Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 287-294.

Markov parsing: lattice rescoring with a statistical parser

Brian Roark
AT&T Shannon Laboratory
180 Park Avenue
Building 103, Room E145
Florham Park, NJ 07932-0971
roark@research.att.com

Abstract

We present a generalization of an incremental statistical parsing algorithm that allows for the re-scoring of lattices of word hypotheses, for use by a speech recognizer. This approach contrasts with other lattice parsing algorithms, which either do not provide scores for strings in the lattice (i.e. they just produce parse trees) or use search techniques (e.g. A-star) to find the best paths through the lattice, without re-scoring every arc. We show that a very large efficiency gain can be had in processing 1000-best lists without reducing word accuracy when the lists are encoded in lattices instead of trees. Further, this allows for processing arbitrary lattices without n-best extraction. This can lead to more interesting methods of combination with other models, both acoustic and language, through, for example, adaptation or confusion matrices.

1 Introduction

Using a statistical parser to improve language models for automatic speech recognition is a topic that has been receiving much attention in recent years, with some documented success in test corpus perplexity reduction (Chelba, 2000; Roark, 2001a; Charniak, 2001) and word error rate reduction through n-best re-ranking or A-star search (Chelba, 2000; Roark, 2001b). The efficacy of these parsers as language models, as well as their primary shortcoming, results from their taking into account dependencies arbitrarily distant in the hypothesis string. For example, the string *'Either you mow the lawn by sundown, or . . .'* has a dependency between the first word (*Either*) and the eighth word (*or*), and this same dependency could have been arbitrarily far apart within the string. Parsing models are able to capture this kind of dependency in a way that more typical n-gram models cannot. To do so, however, requires keeping the entire prefix string around to evaluate the probability of each subsequent word, which carries a very large cost computationally. We will demonstrate, however, that the topology of the word lattice can be followed by an incremental statistical parser, by making a Markov assumption that does not preclude the parser from exploiting some of these long distance dependencies.

2 Background

Let us first consider the case of simple n-gram language models. What a language model provides is a joint probability of hypothesis strings, i.e. for a given string of k+1 words $w_0 \ldots w_k$, it assigns $\mathrm{P}(w_0 \ldots w_k)$. This is generally done via the chain rule (for convenience, let w_0^k stand for $w_0 \ldots w_k$):

$$\mathrm{P}(w_0^k) \;=\; \mathrm{P}(w_0)\prod_{i=1}^{k}\mathrm{P}(w_i|w_0^{i-1}) \qquad (1)$$

This model suffers from the same problem as the parsing models, namely that the entire prefix string must be retained to assign conditional probabilities to each subsequent word. For a strictly lexical language model, this leads to an estimation problem, due to sparse observations; it also causes a processing problem, since competing hypothesis paths, once split, can never re-join. Without lattice re-entrances, the hypothesis space grows exponentially. The solution that is universally adopted is to make a Markov assumption, i.e. assume that words more than some n words apart in the string are conditionally independent, given the intervening words. This eases the sparse data problems, and allows for paths that share the final n words to re-join. The Markov assumption is made by necessity, not because those who make it believe that there is such independence.

In terms of equivalence classes, a Markov language model of order n stipulates that two prefix strings of words belong to the same equivalence class if their final n words are the same. The effect of such a model is that the conditioning information in the chain rule is truncated to include only the previous n-1 words.

$$\begin{aligned} \mathrm{P}(w_0^k) &= \mathrm{P}(w_0)\mathrm{P}(w_1|w_0)\ldots \\ & \mathrm{P}(w_{n-1}|w_0^{n-2}) \prod_{i=n}^{k} \mathrm{P}(w_i|w_{i-n}^{i-1}) \end{aligned} \quad (2)$$

Syntactic parsing has traditionally dealt with given strings, and is performed for the purpose of generating a parse structure, or many possible parse structures, for that given string. Parsing algorithms have been generalized to accept lattices of word hypotheses – e.g. Kiefer et al. (2000; Chappelier et al. (1999) – but these are chart parsing algorithms intended to efficiently yield parse trees, rather than rescore the arcs in the lattice. Chelba (2000) applied his parser incrementally to lattices for the purpose of scoring hypotheses, using A-star search to find the best hypotheses in the lattice. This approach treated the lattice as a tree of hypotheses, without exploiting the re-entrances in the structure of the lattice. The efficient search meant that some arcs in the lattice would not be visited.

We propose a novel method for applying a parser to a lattice of word hypotheses that can take advantage of re-entrances in the lattice, and which allows the parser to visit and score every arc. It does this by making a Markov assumption akin to the kind of Markov assumption made by n-gram models. This is done, not because we feel that the independence assumptions are strictly speaking correct, but for the efficiency gains that they provide.

Before discussing what a Markov assumption looks like for an incremental parser, let us first briefly outline how an incremental parser can serve as a language model.

A PCFG[1] defines a probability distribution over strings of words as follows. Let T_w be the set of all complete trees rooted at the start symbol, with the string of terminals w_0^n as the terminal yield. Then

$$\mathrm{P}(w_0^n) = \sum_{t \in T_w} \mathrm{P}(t) \quad (3)$$

That is, the probability of the string occurring in the space of all possible strings of the language is the probability of the set T_w, i.e. the sum of its members' probabilities.

A PCFG also defines a marginal probability distribution over string prefixes, and we will present this in terms of partial derivations. A partial derivation (or parse) d is defined with respect to a string w_0^j as follows: it is the leftmost derivation[2] of the string, with w_j on the right-hand side of the last expansion in the derivation. Let $D_{w_0^j}$ be the set of all partial derivations for a prefix string w_0^j. Then the marginal probability P_M is the probability of all strings which begin with the prefix string w_0^j, and is defined as

$$\mathrm{P}_M(w_0^j) = \sum_{d \in D_{w_0^j}} \mathrm{P}(d) \quad (4)$$

This is the same as summing the probability of all complete trees which have w_0^j as the first $j+1$ words of their terminal yield.

From these marginals, we can calculate the conditional probabilities of the words given the prefix string. By definition,

$$\begin{aligned} \mathrm{P}(w_{j+1}|w_0^j) &= \frac{\mathrm{P}_M(w_0^{j+1})}{\mathrm{P}_M(w_0^j)} \\ &= \frac{\sum_{d \in D_{w_0^{j+1}}} \mathrm{P}(d)}{\sum_{d \in D_{w_0^j}} \mathrm{P}(d)} \end{aligned} \quad (5)$$

Note that the numerator at word w_j is the denominator at word w_{j+1}, so that the product of all of the word probabilities is the numerator at the final word, i.e. the marginal string prefix probability.

In short, the conditional probability is calculated by summing the probability of all of the trees before the word has been seen, incorporating the word into the parses to form a new set, summing the probabilities of the new set, and normalizing this by the

[1]We assume a familiarity with PCFGs. For an introduction to PCFGs, see, e.g., Manning and Schütze (1999).

[2]A leftmost derivation is a derivation in which the leftmost non-terminal is always expanded.

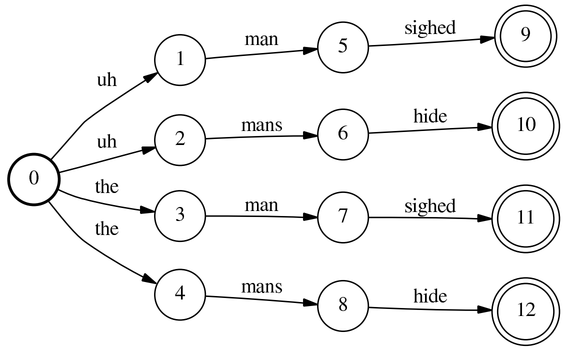

Figure 1: List of hypothesis strings encoded as a non-deterministic finite-state automaton

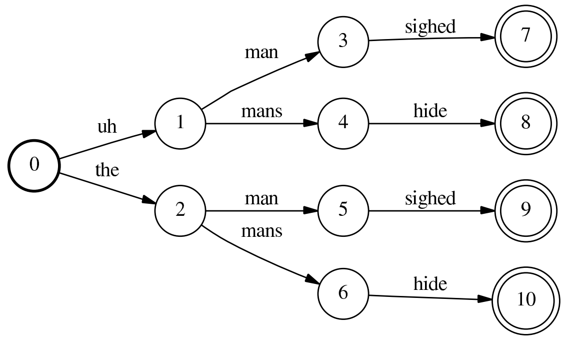

Figure 2: Tree of hypothesis strings encoded as a deterministic finite-state automaton

probability of the original set. The conditional probability can be thought of as the amount of probability mass the parser must expend to incorporate the next word. As shown in Roark (2001a), this can be approximated with a top-down, leftmost beam-search parser by simply performing the sums over the parses that have not been pruned. In practice, the pruned probability mass can be relatively low, yielding a snug lower bound on the joint probability of each string.

3 Markov parsing

At this point we can begin thinking about how to assign probabilities in the manner outlined above to the arcs in a finite-state representation of the hypotheses. Figure 1 shows a list of distinct hypothesis strings for a particular utterance, encoded as a non-deterministic[3] finite-state machine. In these figures, the state in bold is the initial state, the states with double circles are final states, and each arc is labeled with a terminal symbol. Each hypothesis string is represented as a separate path through the machine from the initial state to the final state.

To score the arcs, the parser accumulates at each state, from left-to-right through the machine, a weighted set of candidate parses for the hypothesis to that point. It then incorporates the word labeling the arc into the parses, and deposits the new set of candidate parses at the destination state. The arc probability is the sum of the probabilities of the derivations at the end state divided by the sum of the probabilities of the derivations at the beginning state. Of course, the probability of the derivation at the initial state is one.

For example, state 3 in figure 1 will have a set of partial derivations, all of which have the word ‘the’ as their terminal yield. There are a variety of ways in which the word ‘man’, which labels the transition from state 3 to state 7, can be incorporated into these parses. The arc probability, which is the conditional probability of the word ‘man’ given the previous words, is estimated as the sum of the probabilities of all parses at state 7 divided by the sum of the probabilities of all parses at state 3.

Figure 2 shows the same set of hypotheses encoded as a deterministic finite-state machine. This is a tree of hypothesis strings, and it encodes exactly the same set of hypotheses as the original machine. Once again, for the parser, the states are a set of weighted derivations, and the probability assigned to each arc is the normalized probability of those derivations that traverse the arc.

Figure 3 shows the same set of hypotheses as the previous machines, in the minimal lattice representation. Again, there are four paths from the initial state to the final state, corresponding to the four hypotheses. Unlike the trees, the lattice has reentrances in the graph, at points where distinct hypotheses share the remainder of the string. Finally, figure 4 shows the machine that results from taking the minimal lattice and splitting the states so that every path into a state has the same last word. Thus, both paths that go through state 2 have “man” as the last word of the path to that point. If we interpret these states as independence assumptions, this lat-

[3]The machine is non-deterministic because there are multiple arcs out of the initial state with the same label.

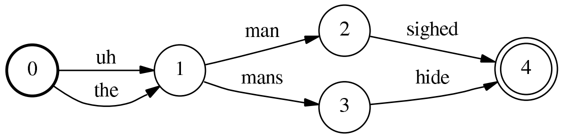

Figure 3: Hypothesis strings encoded in the minimal deterministic finite-state automaton, which represents a Markov assumption of order 0

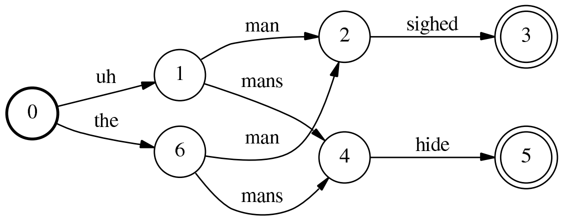

Figure 4: Hypothesis strings encoded in a deterministic finite-state automaton, which has had the states split to represent a Markov assumption of order 1

tice is encoding a Markov assumption of order one.

Which brings us to the question of what a Markov assumption is in the context of incremental parsing. Perhaps this is easiest to see in terms of equivalence classes. As mentioned above, a Markov assumption of order n treats all strings with the same n final words as equivalent. The set of parses associated with the prefix string is the set of all parses in the equivalence class, i.e. all parses for all strings that have the same n final words. Since we are primarily interested in the probabilities of these sets, we can limit the sets to all parses with probability greater than 0. Of course, for a given lattice of word hypotheses, we do have some more restrictions on this set; namely, that the only non-zero probability derivations are for strings which are represented as paths within the lattice. In other words, we do not need to consider all possible strings in the equivalence class, only those represented within the lattice of hypotheses; all others have probability zero.

More formally, for a given Markov order k, let $D_{w_{j-k}^{j-1}}$ be the set of all partial derivations for prefix strings αw_{j-k}^{j-1}, for any α prefix to w_{j-k}^{j-1} in the lattice. Then the marginal probability P_M is defined, analogously to equation 4, as

$$\mathrm{P}_M(w_{j-k}^{j-1}) = \sum_{d \in D_{w_{j-k}^{j-1}}} \mathrm{P}(d) \qquad (6)$$

As with equation 5, by definition:

$$\begin{aligned} \mathrm{P}(w_j | w_{j-k}^{j-1}) &= \frac{\mathrm{P}_M(w_{j-k}^{j})}{\mathrm{P}_M(w_{j-k}^{j-1})} \\ &= \frac{\sum_{d \in D_{w_{j-k}^{j}}} \mathrm{P}(d)}{\sum_{d \in D_{w_{j-k}^{j-1}}} \mathrm{P}(d)} \qquad (7) \end{aligned}$$

This conditional probability can also be approximated with a beam-search parser.

Here is a simple description of how the arcs are scored: the probability of being in a state is the sum of the probabilities of all trees at that state; and the conditional word probability is calculated by traversing the arc (extending the parses) and normalizing the sum of the resulting tree probabilities with the original state probability. Because of the beam-search pruning, the amount of work required to traverse a given arc under a Markov assumption will be much less than traversing the set of outgoing arcs when the states are split.

A brief note about the pruning. Following Roark (2001a), our beam threshold is set according to the highest probability parse that has successfully integrated the current word. Candidate parses with a high figure-of-merit are worked on first. When there is an equivalence class, the set of competing parses will contain parses with different terminal yields. The figures-of-merit for parses can be determined strictly by the parsing model and some look-ahead, or perhaps the parsing and look-ahead models in combination with the acoustic scores, which would boost parses with higher likelihood given the signal. In any case, parses with a low figure-of-merit will be pruned by the parser. For the current study, we only used the parsing and look-ahead models, and allowed parses within an equivalence class to compete in exactly the same way parses with the same terminal yield compete, i.e. relative to the best scoring candidate parse from the set.

Note that this formulation allows the parser to rescore an arbitrary lattice, without placing restrictions on the manner in which the conditional probabilities are calculated by the parser for any given parse. Since each parse has a specific terminal yield, long-distance dependencies may be captured by the conditional parsing model, even if they extend beyond the limit of the Markov assumption. One cannot guarantee, however, that the lowest cost path is contained within the parse trees at the final state.

In practical terms, the parser takes an arbitrary lattice for processing. At each state in the lattice, it accumulates parses. Provided that the parser evaluates the lattice in a topological ordering, it is ensured that all incoming arcs are traversed for any given state before any of its outgoing arcs are traversed. This means that the entire set of parses is available at each state when it is reached. Every outgoing arc is traversed by the parser, adding to the set of parses at the destination state, and allowing the arc to be scored appropriately. The Markov assumption comes into play in the structure of the lattice given to the parser.

The basic idea of following the topology of a given lattice, as it is presented here, can be used with any incremental derivation strategy, such as the Structured Language Model (Chelba and Jelinek, 2000). In that model, they normalize their score over the beam of derivations, rather than summing it, but the same idea holds. Markov parsing is related to the ability to calculate marginal string probabilities from sets of partial derivations. A language model such as that of Charniak (2001) can only calculate string probabilities from the set of complete derivations, since each derivation can generate the terminals in a different order. Hence this model cannot take advantage of our approach.

Since the parser operates on arbitrary lattices, we can give the parser a tree of word hypotheses (i.e. with no Markov assumption in the structure of the automaton) or a minimized lattice that has had its states split according to some desired Markov order. Empirical results will be presented for n-best search both with and without a Markov assumption, as well as for word lattices without n-best extraction.

4 Empirical evaluation

The first question that must be asked is what an effective Markov assumption is. To address this question, we can evaluate the performance of the parser on n-best hypotheses encoded as either a tree (i.e. no re-entrances) or as a lattice with states that encode a particular Markov order.

The parser that we used for these trials follows the basic algorithm from Roark (2001a), as well as modifications and tree transformations presented in Roark (2001b). The beam-search parameters that are used are also the same as those used in the above papers. Due to space constraints, we refer the reader to those publications for parsing algorithm details. Briefly, we use an incremental beam-search parser that follows a top-down, leftmost derivation strategy, and conditions rule probabilities on events from the left-context, including non-terminal labels and lexical heads. There is one word of look-ahead, and parsing at each word continues until the beam threshold is reached, at which point the parser moves to the next word. Our parser differs from those presented in the cited papers in that it accepts lattices and trees of hypotheses as input (encoded as finite-state automata), and outputs weighted automata and (optionally) parse trees.

We evaluated recognition performance for a 160,000-word vocabulary North American Business News (NAB) task, specifically on the 300 utterance DARPA NAB Eval '95 test set. We took the lattice output of the multi-pass recognizer (with acoustic model adaptation) that is presented in Mohri et al. (to appear). The language model for the lattices was a Katz backoff trigram (Katz, 1987) trained on 250 million words. These lattices had an oracle word error rate (WER) of 2.9 percent[4], and a best-path WER of 10.6 percent. We pruned the lattices using the AT&T FSM tools (Mohri et al., 2000) using a log cost threshold of 7. This resulted in lattices with an oracle WER of 3.7 percent. We then extracted the best paths from each lattice up to 1000 unique hypotheses, which resulted in an average of 549 per utterance, with 143 out of 300 lattices containing 1000 unique hypotheses. The language model score was stripped off, leaving the acoustic model score ready to be combined with a new language model score. The n-best lists were encoded as deterministic trees, as well as minimized lattices split to various Markov orders.

We trained the parsing model on sections 2-21 of the WSJ Penn Treebank, plus the approximately 40

[4]This is obtained by using an oracle to find the path that gives the lowest error rate.

Mixing	Markov Assumption									
Coeff. λ	None	8	7	6	5	4	3	2	1	0
1.0	89.4									
0.01	89.8	89.7	89.7	89.8	89.9	89.8	89.9	89.7	89.8	88.6
0.50	90.1	90.0	90.1	90.2	90.1	90.1	90.2	90.0	90.1	89.1

Table 1: Word accuracy for n-best re-ranking on the NAB Eval '95 test set. The mixing coefficient λ represents how the trigram and parsing models are mixed together: $\lambda P_{trigram}$ + (1-λ)P_{parser}. The parsing model is always mixed at some epsilon with the trigram, to ensure every arc receives a non-zero score.

million words in the BLLIP 1987-89 WSJ Corpus Release[5], plus our parser's best guess parses from another 60 million words of WSJ from 1994-1996. This total of 100 million words of treebank data was normalized from text to speech formats (e.g. numbers converted to how they would be read, etc.) and used to train the stochastic parsing model that was used to re-score the lattices produced as outlined above. Issues with conversion between treebank and lattice tokenization were dealt with in the standard way, as discussed in Chelba (2000) and Roark (2001b).

The parser produces automata weighted with arc probabilities. These probabilities were interpolated with the trigram model[6] that came with the lattices according to the parameter λ, so that the probability of each arc was $\lambda P_{trigram}$ + (1-λ)P_{parser}. After the language model mixing, the scores were converted to negative log probability. The resulting weighted automaton was intersected with the acoustically scored automaton, and the best path extracted and evaluated.

Table 1 presents the Word Accuracy (100-WER) for the parsing trials at λ = 1.0, 0.01, and 0.5, which represent trigram accuracy, parser model accuracy, and mixed accuracy. Surprisingly, we can see that even with a Markov assumption of order 1, we get nearly the same performance as with no Markov assumption. Figure 5 plots the accuracy at various Markov orders alongside the non-Markov performance, and figure 6 plots the accuracy versus the number of arcs that must be traversed to score the

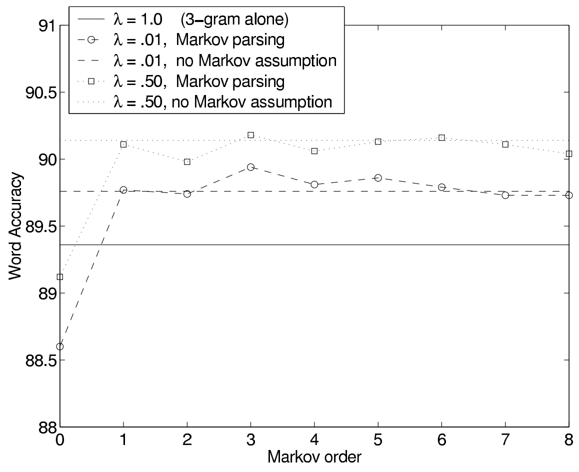

Figure 5: Markov order versus word accuracy, for n-best reranking with and without Markov assumptions, n=1000

automaton, when it is encoded non-deterministically (i.e. every path separately, as in figure 1), as a deterministic tree of hypotheses (as in figure 2) and as lattices of various Markov orders. This is an efficiency measure, to the extent that the arcs represent an equivalent amount of work. That is not quite true, since the number of derivations at each state can cause more or less work to traverse the arc, given the particular beam-search heuristic, so figure 7 shows accuracy versus time per utterance for the deterministic tree and the lattices[7]. We can see that it is more-or-less the same graph as figure 6. We note that a minute per utterance may seem like a lot of time, but recall that we are speaking of processing up to a thousand hypothesis strings of business news per utterance.

[5]This corpus consists of the best guess parses from Eugene Charniak's very high accuracy statistical parser (LDC Catalog #LDC2000T43).

[6]Since the parser can garden path (and hence provide a zero score to arcs), we always interpolate the parsing model with an n-gram at some $\lambda \geq \epsilon$.

[7]We didn't actually parse the individual hypothesis strings, since this is equivalent to the deterministic tree.

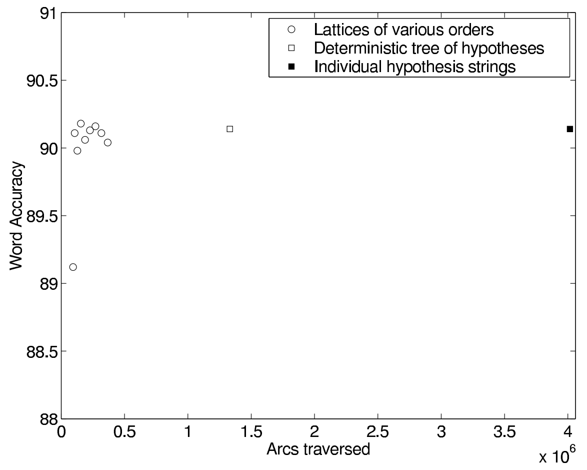

Figure 6: Arcs traversed versus word accuracy, for n-best reranking with and without Markov assumptions, n=1000

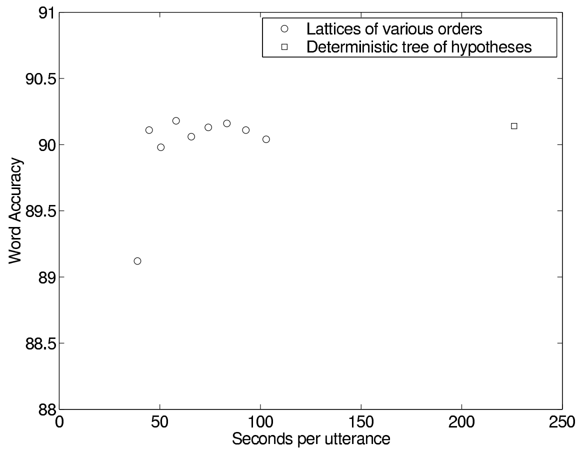

Figure 7: Time in seconds per utterance versus word accuracy, for n-best reranking with and without Markov assumptions, n=1000

To summarize, we have shown that a small Markov assumption results in the same accuracy gains as parsing with no Markov assumption, with a large efficiency gain over processing them as a tree with no Markov assumption, and an even larger gain than processing each string independently.

However, the principal gain in making a Markov assumption when processing lattices is that one does not need to perform n-best extraction in the first place, but can re-score an arbitrary given lattice. The number of distinct hypotheses that can be packed into a relatively small lattice is very large. For example, the lattices for this particular test set before n-best extraction contain, in aggregate, about 23.4 billion unique hypothesis strings, instead of the 300,000 maximum that we place by doing 1000-best extraction. Evaluating all of these as individual strings is not feasible, and even encoded as a tree this is too costly; but we can rescore the lattices by making a Markov assumption, as outlined above.

Table 2 gives the word accuracy for the parsing model at $\lambda = 0.01$ and 0.50 on the lattices, without n-best extraction, under different Markov assumptions[8]. Recall that we are visiting and scoring every arc in the lattices. The first thing to note is that the accuracy goes down relative to the n-best re-scoring accuracy when the parsing model is providing most of the language model probability mass ($\lambda = 0.01$). This is un-surprising, because the n-best extraction is done relative to the trigram scores, so that the parsing model gets some extra information from this other model, via pruning, about what are good hypotheses and what are not-so-good. Notice that when mixed with the trigram model at $\lambda = 0.5$, the scores are as good as with the n-best re-scoring. That is, we are gaining back the points that were lost by explicitly combining with the other model. It is also worthwhile pointing out that the rather surprisingly good performance at the smaller Markov orders in the n-best trials also seems to be the result of the n-best extraction, since we see more degradation of performance as the Markov order drops to that level without it.

One may wonder why we would want to do lattice re-scoring without n-best extraction, if there is no ultimate WER improvement as a result. There are a couple of answers to this. First, the utility of a weighted lattice is not simply in extracting the best path from that lattice. Modern multi-pass recognition techniques involve acoustic and language model adaptation to individual speakers, and these scored lattices can serve as input into this adaptation process. The better the posterior probabilities within the lattice, the better the adaptation to the speaker. This

[8]Three of the lattices in the test set grew to over 1GB in size when split to a Markov assumption of 6. For the results presented, we used a Markov assumption of 5 for just those three utterances in the 6 condition.

Mixing	Markov Assumption					
Coeff. λ	6	5	4	3	2	1
0.01	89.6	89.3	89.4	89.3	89.2	89.1
0.50	90.2	90.2	90.0	90.1	89.7	89.6

Table 2: Word accuracy for lattice re-scoring on the NAB Eval '95 test set. The mixing coefficient λ represents how the trigram and parsing models are mixed together: λP$_{trigram}$ + (1-λ)P$_{parser}$.

allows the syntactic model to take part in the recognition process earlier, which is likely to help in ways that simple re-scoring or re-ranking cannot. Second, these trials were performed on already adapted models in a domain with relatively good accuracy, so that the n-best extraction with a large n is effective. In other domains (e.g. Switchboard), and earlier in the recognition process, such extraction is less effective.

5 Conclusion

We have presented a generalization of an incremental statistical parsing algorithm which allows, via a Markov assumption, the re-scoring of lattices with the parsing model. We have shown that a surprisingly small Markov order can yield the same word accuracy as no Markov assumption when re-scoring a lattice with a statistical parser, with large efficiency gains. With such an algorithm, we can rescore lattices which encode a very large number of unique hypotheses. This approach allows the incorporation of statistical parsers into a recognizer at a much earlier stage of the process.

This paper is not intended to present "compelling" evidence for an incremental parsing model over non-incremental. In fact one could argue that, to the extent that all of these models – n-grams as well as various statistical parsers – capture different kinds of dependencies, they should be jointly used to maximize performance, though perhaps at different stages in the process.

Future work will include evaluating performance when scores from the acoustic and/or n-gram models are incorporated for ranking competing candidate parses. Here we imposed specific Markov orders for all states in all lattices; we would like to investigate more sophisticated methods for deciding when states should be split. Also, comparisons between this approach and A-star search for best path extraction would be of interest. Finally, we will evaluate this approach with respect to how it improves the acoustic model adaptation process, by including the parsing scores earlier.

Acknowledgments

Thanks to Mehryar Mohri and Michael Riley for useful discussion on this topic.

References

J. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of the Sixth confrence sur le Traitement Automatique du Langage Naturel (TALN'99)*, pages 95–104.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, The Johns Hopkins University.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 35(3):400–401.

Bernd Kiefer, Hans-Ulrich Krieger, and Mark-Jan Nederhof. 2000. Efficient and robust parsing of word graphs. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 280–295. Springer, Berlin.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. *to appear*. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*.

Brian Roark. 2001a. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Brian Roark. 2001b. *Robust Probabilistic Predictive Syntactic Processing*. Ph.D. thesis, Brown University. http://arXiv.org/abs/cs/0105019.