Optimizing Grammars for Minimum Dependency Length

Daniel Gildea
Computer Science Dept.
University of Rochester
Rochester, NY 14627

David Temperley
Eastman School of Music
University of Rochester
Rochester, NY 14604

Abstract

We examine the problem of choosing word order for a set of dependency trees so as to minimize total dependency length. We present an algorithm for computing the optimal layout of a single tree as well as a numerical method for optimizing a grammar of orderings over a set of dependency types. A grammar generated by minimizing dependency length in unordered trees from the Penn Treebank is found to agree surprisingly well with English word order, suggesting that dependency length minimization has influenced the evolution of English.

1 Introduction

Dependency approaches to language assume that every word in a sentence is the dependent of one other word (except for one word, which is the global head of the sentence), so that the words of a sentence form an acyclic directed graph. An important principle of language, supported by a wide range of evidence, is that there is preference for dependencies to be short. This has been offered as an explanation for numerous psycholinguistic phenomena, such as the greater processing difficulty of object relative clauses versus subject relative clauses (Gibson, 1998). Dependency length minimization is also a factor in ambiguity resolution: listeners prefer the interpretation with shorter dependencies. Statistical parsers make use of features that capture dependency length (e.g. an adjacency feature in Collins (1999), more explicit length features in McDonald et al. (2005) and Eisner and Smith (2005)) and thus learn to favor parses with shorter dependencies.

In this paper we attempt to measure the extent to which basic English word order chooses to minimize dependency length, as compared to average dependency lengths under other possible grammars. We first present a linear-time algorithm for finding the ordering of a single dependency tree with shortest total dependency length. Then, given that word order must also be determined by grammatical relations, we turn to the problem of specifying a grammar in terms of constraints over such relations. We wish to find the set of ordering constraints on dependency types that minimizes a corpus's total dependency length. Even assuming that dependency trees must be projective, this problem is NP-complete,[1] but we find that numerical optimization techniques work well in practice. We reorder unordered dependency trees extracted from corpora and compare the results to English in terms of both the resulting dependency length and the strings that are produced. The optimized order constraints show a high degree of similarity to English, suggesting that dependency length minimization has influenced the word order choices of basic English grammar.

2 The Dependency Length Principle

This idea that dependency length minimization may be a general principle in language has been discussed by many authors. One example concerns the

[1]English has crossing (non-projective) dependencies, but they are believed to be very infrequent. McDonald et al. (2005) report that even in Czech, commonly viewed as a non-projective language, fewer than 2% of dependencies violate the projectivity constraint.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 184–191,
Prague, Czech Republic, June 2007.

well-known principle that languages tend to be predominantly "head-first" (in which the head of each dependency is on the left) or "head-last" (where it is on the right). Frazier (1985) suggests that this might serve the function of keeping heads and dependents close together. In a situation where each word has exactly one dependent, it can be seen that a "head-first" arrangement achieves minimal dependency length, as each link has a length of one.

We will call a head-first dependency "right-branching" and a head-last dependency "left-branching"; a language in which most or all dependencies have the same branching direction is a "same-branching" language.

Another example of dependency length minimization concerns situations where a head has multiple dependents. In such cases, dependency length will be minimized if the shorter dependent is placed closer to the head. Hawkins (1994) has shown that this principle is reflected in grammatical rules across many languages. It is also reflected in situations of choice; for example, in cases where a verb is followed by a prepositional phrase and a direct object NP, the direct object NP will usually be placed first (closer to the verb) but if it is longer than the PP, it is often placed second.

While one might suppose that a "same-branching" language is optimal for dependency-length minimization, this is not in fact the case. If a word has several dependents, placing them all on the same side causes them to get in the way of each other, so that a more 'balanced" configuration – with some dependents on each side – has lower total dependency length. It is particularly desirable for one or more one-word dependent phrases to be "opposite-branching" (in relation to the prevailing branching direction of the language); opposite-branching of a long phrase tends to cause a long dependency from the head of the phrase to the external head.

Exactly this pattern has been observed by Dryer (1992) in natural languages. Dryer argues that, while most languages have a predominant branching direction, phrasal (multi-word) dependents tend to adhere to this prevailing direction much more consistently than one-word dependents, which frequently branch opposite to the prevailing direction of the language. English reflects this pattern quite strongly: While almost all phrasal dependents are right-branching (prepositional phrases, objects of prepositions and verbs, relative clauses, etc.), some 1-word categories are left-branching, notably determiners, noun modifiers, adverbs (sometimes), and attributive adjectives.

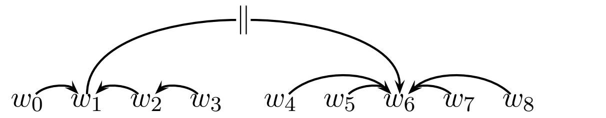

Figure 1: Separating a dependency link into two pieces at a subtree boundary.

This linguistic evidence strongly suggests that languages have been shaped by principles of dependency length minimization. One might wonder how close natural languages are to being optimal in this regard. To address this question, we extract unordered dependency graphs from English and consider different algorithms, which we call Dependency Linearization Algorithms (DLAs), for ordering the words; our goal is to find the algorithm that is optimal with regard to dependency length minimization. We begin with an "unlabeled" DLA, which simply minimizes dependency length without requiring consistent ordering of syntactic relations. We then consider the more realistic case of a "labeled" DLA, which is required to have syntactically consistent ordering.

Once we find the optimal DLA, two questions can be asked. First, how close is dependency length in English to that of this optimal DLA? Secondly, how similar is the optimal DLA to English in terms of the actual rules that arise?

3 The Optimal Unlabeled DLA

Finding linear arrangements of graphs that minimize total edge length is a classic problem, NP-complete for general graphs but with an $O(n^{1.6})$ algorithm for trees (Chung, 1984). However, the traditional problem description does not take into account the projectivity constraint of dependency grammar. This constraint simplifies the problem; in this section we show that a simple linear-time algorithm is guaranteed to find an optimal result.

A natural strategy would be to apply dynamic programming over the tree structure, observing that total dependency length of a linearization can be broken into the sum of links below any node w in the tree, and the sum of links outside the node, by which we mean all links not connected to dependents of the node. These two quantities interact only through the position of w relative to the rest of its descendants, meaning that we can use this position as our dynamic programming state, compute the optimal layout of each subtree given each position of the head within the subtree, and combine subtrees bottom-up to compute the optimal linearization for the entire sentence.

This can be further improved by observing that the total length of the outside links depends on the position of w only because it affects the length of the link connecting w to its parent. All other outside links either cross above all words under w, and depend only on the total size of w's subtree, or are entirely on one side of w's subtree. The link from w to its parent is divided into two pieces, whose lengths add up to the total length of the link, by slicing the link where it crosses the boundary from w's subtree to the rest of the sentence. In the example in Figure 1, the dependency from w_1 to w_6 has total length five, and is divided in to two components of length 2.5 at the boundary of w_1's subtree. The length of the piece over w's subtree depends on w's position within that subtree, while the other piece does not depend on the internal layout of w's subtree. Thus the total dependency length for the entire sentence can be divided into:

1. the length of all links within w's subtree plus the length of the first piece of w's link to its parent, i.e. the piece that is above descendants of w.

2. the length of the remaining piece of w's link to its parent plus the length of all links outside w.

where the second quantity can be optimized independently of the internal layout of w's subtree. While the link from w to its parent may point either to the right or left, the optimal layout for w's subtree given that w attaches to its left must be the mirror image of the optimal layout given that w attaches to its right. Thus, only one case need be considered, and the optimal layout for the entire sentence can be computed from the bottom up using just one dynamic programming state for each node in the tree.

We now go on to show that, in computing the ordering of the d_i children of a given node, not all $d_i!$ possibilities need be considered. In fact, one can simply order the children by adding them in increasing order of size, going from the head outwards, and alternating between adding to the left and right edges of the constituent.

The first part of this proof is the observation that, as we progress from the head outward, to either the left or the right, the head's child subtrees must be placed in increasing order of size. If any two adjacent children appear with the smaller one further from the head, we can swap the positions of these two children, reducing the total dependency length of the tree. No links crossing over the two children will change in length, and no links within either child will change. Thus only the length of the links from the two children will change, and as the link connecting the outside child now crosses over a shorter intermediate constituent, the total length will decrease.

Next, we show that the two longest children must appear on opposite sides of the head in the optimal linearization. To see this, consider the case where both child i (the longest child) and child $i-1$ (the second longest child) appear on the same side of the head. From the previous result, we know that $i-1$ and i must be the outermost children on their side. If there are no children on the other side of the head, the tree can be improved by moving either i or $i-1$ to the other side. If there is a child on the other side of the head, it must be smaller than both i and $i-1$, and the tree can be improved by swapping the position of the child from the other side and child $i-1$.

Given that the two largest children are outermost and on opposite sides of the head, we observe that the sum of the two links connecting these children to the head does not depend on the arrangement of the first $i-2$ children. Any rearrangement that decreases the length of the link to the left of the head must increase the length of the link to the right of the head by the same amount. Thus, the optimal layout of all i children can be found by placing the two largest children outermost and on opposite sides, the next two largest children next outermost and on op-

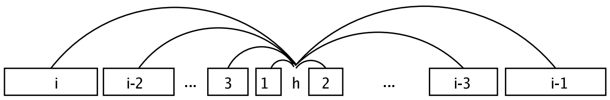

Figure 2: Placing dependents on alternating sides from inside out in order of increasing length.

DLA	Length
Optimal	33.7
Random	76.1
Observed	47.9

Table 1: Dependency lengths for unlabeled DLAs.

posite sides, and so on until only one or zero children are left. If there are an odd number of children, the side of the final (smallest) child makes no difference, because the other children are evenly balanced on the two sides so the last child will have the same dependency-lengthening effect whichever side it is on.

Our pairwise approach implies that there are many optimal linearizations, $2^{\lfloor i/2 \rfloor}$ in fact, but one simple and optimal approach is to alternate sides as in Figure 2, putting the smallest child next to the head, the next smallest next to the head on the opposite side, the next outside the first on the first side, and so on.

So far we have not considered the piece of the link from the head to its parent that is over the head's subtree. The argument above can be generalized by considering this link as a special child, longer than the longest real child. By making the special child the longest child, we will be guaranteed that it will be placed on the outside, as is necessary for a projective tree. As before, the special child and the longest real child must be placed outermost and on opposite sides, the next two longest children immediately within the first two, and so on.

Using the algorithm from the previous section, it is possible to efficiently compute the optimal dependency length from English sentences. We take sentences from the Wall Street Journal section of the Penn Treebank, extract the dependency trees using the head-word rules of Collins (1999), consider them to be unordered dependency trees, and linearize them to minimize dependency length. Automatically extracting dependencies from the Treebank can lead to some errors, in particular with complex compound nouns. Fortunately, compound nouns tend to occur at the leaves of the tree, and the head rules are reliable for the vast majority of structures.

Results in Table 1 show that observed dependency lengths in English are between the minimum achievable given the unordered dependencies and the length we would find given a random ordering, and are much closer to the minimum. This already suggests that minimizing dependency length has been a factor in the development of English. However, the optimal "language" to which English is being compared has little connection to linguistic reality. Essentially, this model represents a free word-order language: Head-modifier relations are oriented without regard to the grammatical relation between the two words. In fact, however, word order in English is relatively rigid, and a more realistic experiment would be to find the optimal algorithm that reflects consistent syntactic word order rules. We call this a "labeled" DLA, as opposed to the "unlabeled" DLA presented above.

4 Labeled DLAs

In this section, we consider linearization algorithms that assume fixed word order for a given grammatical relation, but choose the order such as to minimize dependency length over a large number of sentences. We represent grammatical relations simply by using the syntactic categories of the highest constituent headed by (maximal projection of) the two words in the dependency relation. Due to sparse data concerns, we removed all function tags such as TMP (temporal), LOC (locative), and CLR (closely related) from the treebank. We made an exception for the SBJ (subject) tag, as we thought it important to distinguish a verb's subject and object for the purposes of choosing word order. Looking at a head and its set of dependents, the complete ordering of all dependents can be modeled as a context-free grammar rule over a nonterminal alphabet of maximal projection categories. A fixed word-order language will have only one rule for each set of nonterminals appearing in the right-hand side.

Searching over all such DLAs would be exponentially expensive, but a simple approximation of the

DLA	Dep. len. / % correct order
random	76.1 / 40.5
extracted from optimal	61.6 / 55.4
weights from English	50.9 / 82.2
optimized weights	42.5 / 64.9

Table 2: Results for different methods of linearizing unordered trees from section 0 of the Wall Street Journal corpus. Each result is given as average dependency length in words, followed by the percentage of heads (with at least one dependent) having all dependents correctly ordered.

optimal labeled DLA can found using the following procedure:

1. Compute the optimal layout of all sentences in the corpus using the unlabeled DLA.

2. For each combination of a head type and a set of child types, count the occurrences of each ordering.

3. Take the most frequent ordering for each set as the order in the new DLA.

In the first step we used the alternating procedure from the previous section, with a modification for the fixed word-order scenario. In order to make the order of a subtree independent of the direction in which it attaches to its parent, dependents were placed in order of length on alternating sides of the head from the inside out, always starting with the shortest dependent immediately to the left of the head.

Results in Table 2 (first two lines) show that a DLA using rules extracted from the optimal layout matches English significantly better than a random DLA, indicating that dependency length can be used as a general principle to predict word order.

4.1 An Optimized Labeled DLA

While the DLA presented above is a good deal better than random (in terms of minimizing dependency length), there is no reason to suppose that it is optimal. In this section we address the issue of finding the optimal labeled DLA.

If we model a DLA as a set of context-free grammar rules over dependency types, specifying a fixed ordering for any set of dependency types attaching to a given head, the space of DLAs is enormous, and the problem of finding the optimal DLA is a difficult one. One way to break the problem down is to model the DLA as a set of weights for each type of dependency relation. Under this model the word order is determined by placing all dependents of a word in order of increasing weight from left to right. This reduces the number of parameters of the model to T, if there are T dependency types, from T^k if a word may have up to k dependents. It also allows us to naturally capture statements such as "a noun phrase consists of a determiner, then (possibly) some adjectives, the head noun, and then (possibly) some prepositional phrases", by, for example, setting the weight for NP→DT to -2, NP→JJ to -1, and NP→PP to 1. We assume the head itself has a weight of zero, meaning negatively weighted dependents appear to the head's left, and positively weighted dependents to the head's right.

4.1.1 A DLA Extracted from English

As a test of whether this model is adequate to represent English word order, we extracted weights for the Wall Street Journal corpus, used them to reorder the same set of sentences, and tested how often words with at least one dependent were assigned the correct order. We extracted the weights by assigning, for each dependency relation in the corpus, an integer according to its position relative to the head, -1 for the first dependent to the left, -2 for the second to the left, and so on. We averaged these numbers across all occurrences of each dependency type. The dependency types consisted of the syntactic categories of the maximal projections of the two words in the dependency relation.

Reconstructing the word order of each sentence from this weighted DLA, we find that 82% of all words with at least one dependent have all dependents ordered correctly (third line of Table 2). This is significantly higher than the heuristic discussed in the previous section, and probably as good as can be expected from such a simple model, particularly in light of the fact that there is some choice in the word order for most sentences (among adjuncts for example) and that this model does not take the lengths of the individual constituents into account at all.

We now wish to find the set of weights that minimize the dependency length of the corpus. While the size of the search space is still too large to search exhaustively, numerical optimization techniques can be applied to find an approximate solution.

4.1.2 NP-Completeness

The problem of finding the optimum weighted DLA for a set of input trees can be shown to be NP-complete by reducing from the problem of finding a graph's minimum Feedback Arc Set, one of the 21 classic problems of Karp (1972). The input to the Feedback Arc Set problem is a directed graph, for which we wish to find an ordering of vertices such that the smallest number of edges point from later to earlier vertices in the ordering. Given an instance of this problem, we can create a set of dependency trees such that each feedback arc in the original graph causes total dependency length to increase by one, if we identify each dependency type with a vertex in the original problem, and choose weights for the dependency types according to the vertex order.[2]

4.1.3 Local Search

Our search procedure is to optimize one weight at a time, holding all others fixed, and iterating through the set of weights to be set. The objective function describing the total dependency length of the corpus is piecewise constant, as the dependency length will not change until one weight crosses another, causing two dependents to reverse order, at which point the total length will discontinuously jump. Non-differentiability implies that methods based on gradient ascent will not apply. This setting is reminiscent of the problem of optimizing feature weights for reranking of candidate machine translation outputs, and we employ an optimization technique similar to that used by Och (2003) for machine translation. Because the objective function only changes at points where one weight crosses another's value, the set of segments of weight values with different values of the objective function can be exhaustively enumerated. In fact, the only significant points are the values of other weights for dependency types which occur in the corpus attached to the same head as the dependency being optimized. We build a table of interacting dependencies as a preprocessing step on the data, and then when optimizing a weight, consider the sequence of values between consecutive interacting weights. When computing the total corpus dependency length at a new weight value, we can further speed up computation by reordering only those sentences in which a dependency type is used, by building an index of where dependency types occur as another preprocessing step.

[2] We omit details due to space.

Training Data	Test Data WSJ	Test Data Swbd
WSJ	42.5 / 64.9	12.5 / 63.6
Swbd	43.9 / 59.8	12.2 / 58.7

Table 3: Domain effects on dependency length minimization: each result is formatted as in Table 2.

This optimization process is not guaranteed to find the global maximum (for this reason we call the resulting DLA "optimized" rather than "optimal"). The procedure is guaranteed to converge simply from the fact that there are a finite number of objective function values, and the objective function must increase at each step at which weights are adjusted.

We ran this optimization procedure on section 2 through 21 of the Wall Street Journal portion of the Penn Treebank, initializing all weights to random numbers between zero and one. This initialization makes all phrases head-initial to begin with, and has the effect of imposing a directional bias on the resulting grammar. When optimization converges, we obtain a set of weights which achieves an average dependency length of 40.4 on the training data, and 42.5 on held-out data from section 0 (fourth line of Table 2). While the procedure is unsupervised with respect to the English word order (other than the head-initial bias), it is supervised with respect to dependency length minimization; for this reason we report all subsequent results on held-out data. While random initializations lead to an initial average dependency length varying from 60 to 73 with an average of 66 over ten runs, all runs were within $\pm.5$ of one another upon convergence. When the order of words' dependents was compared to the real word order on held-out data, we find that 64.9% of words

Training Sents	Dep. len. / % correct order
100	13.70 / 54.38
500	12.81 / 57.75
1000	12.59 / 58.01
5000	12.34 / 55.33
10000	12.27 / 55.92
50000	12.17 / 58.73

Table 4: Average dependency length and rule accuracy as a function of training data size, on Switchboard data.

with at least one dependent have the correct order.

4.2 Domain Variation

Written and spoken language differ significantly in their structure, and one of the most striking differences is the much greater average sentence length of formal written language. The Wall Street Journal is not representative of typical language use. Language was not written until relatively recently in its development, and the Wall Street Journal in particular represents a formal style with much longer sentences than are used in conversational speech. The change in the lengths of sentences and their constituents could make the optimized DLA in terms of dependency length very different for the two genres.

In order to test this effect, we performed experiments using both the Wall Street Journal (written) and Switchboard (conversational speech) portions of the Penn Treebank, and compared results with different training and test data. For Switchboard, we used the first 50,000 sentences of sections 2 and 3 as the training data, and all of section 4 as the test data.

We find relatively little difference in dependency length as we vary training data between written and spoken English, as shown in Table 3. For the accuracy of the resulting word order, however, training on Wall Street Journal outperforms Switchboard even when testing on Switchboard, perhaps because the longer sentences in WSJ provide more information for the optimization procedure to work with.

4.3 Learning Curve

How many sentences are necessary to learn a good set of dependency weights? Table 4 shows results for Switchboard as we increase the number of sentences provided as input to the weight optimization procedure. While the average dependency length on held-out test data slowly decreases with more data, the percentage of correctly ordered dependents is less well-behaved. It turns out that even 100 sentences are enough to learn a DLA that is nearly as good as one derived from a much larger dataset.

Label	Interpretation	Weight
S→NP	verb - object NP	0.037
S→NP-SBJ	verb - subject NP	-0.022
S→PP	verb - PP	0.193
NP→DT	object noun - determiner	-0.070
NP-SBJ→DT	subject noun - determiner	-0.052
NP→PP	obj noun - PP	0.625
NP-SBJ→PP	subj noun - PP	0.254
NP→SBAR	obj noun - rel. clause	0.858
NP-SBJ→SBAR	subject noun - rel. clause	-0.110
NP→JJ	obj noun - adjective	0.198
NP-SBJ→JJ	subj noun - adjective	-0.052

Table 5: Sample weights from optimized DLA. Negatively weighted dependents appear to the left of their head.

4.4 Comparing the Optimized DLA to English

We have seen that the optimized DLA matches English text much better than a random DLA and that it achieves only a slightly lower dependency length than English. It is also of interest to compare the optimized DLA to English in more detail. First we examine the DLA's tendency towards "opposite-branching 1-word phrases". English reflects this principle to a striking degree: on the WSJ test set, 79.4 percent of left-branching phrases are 1-word, compared to only 19.4 percent of right-branching phrases. The optimized DLA also reflects this pattern, though somewhat less strongly: 75.5 percent of left-branching phrases are 1-word, versus 36.7 percent of right-branching phrases.

We can also compare the optimized DLA to English with regard to specific rules. As explained earlier, the optimal DLA's rules are expressed in the form of weights assigned to each relation, with positive weights indicating right-branching placement. Table 5 shows some important rules. The middle column shows the syntactic situation in which the relation normally occurs. We see, first of all, that object NPs are to the right of the verb and subject NPs are to the left, just like in English. PPs are also the right of verbs; the fact that the weight is greater than for NPs indicates that they are placed further to the right, as they normally are in English. Turning to the internal structure of noun phrases, we see that determiners are to the left of both object and subject nouns; PPs are to the right of both object and subject nouns. We also find some differences with English, however. Clause modifiers of nouns (these are mostly relative clauses) are to the right of object nouns, as in English, but to the left of subject nouns; adjectives are to the left of subject nouns, as in English, but to the right of object nouns. Of course, these differences partly arise from the fact that we treat NP and NP-SBJ as distinct whereas English does not (with regard to their internal structure).

5 Conclusion

In this paper we have presented a dependency linearization algorithm which is optimized for minimizing dependency length, while still maintaining consistent positioning for each grammatical relation. The fact that English is so much lower than the random DLAs in dependency length gives suggests that dependency length minimization is an important general preference in language. The output of the optimized DLA also proves to be much more similar to English than a random DLA in word order. An informal comparison of some important rules between English and the optimal DLA reveals a number of striking similarities, though also some differences.

The fact that the optimized DLA's ordering matches English on only 65% of words shows, not surprisingly, that English word order is determined by other factors in addition to dependency length minimization. In some cases, ordering choices in English are underdetermined by syntactic rules. For example, a manner adverb may be placed either before the verb or after ("He ran quickly / he quickly ran"). Here the optimized DLA requires a consistent ordering while English does not. One might suppose that such syntactic choices in English are guided at least partly by dependency length minimization, and indeed there is evidence for this; for example, people tend to put the shorter of two PPs closer to the verb (Hawkins, 1994). But there are also other factors involved – for example, the tendency to put "given" discourse elements before "new" ones, which has been shown to play a role independent of length (Arnold et al., 2000).

In other cases, the optimized DLA allows more fine-grained choices than English. For example, the optimized DLA treats NP and NP-SBJ as different; this allows it to have different syntactic rules for the two cases – a possibility that it sometimes exploits, as seen above. No doubt this partly explains why the optimized DLA achieves lower dependency length than English.

Acknowledgments This work was supported by NSF grants IIS-0546554 and IIS-0325646.

References

J. E. Arnold, T. Wasow, T. Losongco, and R. Ginstrom. 2000. Heaviness vs. newness: the effects of structural complexity and discourse status on constituent ordering. *Language*, 76:28–55.

F. R. K. Chung. 1984. On optimal linear arrangements of trees. *Computers and Mathematics with Applications*, 10:43–60.

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

Matthew Dryer. 1992. The Greenbergian word order correlations. *Language*, 68:81–138.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the International Workshop on Parsing Technologies (IWPT)*, pages 30–41.

Lyn Frazier. 1985. Syntactic complexity. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 129–189. Cambridge University Press, Cambridge.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76.

John Hawkins. 1994. *A Performance Theory of Order and Constituency*. Cambridge University Press, Cambridge, UK.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL-03*.