Impact of Quality and Quantity of Corpora on Stochastic Generation

Srinivas Bangalore
AT&T Labs – Research
180 Park Ave
Florham Park, NJ 07932
USA
srini@research.att.com

John Chen[*]
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
jchen@cis.udel.edu

Owen Rambow
AT&T Labs – Research
180 Park Ave
Florham Park, NJ 07932
USA
rambow@research.att.com

Abstract

Recently, there has been some interest in using stochastic approaches in generation. However, there has been little research so far on the question of how the quality, size, and genre of training corpora influence the quality of stochastic generation components. In this paper, we investigate these issues using the FERGUS system. FERGUS uses two distinct stochastic models, a tree model which refers to a grammar, and a linear language model. We use automatic grammar extraction techniques to extract grammars from different-sized tree banks, and then use these extracted grammars to train the tree models. We also investigate the impact of the quality of the annotated corpus, by using a hand-annotated corpus as well as an automatically annotated corpus. Our results show that automatic grammar extraction is a viable alternative to hand crafted grammars for generation; furthermore, as expected, both quality and size of the training corpus matter

1 Introduction

Recently, there has been some interest in using stochastic approaches in generation (Knight and Hatzivassiloglou, 1995; Langkilde and Knight, 1998; Langkilde, 2000; Oh and Rudnicky, 2000; Ratnaparkhi, 2000; Bangalore and Rambow, 2000). For generation, stochastic methods promise to be superior to hand-crafted generators in (at least) two different contexts:

- When the range of output to be generated is wide, for example in general-purpose machine translation systems.
- When a generation system needs to be created very quickly.

However, there has been little research so far on the question of how the quality, size, and genre of training corpora influence the quality of stochastic generation components. These questions are crucial: for both wide-range output and for rapid development, we need to know whether the quality of the generator can be brought to an adequate level by increasing the size of the training corpus.

In this paper, we present answers to these questions using the framework developed for the FERGUS system (Bangalore and Rambow, 2000). In FERGUS, an input dependency tree representing lexical predicate-argument structure is first annotated for syntactic information using a stochastic tree model. This syntactic information is in the form of supertags, i.e., references to trees from a Tree Adjoining Grammar (TAG). The dependency tree annotated with supertags allows several linearizations; FERGUS uses a standard linear language model to choose the best of these linearizations. Note the presence of two distinct stochastic models, the tree model which refers to a grammar, and the linear language model. As we have shown in previous work, the use of both models increases the performance of the system over a version of the system with only one of the two models. However, the use of two models comes at a cost: the tree model requires both the existence of a hand-crafted grammar, and of a corpus which has been syntactically annotated with reference to the hand-crafted grammar. If we wish to generate in new languages, or if we wish to generate in new sublanguages of previously covered languages, such an approach is unappealing because of the required amount of hand-crafting (of grammars and annotated corpora).

We address this problem by using automatic grammar extraction techniques. We use the TAG-extraction algorithm of Chen and Vijay-Shanker (2000) to extract grammars from different-sized annotated corpora, and then use these extracted grammars to train the tree models. While performance improvements still require the existence of a syntactically annotated corpus, we no longer need a hand-crafted grammar. Furthermore, we do not even require that the annotated corpus be hand-annotated: we use a corpus which has been syntactically annotated by an automatic parser. To our knowledge, this is the first time that automatically extracted grammars are used for generation. Using automatic extraction of generation grammars, we then investigate the relation between corpus quality and size on the one hand and output quality on the other hand by learning different tree and language mod-

[*] This work was done when the author was at AT&T Labs-Research, Shannon Laboratories, 180 Park Ave, Florham Park NJ 07932

els. As expected, our results show that both quality and size matter, and that the quality of the syntactically annotated corpus can be traded against its size.

The paper is structured as follows. In Section 2, we present the grammar extraction algorithm of Chen and Vijay-Shanker (2000). We then present our generation system, FERGUS (Section 3). We investigate the issue of the quality of the training corpus in Section 4. The issue of size of corpora is discussed in Section 5 and in Section 6, concentrating first on small corpora, and then on very large corpora. We conclude with a summary and an outlook.

2 Automated TAG Extraction from Bracketed Corpora

There are a number of approaches for the extraction of Tree Adjoining Grammars (TAGs) from corpora annotated in the style of the Penn Treebank (Marcus et al. (1993)), as work by Neumann (1998), Xia (1999), and Chiang (2000) show. In fact, in Chen and Vijay-Shanker (2000) alone, no fewer than eight different kinds of extracted grammars are compared. For the following experiments, we adopt one out of the many approaches described in Chen and Vijay-Shanker (2000) which we describe here. As background, we review useful TAG terminology and discuss the linguistic principles that guide the formation of both manually constructed TAG grammars and the automatically extracted variety; for a more complete introduction, see (Abeillé and Rambow, 2000). Subsequently, We describe the details of the grammar extraction algorithm and the modifications that are required in order to make the algorithm work with different kinds of corpora.

2.1 Background

A TAG G is defined as a set of **elementary trees** T which are partitioned into a set I of **initial trees** and a set A of **auxiliary trees**. If GC is **lexicalized**, the frontier of each elementary tree includes a **lexical anchor**; the other nodes on the frontier are **substitution nodes**, and in the case of an auxiliary tree, one node on the frontier will be a **foot node**. The foot node of a tree β is labeled identically with the root node of β. The **spine** of an auxiliary tree is the path from its root to its foot node. It is to be distinguished from the **trunk** of an elementary tree which is the path from its root node to the lexical anchor.

Although the formalism of TAG allows wide latitude in how trees in T may be defined, several linguistic principles generally guide their formation. First, dependencies including long distance dependencies are typically localized in the same elementary tree by appropriate grouping of syntactically or semantically related elements; i.e. positions for complements of a lexical item are included in the same tree as the lexical item, as shown in Figure 1(b). Second, recursion is factored into separate auxiliary trees. There are **modifier auxiliary trees** which generally represent syntactic adjuncts; the foot nodes in these trees represent the objects of modification as shown in Figure 1(c). There are also **predicative auxiliary trees**; the foot nodes in these trees represent sentential complements. They are used (rather than initial trees) in order to handle long-distance extraction.

2.2 Extraction from the Penn Treebank

Given a bracketed sentence S in the corpus, an elementary tree γ lexicalized by a word $w \in S$ is created as follows. First, a **head percolation table** is used to determine the trunk of γ. Introduced in Magerman (1995), a head percolation table assigns to each node in S a **headword** using local structural information. The trunk of γ is defined to be that path through the tree for S whose nodes are labeled with syntactic projections from the headword w. See Figure 2(a). Each node η' that is immediately dominated by a node η on the trunk of γ may either be itself on the trunk, a complement of w, or an adjunct of w. If η' is a complement of w, then the node is made into a substitution node of γ. (The subtree rooted at η' will be associated with a different headword, that of the complement.) If η' is an adjunct of w, then it belongs to another (auxiliary) tree β which modifies γ.

It is therefore necessary to determine a node's status as either complement or adjunct. The procedure used by our algorithm is based on a similar one that is used by Collins (1997). Like Collins (1997), it bases its decision on the node's label, its semantic tags (see Marcus et al. (1993)), and local structural information. For example, a node that is labeled NP-DIR would be labeled as an adjunct because of the semantic tag DIR, signifying an adverbial that answers the questions "from where?" or "to where?" The main difference lies in our attempt to use lexical predicate-argument structure as a basis for determining the shape of the trees in our grammar. For instance, *wh*-moved constituents are treated by our procedure as complements of their head and therefore positions for moved elements may be included in verbal trees. Our procedure operates in two steps. In the first step, the label and semantic tags of a node η and the parent node of η are used as an index into a manually constructed **complement-adjunct table** which determines complement or adjunct status. The table is sparse; should the index not be found in the table then the second step of the procedure is invoked. This second step uses only the semantic tags of node η to determine the complement or adjunct status. Should η lack any semantic

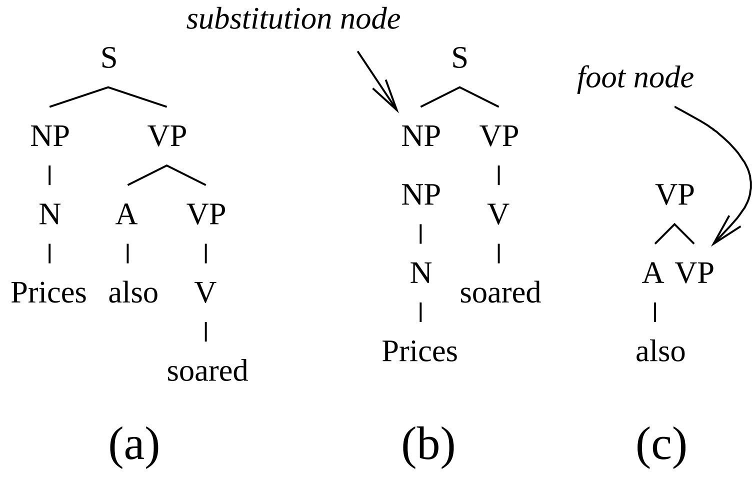

Figure 1: (a) Sentential structure (b) Localizing argument dependencies in the same elementary tree (c) Each instance of recursion is factored into a separate elementary tree.

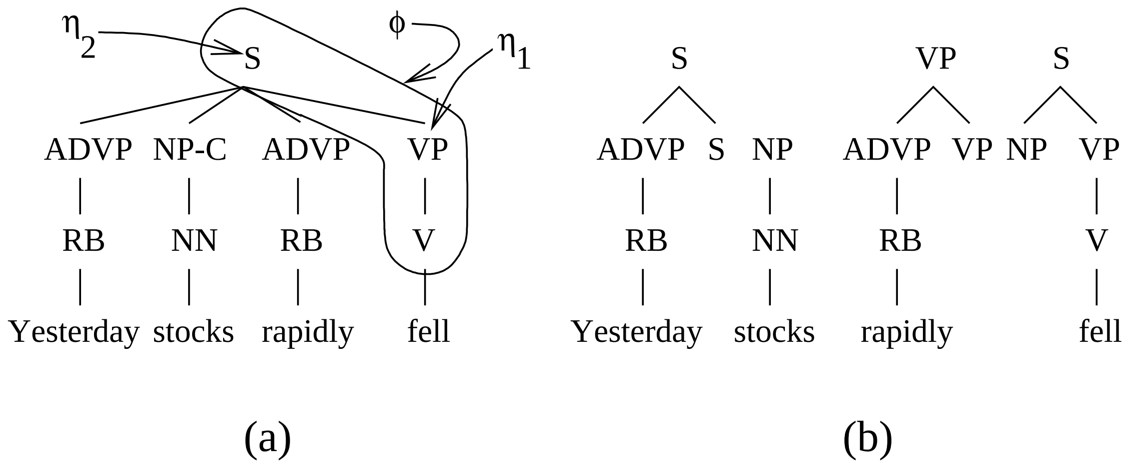

Figure 2: (a) Original bracketing with head sibling η_1 and its parent η_2 both on the trunk (circled) associated with the headword "fell." (b) Extracted trees.

tags, it is labeled as an adjunct.

A recursive procedure is used to extract trees bottom-up given a particular bracketed sentence S. As shown in Figure 2, among all of the children of node η_2, one child η_1 is selected using the head percolation table so that the trunk ϕ associated with η_1 is extended to η_2. η_1's siblings are subsequently marked as either complement or adjunct. Complements are attached to the trunk ϕ as substitution nodes and the trees that they dominate become initial trees. Adjuncts are factored into modifier auxiliary trees.

Besides modifier auxiliary trees, there are predicative auxiliary trees which are generated as follows. During the bottom-up extraction of trees, suppose trunk ϕ has a node η that shares the same label as another node η' where η' is a complement, not on ϕ, but is immediately dominated by a node on ϕ. In this case, a predicative auxiliary tree is extracted where η is its root, η' is its foot and with ϕ serving as its trunk. Subsequently, the path ϕ' dominated by η' becomes a candidate for being extended further.

2.3 Extension to Other Bracketed Corpora

Although we have described extraction of TAG specifically from the Penn Treebank, it is not difficult to adapt this procedure to other bracketed corpora. Alterations to this procedure are basically necessitated by the differences between label sets and bracketing conventions of various bracketed corpora. Given a bracketed corpus with a novel label set, configurational relationships between different labels as they appear in the corpus need to be investigated. This allows the construction of a new head percolation table in order to specify the trunks of the extracted elementary trees. This also allows the formulation of a new complement-adjunct table in order to specify which nodes are factored into auxiliary trees. It is also necessary to specify a new list of empty elements to be pruned as well as the mapping to a set of simplified labels, if these are required. These methods were employed in order to change the algorithm to extract grammars from the BLLIP Treebank (Charniak, 2000).

3 The FERGUS Generation System

FERGUS is composed of three modules: the Tree Chooser, the Unraveler, and the Linear Precedence (LP) Chooser (Figure 3). The architecture of FERGUS is shown in Figure 3. The input to the system is a dependency tree as shown in Figure 4. Note that the nodes are unordered and are labeled only with lexemes, not with any sort of syntactic annotations.[1] The Tree Chooser uses a **stochastic tree model** to choose syntactic properties for the nodes in the input structure. These properties are expressed as **supertags**, i.e., as names of trees in a Tree Adjoining Grammar. We will refer to this grammar, which must be specified in advance, as the **generation grammar**. This step can be seen as analogous to "supertagging" (Bangalore and Joshi, 1999), except that now supertags (i.e., names of trees which encode the syntactic properties of a lexical head) must be found for words in a tree rather than for words in a linear sequence. The Tree Chooser makes the simplifying assumptions that the choice of a tree for a node depends only on its daughter nodes, thus allowing for a top-down algorithm. The Tree Chooser draws on a tree model, which is a statistical model trained from the syntactic dependencies present in the **tree model corpus**. In the remainder of this paper, we will be investigating the effect of using different tree model corpora.

The supertagged tree which is output from the Tree Chooser still does not fully determine the surface string, because there typically are different ways to attach a daughter node to her mother (for example, an adverb can be placed in different positions with respect to its verbal head). The Unraveler therefore uses the generation grammar to produce a lattice of all possible linearizations that are compatible with the supertagged tree. Specifically, the daughter nodes are ordered with respect to the head at each level of the derivation tree. In cases where the generation grammar allows a daughter node to

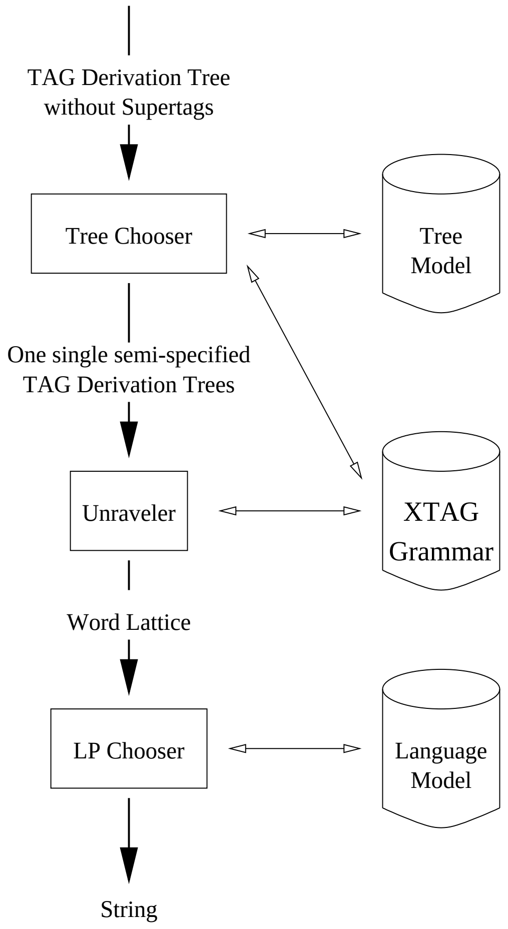

Figure 3: Architecture of FERGUS

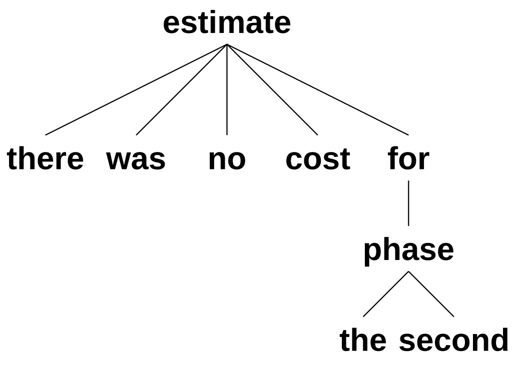

Figure 4: Input to FERGUS

be attached at more than one place in the mother supertag (as is the case in our example for *was* and *for*; generally, such underspecification occurs with adjuncts and with arguments if their syntactic role is not specified), a disjunction of all these positions is assigned to the daughter node. A bottom-up algorithm then constructs a lattice that encodes the

[1] In the system that we used in the experiments described in this paper, all words (including function words) need to be present in the input representation, fully inflected. Furthermore, there is no indication of syntactic role at all. This may be unrealistic for some applications – however, it provides a convenient baseline.

Grammar	Size of Grammar	StringAccuracy
Penn Treebank (XTAG)	444	0.742
Penn Treebank (Extracted)	3063	0.749
BLLIP Treebank (Extracted)	3763	0.727

Table 1: Results from FERGUS on training on 1 million words of annotated corpus with different qualities of annotation

strings represented by each level of the derivation tree. The lattice at the root of the derivation tree is the result of the Unraveler.

Finally, the LP Chooser chooses the most likely traversal of this lattice, given a linear **language model** (n-gram). The lattice output from the Unraveler encodes all possible word sequences permitted by the supertagged dependency structure. We rank these word sequences in the order of their likelihood by composing the lattice with a finite-state machine representing a trigram language model which has been trained on an unannotated corpus, the **language model corpus**. We pick the best path through the lattice resulting from the composition using the Viterbi algorithm, and this top ranking word sequence is the output of the LP Chooser and the generator. We evaluate the results of our generator by using the string-edit distance from a reference string. For a detailed discussion on evaluation for FERGUS, including the limitations of the string-edit distance and proposals for other evaluation metrics, see (Bangalore et al., 2000).

4 Quality of Annotated Corpora

The tree model in FERGUS is a stochastic model that assigns supertags to the nodes of the input derivation tree. The parameters of this model are estimated from a training corpus of annotated derivation trees, where each node is annotated with a supertag from the grammar. The performance of the tree model depends on two aspects: the grammar which they refer to and the quality of annotation of the corpus. In this section, we discuss the impact of these aspects on the performance of FERGUS.

In order to investigate the impact of the quality of the grammar on the tree model, we used the supertags of the XTAG grammar (XTAG-Group, 1999) and the supertags of a grammar extracted automatically from the Penn Treebank as described above in Section 2. The grammar from XTAG contains 444 supertags while the extracted grammar contains 3063 supertags. Two tree models were trained on the two versions of the same one million word corpus annotated with the two supertag sets. The performance of FERGUS using these two tree models but with the same language model is shown in rows one and two of Table 1. It is interesting to note that the performance of FERGUS using the extracted grammar is as good as[2] the performance using the XTAG grammar. The supertag-annotated corpus based on the extracted grammar is more consistent than the supertag annotated corpus based on the XTAG grammar. This is because the Penn Treebank and the XTAG grammar were developed independently of each other, and as a result the annotations in the PTB and the structures in XTAG are not congruent. As a result, assigning supertags to the items in the PTB is based on heuristics, which are often but not always correct. This suggests that in future, grammars and treebanks should be developed so as to maintain this congruence.

In order to investigate the impact of the quality of the annotated corpus, we compared the tree model trained on the Penn Treebank with the grammar extracted from the PTB against a tree model trained on the BLLIP Treebank (BTB) (Charniak, 2000), with the grammar extracted from the BTB. The BLLIP Treebank has been automatically annotated with a statistical parser (Charniak, 2000). We automatically extracted grammars from one million words of each corpus. As mentioned earlier, the grammar from the PTB has 3063 supertags, while the BTB-extracted grammar has 3763 supertags. The performance of FERGUS using these two tree models but with the same language model is shown in rows two and three of Table 1. As expected, the performance of the tree model trained on the BTB is worse than that of the model trained on the PTB. However, the attraction of this model is that it was trained on automatically annotated corpus and hence it is straightforward to increase the quantity of the corpus. The impact of the increase in quantity of the corpus is illustrated in Section 6.

Our test suite is annotated for predicate-argument structure independently of the grammar being used for generation (we use the same test suite in all experiments reported in this paper). As the figures in Table 1 show, the performance of the automatically extracted grammars is comparable to that of the hand-crafted TAG. This means that the grammars automatically extracted from the PTB and

[2]Even slightly better, but that may not be significant.

	5K LM	10K LM	50K LM	100K LM	500K LM	1M LM
0K TM	0.434	0.480	0.561	0.599	0.661	0.701
5K TM	0.497	0.530	0.540	0.595	0.626	0.642
10K TM	0.507	0.574	0.542	0.651	0.672	0.690
50K TM	0.561	0.596	0.603	0.623	0.707	0.738
100K TM	0.563	0.595	0.610	0.644	0.697	0.729
500K TM	0.587	0.649	0.667	0.671	0.730	0.751
1M TM	0.600	0.627	0.645	0.658	0.738	0.750

Table 2: Results from FERGUS on training on Penn Tree Bank tree models (y axis) and linear language models (x axis) of different sizes

from the BTB not only account for the phrase structures found in these corpora, but also derive these phrase structures in a linguistically plausible manner (as we would expect from a hand-crafted TAG), so that the input derivation structures in the test suite can be used as derivation structures in those extracted grammars. This should be contrasted with, for instance, the approach chosen in Data-Oriented Parsing (Bod, 1998), where phrase structure is cut up into pieces which are not motivated by linguistic considerations. As a consequence, it is not clear how a grammar of the type extracted in DOP could be used for generation.

A point worth noting is that a TAG grammar such as XTAG contains not only information that maps predicate-argument to syntax, as do the automatically extracted grammars, but also information about grammatical roles (deep subject, deep object, modifier, and so on).[3] The impact of this information on the performance is not tested in the experiments presented here since the input is always assumed to be an unannotated dependency tree without role information.

5 Small Corpora

In this section, we investigate the effect of resource size when relatively small amounts of data (one million words or fewer) are available. This is a typical situation when a new generator is needed, either for a new domain or for a new language. Unfortunately, for the purpose of our investigation we need to resort to the English Penn Treebank (PTB), since we did not have access to any other tree bank of sufficiently large size (neither for a new domain in English, nor for a new language).

We assume that no hand-crafted grammar is available and that we will work with extracted grammars. We vary both the size of the syntactically annotated corpus from which the grammar is extracted and on which the tree model is trained (the ''TM corpus"), and the size of the unannotated corpus on which the linear language model is trained (the "LM corpus"). The results are shown in Table 2 and garphically in Figure 5. In this graph, each plot represents an LM corpus of a fixed size; the x-axis shows the natural logarithm of the size of the TM corpus, and the y-axis shows the string accuracy. Needless to say, the corpora for training the language model are easier to come by, and in practical terms it is inconceivable that the LM corpus would be smaller than the TM corpus, but we give all figures for the sake of completeness.

There are several conclusions to be drawn from these results.

- By and large, more is better: with the size of one model fixed, increasing the size of the other model also increases the quality of the output. There seems to be, however, a leveling-off effect at a TM corpus size of 500K, with little improvement when the TM corpus is doubled to 1M. Presumably, this is because of the increase in grammar size from the larger TM, the stochastic tree model becomes less precise.
- We see that the tree model contributes to the quality of the output starting with tree models of size 10K words; the 5K TM does not have a great impact on quality, except for the smaller language models. In fact, it appears that the tree model must be at least a tenth the size of the language model in order to have a positive impact.
- On the face of it, the impact of increasing the size of the LM is greater than the impact of increased TM: for example, if we have a 5K TM corpus and a 5K LM corpus with a performance of 0.497, we are better off increasing the LM corpus to 1M (0.642) than the TM corpus to 1M (0.600). Of course, this is an unrealistic trade-off, as the LM corpus will in practice always be at least as large as the TM corpus.
- More practically, if we have a LM corpus, it is worthwhile to increase the TM corpus to the size of the LM corpus, i.e., to annotate the entire LM corpus syntactically.

[3]This information might be extractable from annotated treebank as well.

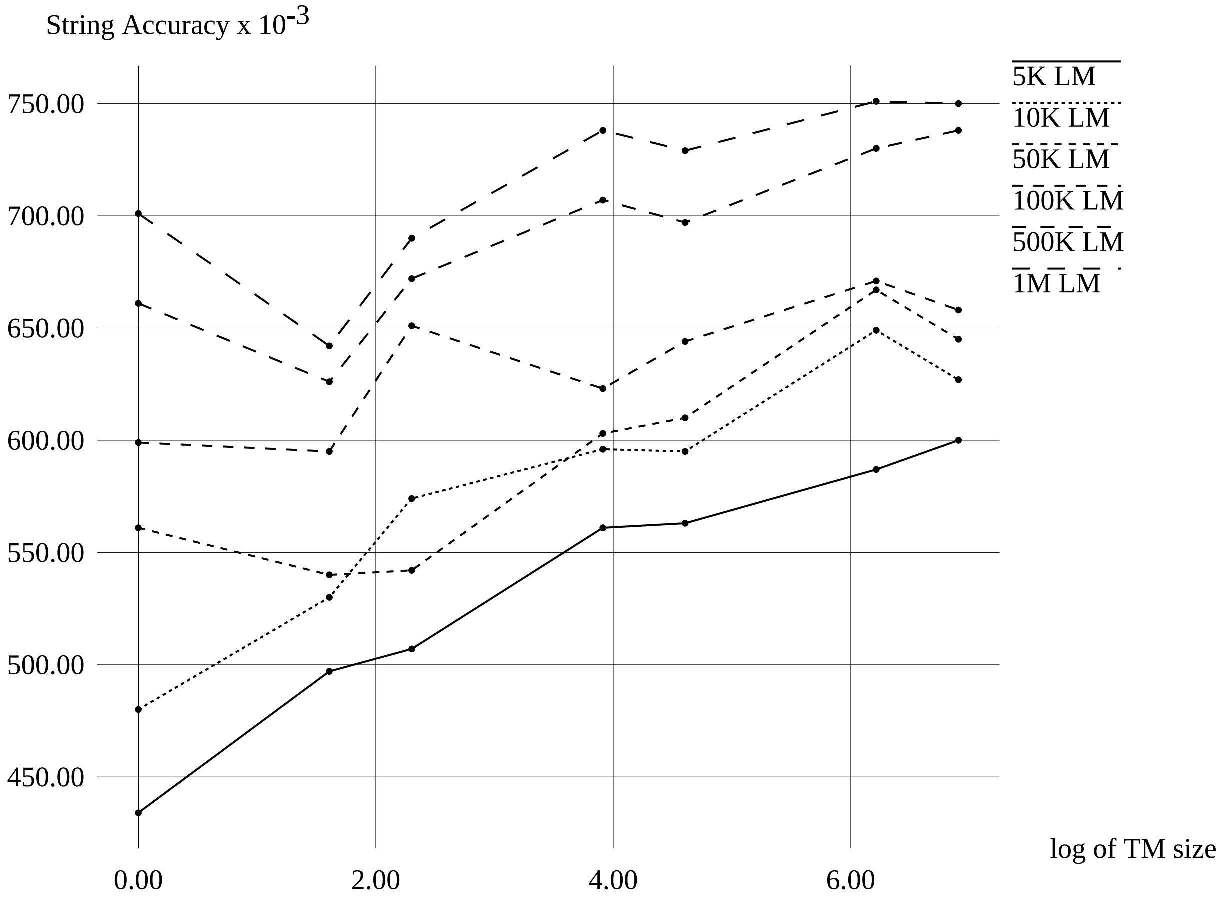

Figure 5: Results from FERGUS on training on Penn Tree Bank tree models (x-axis is log scale of size of TM corpus) and linear language models (plots) of different sizes

6 Large Corpora

In this section, we investigate the effect of resource size when a very large amount of data (more than one million words) is available. Specifically, we investigate whether we can overcome the disadvantage of using an automatically annotated TM corpus by annotating a large amount of data. To do so, we use larger and larger portions of the BLLIP Treebank (BTB). The results are shown in Table 3.

There are several conclusions to be drawn from these results.

- Increasing the size of the TM corpus increases the quality of the output, even if the TM corpus is low-quality, i.e., automatically annotated. In fact, our best results are achieved with grammars extracted from very large automatically annotated corpora. This result is significant, since if a very large unannotated corpus and a parser are available, there is no additional human work needed to syntactically annotate more data.
- We observe a leveling-off after 10M. Presumably, the additional data provided is rare and does not help the test suite.
- Clearly, increasing the size of the LM corpus also contributes to output quality. We intend to investigate larger LM corpora in future work;

	1M LM	2M LM
0K TM	0.685	0.703
1M TM	0.71	0.727
10M TM	0.764	0.786
20M TM	0.775	0.787
30M TM	0.767	0.784
40M TM	0.774	0.791

Table 3: Results from FERGUS on training on BTB tree models (y axis) and BTB linear language models (x axis) of different sizes

the emphasis of the work reported in this section has been the use of TM corpora of different sizes.

7 Conclusion

We have shown that automatically extracted grammars can be used in syntactic generation. We have found:

- An automatically extracted grammar can perform as well as a hand-crafted grammar, presumably because the corpus annotation is not perfect in case the hand-crafted grammar does not match the hand-crafted corpus. (This problem can be overcome by developing grammars and corpora in parallel.)
- Even a small amount of syntactically annotated data for grammar extraction improves performance over a system based solely on a linear language model, though the tree model corpus should be at least a tenth the size of the language model corpus.
- If no syntactically annotated corpus is available, but a high-performance parser is, a much larger corpus can compensate for the lower quality of the automatically annotated corpus.

This paper is about the relation between available corpora and generation quality when automatically extracting grammars. We observe, however, that generation may provide a general test-bed for evaluating the quality of grammars in a more application-independent manner. Generation is the mapping of a lexical (or semantic) predicate-argument structure to a surface string. The grammars that we use in FERGUS are Tree-Adjoining Grammars, i.e., declarative grammars which are independent of any application such as parsing or generation. We are thus assessing the ability of our grammars to map between surface string and some sort of meaning representation – which is exactly what grammar is generally assumed to do.

References

Anne Abeillé and Owen Rambow. 2000. Tree Adjoining Grammar: An overview. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*, pages 1–68. CSLI Publications.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–266.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *COLING2000*, Saarbrücken, Germany.

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation Metrics for Generation. In *Proceedings of International Conference on Natural Language Generation*, Mitzpe Ramon, Isreal.

Rens Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI Publications, Cambridge University Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the ANL/NAACL 2000 Workshop on Conversational Systems*, Seattle. ACL.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 65–76.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

Michael Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.

Kevin Knight and V. Hatzivassiloglou. 1995. Two-level many paths generation. In *Proceedings of ACL*, Boston. ACL.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *acl98*, pages 704–710, Montréal, Canada.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *ANLP00*, pages 170–177, Seattle, WA.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Günter Neumann. 1998. Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 120–123.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialog systems. In *Proceedings of the ANL/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, Seattle. ACL.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of First North American ACL*, Seattle, USA, May.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Fifth Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

The XTAG-Group. 1999. A lexicalized Tree Adjoining Grammar for English. Technical Report `http://www.cis.upenn.edu/~xtag/tech-report/tech-report.html`, The Institute for Research in Cognitive Science, University of Pennsylvania.