Journal of Universal Computer Science, vol. 23, no. 11 (2017), 1019-1034
submitted: 23/4/17, accepted: 26/11/17, appeared: 28/11/17 © J.UCS

An Evaluation of Structured Language Modeling for Automatic Speech Recognition

Johanna Björklund
(Dept. Computing Science, Umeå University, 901 87 Umeå, Sweden
`johanna@cs.umu.se`)

Loek Cleophas
(Dept. Information Science, Stellenbosch University
7602 Matieland, Republic of South Africa
`loek@fastar.org`)

My Karlsson
(Codemill AB, 903 36 Umeå, Sweden
`my.karlsson@codemill.se`)

Abstract: We evaluated probabilistic lexicalized tree-insertion grammars (PLTIGs) on a classification task relevant for automatic speech recognition. The baseline is a family of n-gram models tuned with Witten-Bell smoothing. The language models are trained on unannotated corpora, consisting of 10,000 to 50,000 sentences collected from the English section of Wikipedia. For the evaluation, an additional 150 random sentences were selected from the same source, and for each of these, approximately 3,200 variations were generated. Each variant sentence was obtained by replacing an arbitrary word by a similar word, chosen to be at most 2 character edits from the original. The evaluation task consisted of identifying the original sentence among the automatically constructed (and typically inferior) alternatives. In the experiments, the n-gram models outperformed the PLTIG model on the smaller data set, but as the size of data grew, the PLTIG model gave comparable results. While PLTIGs are more demanding to train, they have the advantage that they assign a parse structure to their input sentences. This is valuable for continued algorithmic processing, for example, for summarization or sentiment analysis.

Key Words: language modeling, automatic speech recognition, probabilistic lexicalized tree-insertion grammars
Category: F.4.2, I.2.7

1 Introduction

Language models form a central concept in natural language processing. At an abstract level, they are parametrized families of functions that assign probabilities to natural language sentences. A 'good' language model should assign a higher probability to natural-sounding sentences and sentence fragments, compared to unlikely or even ungrammatical alternatives. Language models are commonly used in machine translation to rank candidate translations, in automatic speech recognition (ASR) to narrow the search-space by discarding unlikely transcriptions, and in summarization to shorten text while preserving readability.

The reigning approach to language modeling is to train structure-agnostic n-grams on large sets of data. In the n-gram model, the likelihood of a sequence of words $s = w_1, w_2, \ldots, w_n$ is the product of the likelihood of every subsequence of $\leq n$ consecutive words at the start of s, with respect to some given set of training data. When greater precision is needed, the size of n can be increased, or some weighting or smoothing technique can be introduced to better leverage the training data. These language models have the advantage of being (i) surprisingly powerful considering their simplicity, (ii) easy to train from data, and (iii) applicable in linear or even log-linear time [Klakow 1998].

On the downside, the size of n-gram models grows exponentially in n. This means that for large n, only a fraction of all possible n-grams will appear in the training data, and to keep the model at a reasonable size, only the most frequent n-grams can be taken into account at any rate. For these reasons, Goodman [2001] concluded that proceeding beyond 5-grams is unlikely to be practically motivated. A decade later, this still seemed the case. The flagship of the field, Google's n-gram viewer, contained 500 billion words compiled from 20 million books. In its construction, the value of n was restricted to 5 to limit the model's size, and all n-grams encountered fewer than 40 times were discarded without updating the model [The Google Books Team 2010]. Recently, however, Chelba et al. [2013] proposed a hybrid approach based on neural networks that uses a history of up to 9 previous words and additional features of 15 previous words (taken as an unordered set). Whether this contradicts Goodman's conjecture can be argued, but it is clear that with a limited dictionary of 1000 words, we need a table with one octillion entries, i.e., with 10^{27} bytes, to do full 9-grams without back-offs. Even if only a percentage of a percentage of the entries contain non-zero probabilities, the storage alone requires a cluster of billions of computers with petabytes of memory each. We conclude that even though the approach put forth in [Chelba et al. 2013] may handle a large number of 9-grams explicitly, most of the computations are done on shorter n-grams, and that this will continue to be the case for a long time to come.

The need to restrict n causes problems. Consider the following two sentence variations:

> She *rowed* all the way from the excavation site at Yellowknife, to the base camp Benchoko, in her weathered *canoe*.
>
> She *rowed* all the way from the excavation site at Yellowknife, to the base camp Benchoko, in her weathered *car*.

The first sentence is arguably more likely due to the semantic relation between *rowed* and *canoe*. However, no n-gram model could make this connection for a value of n less than 18, since no n-gram of shorter length contains the history of words from *rowed* up until *canoe*, or *rowed* and *car*. Such simpler models would instead prefer the second sentence, *car* being a more common form of transport than *canoe*.

Another thing that is missing is a syntactical analysis of the input sentence, that is, n-grams do not support parsing. Parsing is useful in itself, as it tells us whether a sentence is likely to be perceived as grammatical. Furthermore, parse trees are suitable for continued algorithmic processing, for example, to obtain a semantic analysis. If we want to understand who does what to whom in a sentence, it helps to know how the sentence is put together, in particular, what the central verbs and arguments are.

Context-free grammars (CFG) can be used for syntactical analysis, but are typically worse at predicting word sequences than n-grams, unless they are lexicalized. In a lexicalized CFG, each production rule produces at least one lexical item [Schabes and Waters 1993]. CFGs can be lexicalized using Greibach normal form, but this skews the syntactical structure. Another alternative is to use Tree-adjoining Grammars (TAG) [Joshi et al. 1975]. TAGs represent a language of parse trees as a set of parse-trees fragments, together with rewrite rules that regulate how the fragments may be pieced together into larger structures. TAGs were developed by linguists and have several appealing properties; they are expressive, straight-forward to lexicalize, and offer parsing [Abeillé et al.1990)Abeillé Schabes and Joshi; Shieber and Schabes 1990]. However, the parsing complexity is as high as $O\big(n^6\big)$, which cannot be considered practical.

Aiming for the middle ground, Hwa [2001] suggested the use of probabilistic lexicalized tree insertion grammars (PLTIGs), a family of TAGs with simplified rewrite rules. PLTIGs are as good as trigrams at predicting word sequences, but also offer syntax-aware language modeling. Their parsing complexity is $O\big(n^3\big)$, equal to that of probabilistic context-free grammars (PCFGs) and a factor n^3 faster than TAGs. PLTIGs also have the advantage that their expectation-maximization training converges faster than for similarly sized PCFGs [Hwa 2001, Ch. 3].

In this article we evaluate PLTIGs as an alternative for n-grams in automatic speech recognition, more precisely, the translation of spoken words into text. To allow comparison with previous studies, we focus on the English language. Modern ASR systems are typically built around an acoustic model, a lexical model, and a language model. The acoustic one maps utterances into phonemes (oftentimes by means of sequential transducers [Mohri 1997]); the lexical one concatenates these to match a dictionary of known words; and the language model combines words into sentences. Today, n-grams and Hidden Markov Models are commonly used language models in ASR systems, and we are interested to learn how PLTIGs stand up to these. To isolate the influence of the language models from the ASR system at large, we focus on a classification task in which the models pick out a sentence s from a set of alternatives, generated from s by replacing a word by a similar but inappropriate word.

The rest of this article is organized as follows. Section 2 revises the relevant language models. Section 3 describes the experimental setup, and Section 4 reports and discusses the results. Section 5 concludes the paper and outlines future work.

2 Theoretical Background

We begin by recalling the definitions of n-grams, TAGs (as an intermediary step), and finally PLTIGs. Since the theoretical results that support this study are already in place, including the parsing complexity and the correctness of the Expectation-Maximization (EM) algorithm, we describe the models at a fairly high level and refer to [Hwa 1998] for the formal definitions. Those familiar with n-grams and PLTIGs may want to proceed directly to Section 3.

2.1 n-grams

The likelihood $P(w_{1,n})$ of a sequence of words $w_{1,n} = w_1, \ldots, w_n$ can be computed using the chain rule of probability

$$P(w_{1,n}) = \prod_{i=1}^{n} P(w_i | w_{1,i-1}) \ . \tag{1}$$

With n-grams, the likelihood of w_i in $w_{1,n}$ is approximated by

$$P(w_i | w_{1,i-1}) \approx P(w_i | w_{i-n+1,i-1}). \tag{2}$$

Substituting this into Eq. 1, the likelihood of the sequence is approximated by

$$P(w_{1,n}) \approx \prod_{i=1}^{n} P(w_i | w_{i-n+1,i-1}) \ . \tag{3}$$

For n equal to 1, 2, 3, 4, and 5, the model is often referred to as unigram, bigram, trigram, quadgram, and quintgram, respectively. It is straight-forward to learn n-grams from training data through a Maximum Likelihood Estimation (MLE) process. The probability of a word w_i is

$$P_{MLE}(w_i | w_{i-n+1,i-1}) = \frac{c(w_{i-n+1,i})}{c(w_{i-n+1,i-1})} \ , \tag{4}$$

where $c(w_{1,k})$ counts how often the sequence of words $w_{1,k}$ appears in the data. However, for larger values of n, the training data cannot be expected to contain

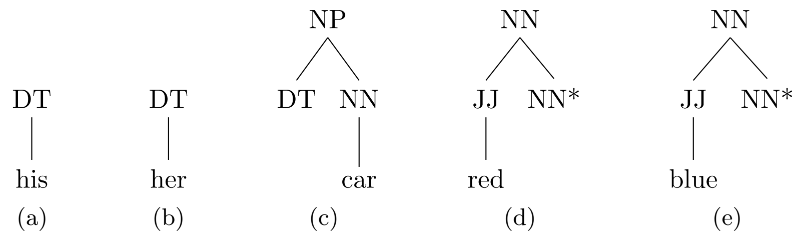


Figure 1: Examples of elementary trees in a TAG. Elementary trees (a-c) show initial trees, while (d-e) are auxiliary trees. The auxiliary trees each have a foot node marked with a $*$ of the same type as the root node.

all n-grams that will later be encountered in the application data. For this reason, the MLE learning is usually followed by a round of *smoothing*; a redistribution of the probability mass assigning a non-zero probability also to here-to unseen n-grams. There are several popular smoothing methods to choose between, for example, Laplace, Good-Turing, and Kneser-Ney. In this article, we use Witten-Bell smoothing. At the core of Witten-Bell is the observation that when we see some words, we can easily guess what the next word will be, whereas for others, it is more difficult. The word *want* for instance is followed by *to* more often than not, but the word *to*, in turn, tells us less about the future. With Witten-Bell, the nth order smoothed model is a linear interpolation between the nth order unsmoothed model and the $(n-1)$th order smoothed model, and the shape of the interpolation is affected by the *predictiveness* of the words [Bell et al. 1990].

2.2 Tree Adjoining Grammars

As previously mentioned, tree-insertion grammars are a restricted form of tree adjoining grammars [Joshi et al. 1975]. In contrast to n-grams, probabilistic TAGs assign likelihoods to parse trees rather than their surface forms, that is, the generated sentences. A TAG has rules in form of *elementary trees* of which there are two types; *elementary initial trees* and *elementary auxiliary trees*. Each elementary initial tree has a number of nonterminal and terminal leaf nodes. Auxiliary trees also have a special nonterminal leaf node of the same type as the root node called the *foot node*, marked with the special symbol $*$. The sequence of nodes on the path from the root node to the foot node is called the *spine*. Figure 1 shows a few examples of elementary trees.

If the root node of an initial tree matches a nonterminal leaf node of some other tree, then the first tree can be *substituted* into the nonterminal leaf node of the other tree, corresponding to a context-free derivation step. Figure 2 shows an example of this. The determiner *her* in the initial tree in Figure 2a is substituted

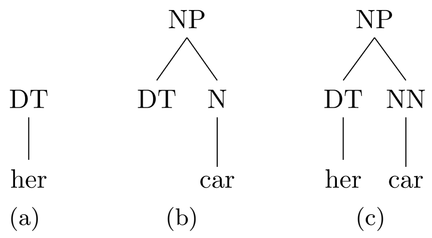


Figure 2: The initial tree in (a) is substituted into a nonterminal node of the initial tree in (b), with the result of the new tree in (c). This is possible since the root node of (a) is of the same type as the nonterminal leaf node in (b).

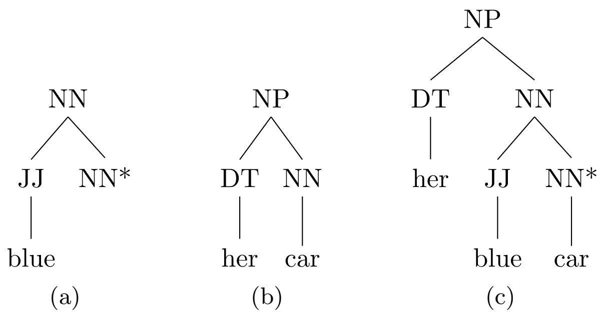


Figure 3: The auxiliary tree in (a) is adjoined into the tree in (b), with the result of the new tree in (c). The tree from (a) can be inserted into (b) at the node `NN` since (a) has a root and a foot node of that type.

into the leaf node of the initial tree for the noun *car* in Figure 2b. The result is the tree in Figure 2c, generating the construct *her car*.

An auxiliary tree can be inserted into an intermediate node of another tree through what is called *adjunction*. Since the root node and the foot node of an auxiliary tree have to have the same symbol, auxiliary trees can be inserted into another tree where such a symbol exists. An example of this is shown in Figure 3. Note that the auxiliary tree in Figure 3a has the same root and foot node as one of the intermediary nodes in the tree in Figure 3b. This allows the adjective *blue* to be inserted into the sentence *her car*, producing the new sentence *her blue car*. TAGs are expressive due to the number of configurations they allow, but have the drawback of high parsing complexity in the order of $O(n^6)$.

More formally, a TAG is a tuple (Σ, V, I, A), where Σ is a finite set of terminal symbols, V is a finite set of nonterminal symbols, I is a finite set of initial trees, and A is a finite set of auxiliary trees. If every elementary tree structure in a

TAG has at least one non-empty terminal leaf node with a lexical item, then that is called a *lexicalized TAG* (LTAG). There is also a probabilistic version of TAGs (and hence of LTAGs), in which three probability distributions are included in the definition. These distributions are defined on the set of initial trees, and on the finite set Ω of possible substitution and adjunction events. A Probabilistic TAG is thus a tuple $(\Sigma, V, I, A, P_I, P_S, P_A)$ where (Σ, V, I, A) is a TAG, P_I is a mapping $I \to [0,1]$ or the probability that an initial tree is used as the start of a derivation, P_S is a mapping $\Omega \to [0,1]$ for the probability of a substitution, and P_A is a mapping $\Omega \to [0,1]$ for the probability of an adjunction [Resnik 1992].

2.3 Tree Insertion Grammars

Tree insertion grammars (TIGs) are a restricted form of TAG in which the rewrite steps are more regulated. A TIG is a quintuple (Σ, V, I, A, S) where (Σ, V, I, A) is a TAG and S is a distinguished nonterminal symbol. The semantics of TIGs are defined as for TAGs, with the following additions [Schabes and Waters 1994].

– Every auxiliary tree has to be either a *left auxiliary tree* or a *right auxiliary tree*, i.e., they only have lexical items on the indicated side of the foot node. Auxiliary trees with lexical items on both sides of the foot node and empty auxiliary trees are not allowed.

– Left auxiliary trees are not allowed to be adjoined on a node that is *on* the path from the root to the foot of a right auxiliary tree, and vice versa.

– No more than one left auxiliary tree and one right auxiliary tree is allowed to be simultaneously adjoined into the same node. Note that two different parse trees can be generated when two auxiliary trees are adjoined at the same node. One where the left auxiliary tree has been applied first, and one where the right auxiliary tree has been applied first.

– Adjunction on nodes that are *to the right of* the path from the root node to the foot node of a *left* auxiliary tree is not allowed, and vice versa for right auxiliary trees.

– Adjunction on root nodes and foot nodes of auxiliary trees is not allowed.

– Adjunction on nodes that can be used for substitution is not allowed.

When combined, these restrictions only allow us to express context-free languages, but they also bring the parsing complexity down to $O(n^3)$.

A TIG is lexicalized (LTIG) if each elementary tree carries a lexical item. A TIG can also be probabilistic (PTIG or PLTIG) if it is parameterized by probabilities that describe how likely it is that operations are applied. A PTIG is defined as a TIG, with the addition of as many as eight probability distributions, $P_I, P_S, P_L, P_R, P_{NL}, P_{NR}, P_{RL}, P_{LR}$, defined as follows:

1. The probability $P_I(t)$ that the derivation starts with the initial tree t, so that $\Sigma_{t \in I} P_I(t) = 1$.

2. The probability $P_S(n, t)$ that an initial tree t is substituted into a leaf node n, so that $\Sigma_{t \in I} P_S(n, t) = 1$.

3. The probabilities $P_L(n, t)$ that a left auxiliary tree t is inserted into the internal nonterminal node n, and the probability $P_{NL}(n)$ that n takes no left adjunction, so that $P_{NL}(n) + \Sigma_{t \in A_L} P_L(n, t) = 1$ where A_L is the set of left auxiliary trees. The distributions P_R and P_{NR} are defined analogously to cover the same situation for the right-hand side.

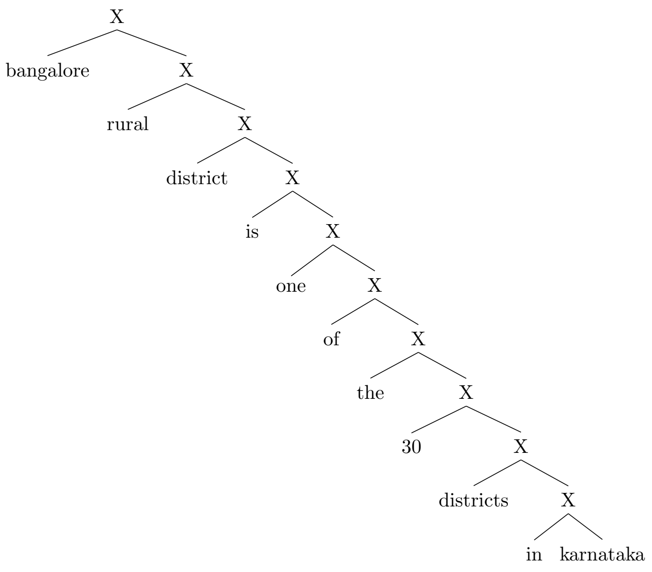
4. The probabilities $P_{LR}(n)$ and $P_{RL}(n)$ that a simultaneous adjunction into the internal nonterminal node n makes a left-adjunction first or a right-adjunction first respectively, so that $P_{LR}(n) + P_{RL}(n) = 1$.

The configuration of the elementary trees affects the possible parse trees that can be derived. Figure 4a shows a parse tree where the elementary trees only allow right-adjunction, which effectively simulates an n-gram. Figure 4b shows another parse tree where the elementary trees allow both left-adjunction and right-adjunction. In this case, the tree-insertion grammar is able to represent hierarchical language structures that n-grams are unable to capture.

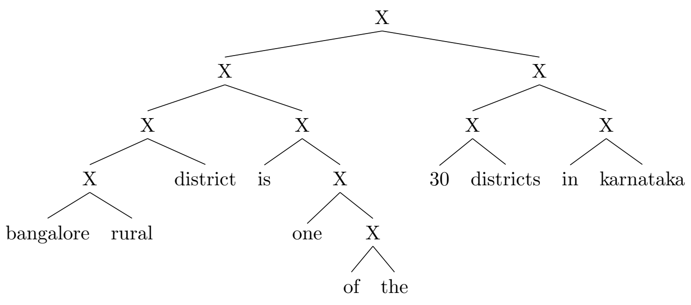
3 Method

The experiments were conducted within the ASR framework CMU Pocket-Sphinx [Lamere et al. 2003]. It was chosen because it is open source, is light-weight for an ASR system, and supports n-grams up to the level of quintgrams. In the initial experiments, the language models were trained on an unannotated corpus, consisting of $10,000$ sentences collected from the English section of Wikipedia. The corpus contains $190,600$ words, divided over $28,500$ unique tokens. The 100 most common tokens (e.g., *the* and *of*) make up half the corpus, whereas two-thirds of the tokens are seen only once (e.g., *sideward* and *boreal*).

In later experiments, a medium-sized corpus of $20,000$ sentences (i.e. an additional $10,000$) and a larger corpus of $50,000$ sentences (i.e. $30,000$ sentences on top of the medium-sized corpus) were similarly collected. The $20,000$ sentence set used over $44,000$ unique tokens, the $50,000$ sentence one used over $79,000$.



(a)

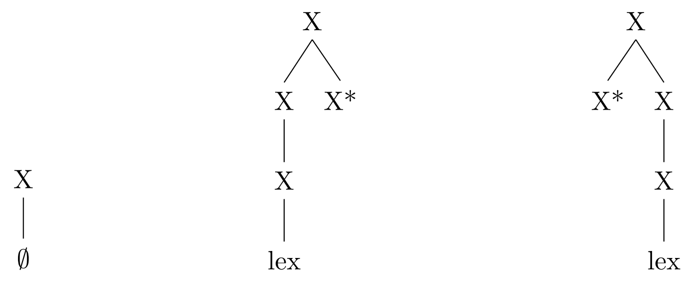


(b)

Figure 4: Two alternative derivations of a structural tree for the sentence "Bangalore Rural District is one of the 30 districts in Karnataka."

3.1 Training

The n-gram models for $n \in \{1, \ldots, 5\}$ were read off the corpus by counting frequencies, and tuned with Witten-Bell smoothing. Inter-sentence punctuation and capitalisation were discarded, because these cues are not accessible from the input audio in most speech recognition systems. This means that the words and abbreviations AU, Au, and au were all represented as au, and the sentence "Guatemala qualified a full team of 4 athletes, 2 men and 2 women." as “guatemala qualified a full team of 4 athletes 2 men and 2 women”.



(a) Initial tree (b) Left-auxiliary tree (c) Right-auxiliary tree

Figure 5: The main elementary tree types. An initial tree (a) with a single empty lexical item, a left-auxiliary tree (b) and a right-auxiliary tree (c).

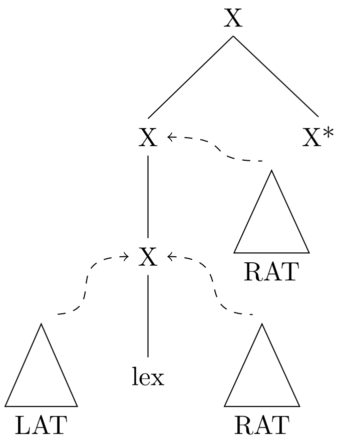
For the PLTIG, we started from a set of prototypical trees. The set contains a single initial tree connected to the empty lexical, representing the start of a sentence. For each lexical entry we also included a left-auxiliary and a right-auxiliary tree (see Figure 5 for an illustration). Like Hwa, we use the left-one right-two (L1R2) insertion paradigm, permitting (but not enforcing) one left adjunction and two right adjunctions into each auxiliary tree [Hwa 2001] (see Figure 6 for an illustration). The PLTIG was then trained through an Expectation-Maximization (EM) process. The EM training algorithm [Dempster et al. 1977] is a hill-climbing algorithm which is used when inducing PLTIGs. The algorithm is based on maximum likelihood estimation (MLE) and determines the adjunction probabilities of a locally optimal grammar. A parser based on the Cocke-Younger-Kasami (CYK) algorithm was used to parse the induced grammar.

The EM algorithm has three steps:

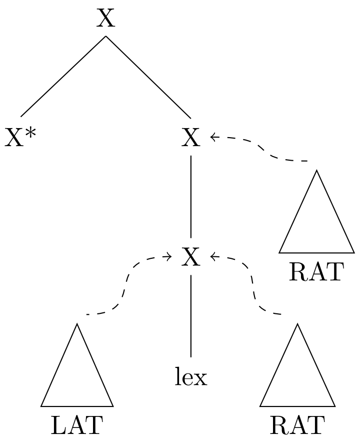
Initialization. Create an initial grammar that is able to create any string, with uniform likelihood of insertion events.

Expectation. Compute the probability that each lexical tree is used when parsing the training sentences.

Maximization. Update parameters based on the outcome of the previous step to maximize the probability of generating the training sentences.



(a) L1R2 left-auxiliary tree



(b) L1R2 right-auxiliary tree

Figure 6: L1R2 insertion allows at most one left and two right adjunctions into each auxiliary tree. LAT and RAT are short for left- and right-auxiliary tree.

Table 1: The percentage of correctly completed evaluation tasks for differently sized data sets (measured in no. sentences).

	10,000	20,000	50,000
PLTIG	96.6 %	97.2 %	97.8 %
unigram	96.5 %	96.9 %	97.0 %
bigram	97.5 %	97.9 %	98.2 %
trigram	97.4 %	97.8 %	98.3 %
quadgram	97.5 %	97.8 %	98.3 %
quintgram	97.5 %	97.8 %	98.3 %

The expectation and maximization steps are run repeatedly until it converges on a local maximum, whereupon the resulting model is smoothed through linear interpolation [Hwa 2001].

3.2 Evaluation

For the evaluation, a fresh set of 150 sentences was randomly selected from the English Wikipedia. A separate corpus was then created for each sentence s, containing on average ca. $3,200$ similar sentences (a total of $479,213$). Each alternative was obtained by replacing a word in s with a different word, also in the original vocabulary and at most two edits (i.e., letter insertions, deletions, or replacements) from the original. This would e.g. turn the template sentence

guatemala qualified a full team of 4 athletes 2 men and 2 women

into the alternatives

guatemala qualified a mule team of 4 athletes 2 men and 2 women

guatemala qualified 88 full team of 4 athletes 2 men and 2 women

guatemala qualified ham full team of 4 athletes 2 men and 2 women

The corpora were manually checked to make sure that the substituted words were inappropriate for their context. A few exceptions may have been overlooked, but in the vast majority of the cases, the alternatives were inferior.

The six language models, i.e. the unigram through quintgram ones and the PLTIG one, were used to compute the probability of each original sentence and its alternatives. In the case of PLTIGs, only the most likely parse tree was considered. The probabilities of the alternatives were then compared against the probability of the original sentence, and the number of sentences with higher and lower probability, respectively, were recorded.

4 Results

The first column of Table 1 shows the outcome of the experiments on the $10,000$-sentence dataset. The number reported for each language model (LM) is the percentage of correctly completed evaluation tasks. As the reader may recall from Section 3.2, these consist in selecting the most likely sentence out of a set of similar but inferior alternatives. The PLTIG language model performs worse than most n-gram models, achieving 96.6 % correct results compared to unigrams with 96.5 % and quintgrams with 97.5 % correct results.

As the data set was comparatively small, it cannot be expected to saturate the more sophisticated language models. In particular the higher-level n-gram models and the PLTIG model are sensitive to overfitting because of their complexity. The experiments were therefore repeated for two larger datasets, a medium-sized one of $20,000$ sentences and the larger one of $50,000$ sentences (described in Section 3). For the medium-sized dataset, the difference between PLTIGs and n-grams shrunk to less than one percentage point—97.2 % for PLTIGs versus 97.9 % for the best-performing n-gram model. For the largest dataset, PLTIGs perform almost as well as the best-performing n-gram model.

The scores of the PLTIG and n-gram models for different training data sets as given in Table 1 are depicted visually on the left of Figure 7. The right graph shows the *difference* between the score of each model and the best-performing

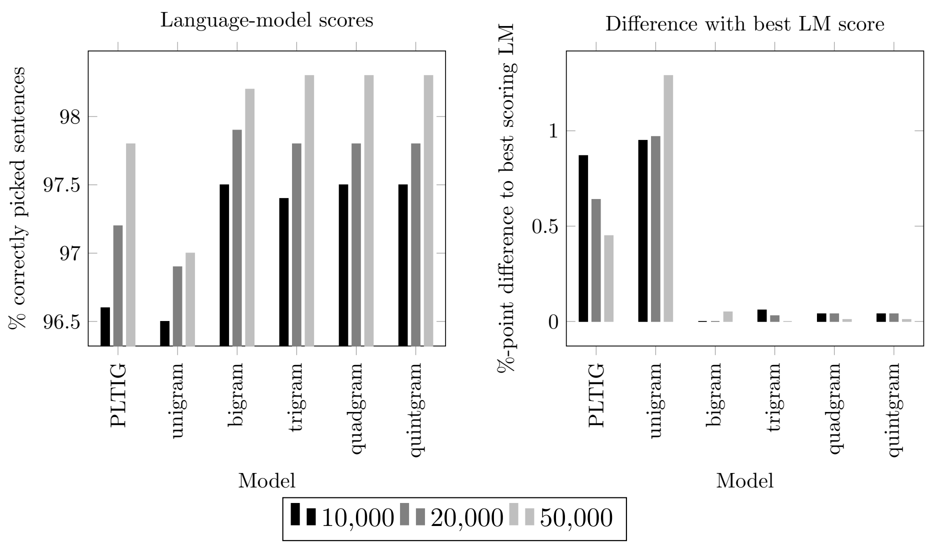


Figure 7: LM scores picking correct sentence out of test set, and distance of particular LM's score to best-scoring LM's score, for LMs trained on data sets of $10,000$, $20,000$, $50,000$ sentences.

model trained on the same data set. This graph shows that the bigram model scores best for the small and medium-sized data set, but that the trigram model does so for the larger data set. The graphs also suggest that while the PLTIG LM is initially inferior to the higher order n-gram models, it benefits more from added data, so the gap to the best-scoring n-gram decreases rather rapidly.

Another way to evaluate the language models is to look at their perplexity relative to the test sentences. This is simply the cross-entropy between the probability distributions predicted by the models and those embodied by the test data. Although perplexity is more loosely correlated with performance in speech recognition systems, in general lower perplexity implies better prediction. Given the likelihood $P(w)$ of a sentence w with respect to a language model M, we compute the perplexity $PP(w)$ of M on w as $\sqrt[n]{1/P_M(w)}$ where $n = |w|$.

The average perplexities of the PLTIG and n-gram models are given in Table 2. We see that on average, the 'surprise' expressed by the PLTIG model is a factor 25 times higher than expressed by the n-grams models. Despite this, the PLTIG model performs reasonably well in the sentence-selection test, especially for the larger data sets. Among the n-gram models, the correlation between perplexity and predictive power is stronger, though not perfect: For example, for the largest data set, the quadgram has the lowest perplexity, but the trigram the highest accuracy.

Table 2: The average perplexities of the n-gram and PLTIG language models on the sentences in the test set.

	PLTIG	unigram	bigram	trigram	quadgram	quintgram
10,000 sent.	49,787	2,126	1,710	1,809	1,826	1,828
20,000 sent.	50,499	2,096	1,208	1,295	1,320	1,323
50,000 sent.	51,505	2,146	1,013	1,079	1,105	1,109

We must also remember that perplexity is strongly affected by smoothing, which decides how much probability mass to reserve for unseen constructions. The n-gram models in CMU sphinx are smoothed with Witten-Bell, but this method is not applicable to PLTIGs, so linear interpolation is used instead (see Section 3). We leave it as an open question to investigate what impact different choices of smoothing have on the perplexity.

In the cases where the models assigned a higher probability to a modified sentence, it is instructive to look at the words that differed between the sentences, and which caused the variation to receive a higher probability. We consider in particular words most often erroneously replaced or substituted in, for the PLTIG model and the trigram model. The top favored words substituted in are *'in', 'of', 'a', 'to', 'the', 'is', 'on', 'he', 'as', 'and'* for the PLTIG model, and for the trigram model they are *'a', 'in', 'of', 'on', 'to', 'at', 'is', 'as', 'the', 'he'*. The top favored words for each consist mostly of common two-letter words, and to a lesser degree single-letter or three-letter ones. This is because with only two edit operations, there are more ways of turning short lexical entries into other short lexical entries, than there are of turning long ones into other valid entries. For instance, every two-letter word can be turned into every other two-letter word.

The top unfavored words erroneously replaced on the other hand, mostly consist of short uncommon words. For the PLTIG model, the top unfavored words erroneously replaced are *'au', 'pan', 'fit', 'jun', 'on', 'bit', 'mono', 'it', 'bad', 'pit'*, for the trigram model they are *'au', 'cpr', 'jun', 'mono', 'bit', 'pan', 'ani', 'pit', 'fit', 't'*. These are also short, as our generation process puts them within edit distance two from the original, but most of them are rare in literary text. Wikipedia is full of acronyms and abbreviations[1], but these are typically much rarer than common short words such as those at the top of the favored lists, yet often within edit distance two from such words.

[1] For example, 'au' is the name of, or short form for, 70 different entities, including gold, absorbance unit, and the series "The Age of Ultron" by Marvel Comics.

5 Conclusion and future work

In our experiments, the n-gram models out-performed the PLTIG model on the smaller data set, but as the size of data grew, the PLTIG model gave comparable results, and its score grew faster with increasing data set size than that of the n-gram models. A natural question to ask is whether the PLTIG model will eventually overtake the n-grams. Judging by the rate with which the accuracies are improving, this will likely happen when the data set contains a couple of hundred thousand sentences, if it happens at all. As parts of the code base used for the experiments are old and not very efficient, experiments on this scale will require a substantial but likely rewarding re-implementation. If it indeed turns out that PLTIGs surpass n-grams at language modeling, then the fact that they provide structural information about sentences will make them an attractive alternative for practical language processing.

The introduction of finite-state machinery has led to improvements in several NLP tasks [Maletti 2015]. In the current work, all internal nodes are labelled with the same non-terminal symbol, so the possibility of distinct internal symbols is not exploited. We are therefore interested in learning strategies that infer richer auxiliary trees, e.g., through a split and merge approach [Petrov 2012]. One may also attempt to infer the number of adjunction sites and the insertion-strategy that offers the best trade-offs between parsing complexity and precision.

References

[Abeillé et al. 1990] Abeillé, A., Schabes, Y., Joshi, A. K.: "Lexicalized tags for machine translation"; 13th Conf. on Computational Linguistics; 1–6; ACL, Stroudsburg, PA, US, 1990.

[Bell et al. 1990] Bell, T. C., Cleary, J. G., Witten, I. H.: Text Compression; Prentice-Hall, Inc., Upper Saddle River, NJ, US, 1990.

[Chelba et al. 2013] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., Robinson, T.: "One billion word benchmark for measuring progress in statistical language modeling"; arXiv preprint arXiv:1312.3005; (2013).

[Dempster et al. 1977] Dempster, A. P., Laird, N. M., Rubin, D. B.: "Maximum likelihood from incomplete data via the EM algorithm"; Journal of the Royal Statistical Society; (1977), 1–38.

[Goodman 2001] Goodman, J. T.: "A bit of progress in language modeling"; Computer Speech & Language; 15 (2001), 4, 403 – 434.

[Hwa 1998] Hwa, R.: "An empirical evaluation of PLTIGs"; 36th Annual Meeting of the ACL; 557–563; ACL, Stroudsburg, PA, US, 1998.

[Hwa 2001] Hwa, R.: Learning probabilistic lexicalized grammars for NLP; Ph.D. thesis; Harvard University; Cambridge, MA, US (2001).

[Joshi et al. 1975] Joshi, A. K., Levy, L. S., Takahashi, M.: “Tree adjunct grammars”; Journal of computer and system sciences; 10 (1975), 1, 136–163.

[Klakow 1998] Klakow, D.: “Log-linear interpolation of language models”; Int. Symp. on Chinese Spoken Language Processing; 1695–1698; ISCA, 1998.

[Lamere et al. 2003] Lamere, P., Kwok, P., Walker, W., Gouvea, E., Singh, R., Raj, B., Wolf, P.: “Design of the CMU Sphinx-4 decoder.”; 8th European conference on speech communication and technology; ISCA, 2003.

[Maletti 2015] Maletti, A.: “Finite-state technology in NLP”; 20th Int. Conf. on Implementation and Application of Automata, Umeå, Sweden; Umeå, Sweden, 2015.

[Mohri 1997] Mohri, M.: “Finite-state transducers in language and speech processing”; Computational Linguistics; 23 (1997), 2, 269–311.

[Petrov 2012] Petrov, S.: Coarse-to-Fine Natural Language Processing; Theory and Applications of Natural Language Processing; Springer Verlag, Berlin Heidelberg, 2012.

[Resnik 1992] Resnik, P.: “Probabilistic tags as a framework for statistical nlp”; 14th Conf. on Computational Linguistics; 418–424; ACL, Stroudsburg, PA, US, 1992.

[Schabes and Waters 1993] Schabes, Y., Waters, R. C.: “Lexicalized context-free grammars”; 31st Annual Meeting on ACL; 121–129; ACL, Stroudsburg, PA, US, 1993.

[Schabes and Waters 1994] Schabes, Y., Waters, R. C.: “Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced”; Technical Report 13; Mitsubishi Research Laboratories; Cambridge, MA (1994).

[Shieber and Schabes 1990] Shieber, S. M., Schabes, Y.: “Synchronous tree-adjoining grammars”; 13th Conf. on Computational Linguistics; 253–258; ACL, Stroudsburg, PA, US, 1990.

[The Google Books Team 2010] The Google Books Team: “Quantitative analysis of culture using millions of digitized books”; Science; (2010).