions must be introduced for each

label. The possible transitions are:

$$\textbf{Shift}\ ([\ldots, w_i]_S, [w_j, \ldots]_B, A)) \Longrightarrow ([\ldots, w_i, w_j]_S, [\ldots]_B, A)$$

$$\textbf{Right-Arc}\ ([\ldots, w_i, w_j]_S, B, A)) \Longrightarrow ([\ldots, w_i]_S, B, A \cup \{(w_i, w_j)\})$$

$$\textbf{Left-Arc}\ [i \neq 0]\ ([\ldots, w_i, w_j]_S, B, A)) \Longrightarrow ([\ldots, w_j]_S, B, A \cup \{(w_j, w_i)\})$$

$$\textbf{Swap}\ [0 < i < j]\ ([\ldots, w_i, w_j]_S, [\ldots]_B, A)) \Longrightarrow ([\ldots, w_j]_S, [w_i, \ldots]_B, A \cup \{(w_j, w_i)\})$$

Intuitively, Shift moves an element from the Buffer to the Stack. Right-Arc builds a dependency arc from the second element (the left word) to the top element. (The dependency arcs build using this transition therefore point to the right; hence, the name.) Left-Arc builds a dependency arc from the top element to the second element, i.e., left arcs. Swap, finally, reorders words by moving the second element on the stack back into the buffer. Consequently, Right-Arc and Left-Arc may build left arcs, resp. right arcs, relative to the original linear order of words. In other words, the intuition behind this parsing algorithm is to reduce discontinuity to adjacency by reordering input words.

Given a dependency structure, there is a canonical derivation of it using the Swap-Lazy algorithm. The derivation sequences constructed from the trees in a treebank are used to train the classifiers that decide which transition to apply in a particular configuration when parsing with the MaltParser [9, 8]. Since each derivation step corresponds to finding a syntactic head for a word, we use the derivation order as a linear reordering simply by printing the words in the order they are attached to the dependency structures. Put differently, a derivation will gradually expand the set of Arcs. For a derivation $([w_0]_s], [w_1, \ldots, w_n]_B, \{\}_A) \Longrightarrow^* ([w_0]_S, []_B, A)$ there is a linear order $\prec$ such that for any w_i, w_j such that $w_i \prec w_j$, w_i was added to A before w_j. It is this linear order whose perplexity is computed in our experiments. In our first experiment, we use the dependency trees from treebanks. In our second experiment, we use the output from four different transition-based dependency parsers.

4 Experiments

4.1 Data

Our experiments cluster the treebanks in three groups: the treebanks used in the CoNLL-X Shared Task (excl. Chinese, which was not available to us), those used in the CoNLL 2007 Shared Task, and the treebanks that were used in both shared tasks and are genuine *dependency treebanks*, i.e. not converted constituent-based treebanks. Using only genuine dependency treebanks have become standard, e.g. [7], when parsing performance is evaluated in terms of exact matches. The third set of treebanks excludes very large treebanks (>200k tokens) and Greek, which our language modeling software (SRI Language Modeling Toolkit [12]) did not process

correctly. The CoNLL-X treebanks is thus a set of 12 treebanks, the CoNLL 2007 treebanks a set of 10 treebanks, and the genuine dependency treebanks is a set of 7 treebanks. In sum, the three clusters of treebanks are as follows:

name	number	languages
C06	12	Arabic, Bulgarian, Czech, Danish, Dutch German, Japanese, Portuguese, Slovene, Spanish, Swedish, Turkish
C07	10	Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, Turkish
gen.dt	7	Arabic(C06), Arabic(C07), Czech, Danish, Slovene, Turkish(C06), Turkish(C07)

4.2 Language Model Parameters

We use a 3-gram language model with modified Kneser-Ney smoothing and interpolation as implemented in the freely available SRI Language Modeling Toolkit [12]. Kneser-Ney smoothing was found to consistently outperform other smoothing techniques in [3].

4.3 Perplexity and State-of-the-Art Accuracy

This experiment was designed to quantify to what extent state-of-the-art parsing accuracy can be predicted from the derivation perplexity of a treebank. For the experiment, we reimplemented the Swap-Lazy algorithm for training the oracle in MaltParser [8] and printed out words in the order of derivation (attachment). Briefly put, the Swap-Lazy algorithm can keep words on the buffer or place them on the stack, but at some point it will attach words to the dependency structure being build, and the words are simultaneously removed from the stack. It is at this point that the word is printed. The result is a linear reordering of the input text that corresponds to the derivation order. The perplexity of this derivation order is computed by training a language model on the reordered training data and running it on the reordered test data. The language model parameters are described above.

What is correlated with state-of-the-art accuracy is not string perplexity and derivation order perplexity, but treebank size over these perplexities. In particular, we correlate (i) treebank size with accuracy (as our baseline), (ii) treebank size over string perplexity with accuracy and (iii) treebank size over derivation order perplexity with accuracy.

The experiment was done for the CoNLL-X shared task treebanks, as well as for the CoNLL 2007 treebanks. Treebank sizes over perplexities are then correlated with state-of-the-art results, i.e. the best results obtained in the shared tasks. We also report correlations with the average results of the shared task participants. This may in fact be a more interesting measure for treebank evaluation, since a treebank where 10 participants achieve an UAS $> 90\%$ is probably "easier" than one where only one participant does so, even if the best scores are identical. It is, however,

more common to focus on the best results obtained in the shared tasks or in the literature [6, 9, 13].

4.4 Perplexity as a Metric for Parser Evaluation

We ran our parsing evaluation experiments on the genuine dependency treebanks. This is a common practice used, for example, in [7] and [8]. We ran the MaltParser [10] with four different parsing algorithms (Arc-Eager, Arc-Standard, Swap-Eager and Swap-Lazy) and default feature settings to obtain four different outputs on each of the CoNLL test sections. UAS, EM and perplexity of derivation order were computed, and Pearson ρ was calculated from these numbers. The average Pearson ρ which is reported below, is the average Pearson ρ for the seven treebanks.

5 Results

We first list the average perplexities and derivation perplexities of the treebanks.

name	av. perplexity	av. deriv. perpl.	increase
C06	292.5	532.8	82.2%
C07	320.3	508.8	37.0%
gen.dt	278.2	447.8	96.6%

Note the higher increase from perplexity to derivation perplexity with genuine dependency treebanks.

5.1 Perplexity and State-of-the-Art Accuracy

The correlation coefficients for state-of-the-art parse accuracy and treebank size over string/derivation perplexity are presented in the table below.

Pearson ρ	C06		C07	
	best	av	best	av
treebank size	0.21245816	0.037759424	0.72245934	0.55952737
string perplexity	**0.47495902**	0.506099378	0.643438377	0.548117203
deriv perplexity	0.47015543	**0.514647899**	**0.810661115**	**0.737857396**

The results indicate that (treebank size over) derivation order perplexity is much better at predicting state-of-the-art parsing accuracy than treebank size only. While there is a strong correlation between (treebank size over) string perplexity and accuracy, the notion of derivation order perplexity seems more relevant than mere string perplexity.

Since the correlation between derivation perplexity and accuracy cuts across morphological complexity, and since morphological complexity is orthogonal to perplexity, this questions the importance of morphological complexity to parsing performance. Morphologically poor languages such as Chinese typically lead to

very high perplexities (in our case ∼900), while morphologically rich languages such as Turkish typically exhibit moderate perplexities (in our case ∼120).

5.2 Perplexity as a Metric for Parser Evaluation

We only ran our parsing evaluation experiments on the seven genuine dependency treebanks. We ran the MaltParser [10] with four different parsing algorithms (Arc-Eager, Arc-Standard, Swap-Eager and Swap-Lazy) and default feature settings to obtain four different outputs on each of the CoNLL test sections, thus running a total of 28 dependency parsers. We computed the Pearson ρ correlations between UAS, EM and perplexity of derivation order for each treebank and averaged over these numbers. Perplexity of derivation order (P) correlates as well with UAS and EM as they correlate internally. All correlations were significant ($p < 0.05$), where significance is derived from ρ.

	ρ	p-value
P/UAS	**-0.5670**	0.0172
P/EM	-0.5464	0.0217
UAS/EM	0.5510	0.0206

Of course this result only says that the three metrics are correlated, but not which of the three is more useful. Since they capture different aspects and have different weaknesses, as argued above, we suggest to use all three metrics in liaison. It would again be interesting to correlate these metrics with pipeline evaluations.

6 Conclusion

We have introduced the notion of derivation order perplexity which is much better at predicting state-of-the-art parsing accuracy than treebank size only. It was shown that there is a strong correlation between state-of-the-art parsing accuracy and derivation order perplexity across morphologically poor and rich languages. Derivation order perplexity can also be used as a metric for parser evaluation and has the advantages that it captures global aspects of syntactic analyses and is fine-grained enough to obtain statistically significant results on small data sets.

References

[1] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*, 2006.

[2] John Caroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: a survey and a new proposal. In *LREC*, 1998.

[3] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.

[4] Michel Galley and Christopher Manning. Quadratic-time dependency parsing for machine translation. In *ACL*, Singapore, 2009.

[5] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Nonprojective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*, 2005.

[6] Joakim Nivre. Data-driven dependency parsing across languages and domains: perspectives from the CoNLL 2007 shared task. In *IWPT*, 2007.

[7] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *ACL-IJCNLP*, 2009.

[8] Joakim Nivre. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th international conference on parsing technologies*, pages 73–76, Paris, France, 2009.

[9] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL'07*, pages 915–932, Prague, Czech Republic, 2007.

[10] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[11] Laura Rimell, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *EMNLP*, Singapore, Singapore, 2009.

[12] Andreas Stolcke. SRILM – an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*, 2002.

[13] Reut Tsarfaty, Djame Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. Statistical parsing of morphologically rich languages. In *The 1st Workshop on Statistical Parsing of Morphologically Rich Languages, NAACL*, 2010.

[14] Deniz Yuret, Aydin Han, and Zehra Turgut. SemEval-2010 Task 12: parser evaluation using textual entailments. In *SemEval*, 2010.