Probabilistic Lexicalized Tree-Insertion Grammars in Automatic Speech Recognition

Johanna Björklund and Marcus Karlsson

Department of Computing Science, Umeå University, 901 87 Umeå, Sweden

Abstract. We evaluate probabilistic lexicalized tree-insertion grammars (PLTIGs) on a classification task relevant for automatic speech recognition. The baseline is a trigram model, smoothed through absolute discounting. Both language models are trained on an unannotated corpus, consisting of 10 000 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, approx. 3 200 variations were generated. Each variated sentence was obtained by replacing an arbitrary word by a similar word, chosen to be at most 2 edits from the original. In our experiments, the N-gram model preferred one of these alternative sentences in 43.1 percent of the cases, while the PLTIG was only mistaken in 3 percent of the cases.

1 Introduction

Language models are 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. This means for instance that *for all intents and purposes* should be preferred to *for all intensive purposes*, and *with all due respect* to *with all do respect*. Language models are commonly used in machine translation (MT) 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 practical language modelling is to train structurally-agnostic N-grams on vast 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 N consecutive words in s, with respect to some given set of training data. When a 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. N-grams have the advantages of being surprisingly powerful considering their simplicity, easy to train from data, and can typically be applied to a sentence in linear or log-linear time [12].

On the downside, the size of N-gram models grows exponentially in N. This means that for large values of 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 [7] meant that that proceeding beyond 5-grams is unlikely to be practically motivated. This conjecture has not been contradicted a decade later. The flagship of the field, Google's N-gram viewer, contains 500-billion words compiled from 20 million books. In its construction, the value of N had to be restricted to 5 to limit the model's size, and all N-grams encountered fewer than 40 times were discarded without updating the model [14].

The fact that N has to be kept small causes problems. Take the following pair of sentences:

She *rowed* between Yellowknife and Benchoko in a weathered *canoe*.
She *rowed* between Yellowknife and Benchoko in a 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 9, since no N-gram of shorter length contains both 'rowed' and '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 their 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. CFGs can be lexicalized using e.g. Greibach normal form, but this confuses the syntactical information. Another alternative is to use Tree-adjoining Grammars (TAG). TAGs represents a language of parse trees as a set of parse-trees fragments, together with a set of rewrite rules that regulate how they can be assembled. TAGs were developed by linguistics and have many appealing properties, in particular they are both easy to lexicalize and offer parsing. The problem here is rather the reverse; they are too powerful to be practical with a parsing complexity of $O\big(n^6\big)$.

To find a practical middle ground, Hwa [9] suggested the use of probabilistic lexicalized tree insertion grammars (PLTIG). PLTIGs are a type of probabilistic lexicalized TAG, in which the number of ways in which parse-tree fragments can be combined is limited. They are as good as trigrams at predicting word sequences, but also offer syntax-aware language modelling. Their parsing complexity is $O\big(n^3\big)$, which is equal to that of probabilistic context-free grammars (PCFGs) and a factor n^3 faster than TAGs. PLTIGs also have the advantage that expectation-maximisation (EM) training of PLTIGs on structurally annotated data converges faster than EM training for similarly sized PCFGs (see Chapter 3 of [9]).

In this paper we evaluate PLTIGs as an alternative for N-grams in Automatic speech recognition (ASR), i.e. the translation of spoken words into text. Modern ASR systems are typically build around an acoustic model, a lexical model, and a language model. The acoustic model maps utterances into phonemes, the lexical model concatenates phonemes to match a dictionary of known words, and the language model predicts how words can be combined into sentences. Today, N-grams and Markow models are common in ASR systems, and we are interested to learn how PLTIGs stands up to these. To isolate the influence of the language-models, we focus on a classification task in which the models are to pick out a sentence s from a set of alternatives, generated from s by replacing one or a few words by similar but inappropriate words.

1.1 Background and related work

Automatic speech recognition originates from the post-war era. The scope was initially limited to very small vocabularies, e.g., single numeric digits [13]. An early attempt to build a hardware device capable of recognizing speech was made by Smith in 1951 [20]. His machine passed the input acoustic signal through a sequence of contiguous band-pass filters, generating 32 signals that were fed to a switch selector panel, which in turn mapped their energy distributions to phonemes. Other dedicated recognition devices were constructed in the following years, but computers in the modern sense did not come into play until the 1950s.

In 1959 Forgie and Forgie [6] presented a system similar to Smith's, with the exception that the recognition was done by a computer program running on a general-purpose machine. This meant that, unlike its predecessors, it could easily be adapted to new speakers. The success rate was now up to 93 %, but the program was only capable of differentiating between ten English vowels. Progress continued to be slow throughout the 1960s, and the entire field was heavily criticized for not making any substantial results. Real-world applications were considered distant [15].

Language models came into focus during the 1970s. Notably in 1976, Jelinek [10] introduced the *New Raleigh* language model, which was essentially a hidden Markov model. A similar system had also been presented by Baker the year before [3]. The advantage of these models was that they allowed the ASR system to work comparably well under high error rates. From then on, probabilistic models and specifically N-gram models became a commodity in the field.

Interest in other types of statistical language models increased during the late 1980s, in particular around different types of tree models. Tree-adjoining grammars in particular had already been studied for many years in the context of formal languages [11], but in 1990 Shieber et al. [1, 19] successfully used a variant of it for modelling natural languages. The intention was to use TAGs for a wider range of applications, including semantic interpretation as well as translation. The idea was to use so called *synchronous TAGs* where elementary trees were mapped to a semantic representation unrelated to syntax. Two practical problems with this approach were the computational cost of parsing, and the lack of suitably annotated datasets.

Some of the early work on lexicalized tree insertion grammars were conducted in 1994 by Schabes and Waters [18]. In their paper they mention that the probabilities with which adjunctions and substitutions occur can be controlled by adding a stochastic parameter, just as they had previously done for context-free grammars [17]. Hwa [8] later provided a semi-supervised expectation-maximisation algorithm for probabilistic lexicalized TIGs, and discussed how to infer high-quality grammars with as minimal data annotation. As mentioned previously, parsing with respect to TIGs is substantially easier than with TAGs, but still expensive compared to N-grams.

In the 2000s, the advent of grid and later cloud computing moved the boundaries for what could be considered computationally feasible. As the number of high-quality syntactically and semantically annotated corpora is growing, new possibilities for machine learning open up and we are interested to reconsider TIGs in the light of this.

1.2 Outline

This paper is organised as follows. Section 2 revises the relevant language models. Section 4 describes the experimental setup, and Section 4 reports and discusses the results. Section 5 concludes the paper by outlining directions for future work.

2 Theory

Let us start by recalling the definitions of N-grams, TAGs (as an intermediary step), and finally PLTIGs. Since the theoretical results that are the fundament of this practical study are already in place, e.g., parsing complexity and the correctness of the EM algorithm, it suffices to describe the semantics of the two latter models at a fairly high level, and point the interested reader to e.g. [8] for the formal definitions.

2.1 N-grams

The likelihood $P(w_1^n)$ of a sequence of words w_1^n can be computed using the chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\cdots P(w_n|w_1^n) \tag{1}$$

$$= \prod_{i=1}^{n} P(w_i|w_1^{i-1}) \ . \tag{2}$$

The idea behind N-grams is to approximate the likelihood of each word w_i in w_1^n by

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-N}^{k-1}) \ . \tag{3}$$

Substituting Equation 3 into Equation 2, the likelihood of the entire sequence is approximated by

$$P(w_1^n) \approx \textstyle\prod_{i=1}^{n} P(w_i|w_{i-N}^{i-1}) \ .$$

When the value of N is 1, 2, or 3, the model is often referred to as unigram, bigram, or trigram, respectively.

N-grams are straight-forward to learn from training data through a Maximum Likelihood Estimation (MLE) process. The probability of a word w_i under and N-gram is

$$P_{MLE}(w_i|w_{i-N+1}^{i-1}) = \frac{c(w_{i-N+1}^{i})}{c(w_{i-N+1}^{i-1})} \ ,$$

where $c(u_1^k)$ counts how many times the word sequence u_1^k is seen in the training data.

For most applications, the training data cannot be expected to contain all N-grams that will later be encountered in the application data. As can be expected, this problem of data sparsity diminishes as the size of the training data increases [4], but it seldom disappears completely [2]. 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, e.g. Laplace, Interpolation, Good-Turing, and Kneser-Ney. In this paper, we use absolute discounting, since it is easy to implement and reasonably effective. Smoothing is thus done by decreasing the probability of each observed N-gram by a small constant, and dividing the combined probability mass over the unobserved word combinations.

2.2 Tree Adjoining Grammars

As previously mentioned, probabilistic tree-insertion grammars are a restricted form of tree adjoining grammars [11]. 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 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 non-terminal leaf node of the same type as the root node called the *foot node*, marked with the special symbol $*$. The 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 lead node of some other tree then the first tree can be *substituted* into the nonterminal leaf node of the other tree, a way to rewrite the symbol in a context-free derivation. Figure 2 shows an example of this. The determiner *her* in the initial tree in Figure 2a is substituted 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 has to have the same symbol, auxiliary trees can be inserted into another tree where such symbol exists. An example of this is shown in Figure 3.

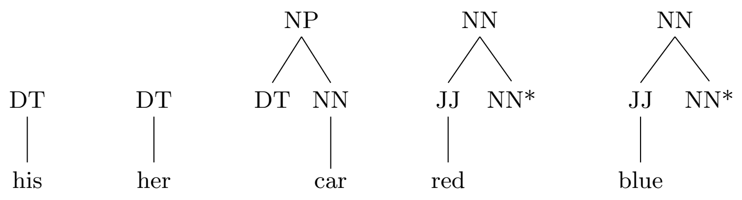

Fig. 1: Examples of elementary trees in a TAG. Elementary trees (a-c) show initial trees, while (d-e) are auxiliary trees. The auxiliary trees each have a foot node marked with a * and is of the same type as the root node.

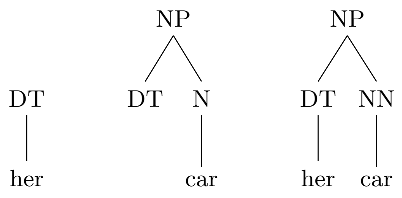

Fig. 2: The initial tree in (a) is substituted into a nonterminal node of the initial tree in (b), with the result of the new tree in (c). This is possible since the root node of (a) is of the same type as the nonterminal leaf node in (b).

Note that the auxiliary tree in Figure 3a has the same root 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 has 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 set of terminal symbols, V is a set of nonterminal symbols, I is the initial trees, A is the 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 TAG (and hence LTAG), in which three probability distributions are included in the definition. A probability Ω is defined as the set of possible substitution and adjunction events. Probabilistic TAG is then defined as the quintuple $(\Sigma, V, I, A, P_I, P_S, P_A)$ 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 [16].

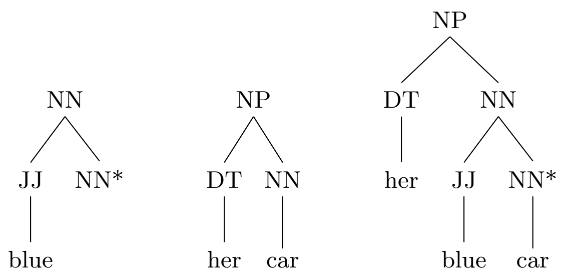

Fig. 3: The initial tree in (a) is adjoined into an intermediary node of the tree in (b), with the result of the new tree in (c). The tree from (a) is inserted into (b) at the intermediary node `NN` since (a) has both a root node and a foot node of that type.

2.3 Tree Insertion Grammars

Tree insertion grammars (TIG) is 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 [18].

- Every auxiliary tree has to be either a *left auxiliary tree* or a *right auxiliary tree*, meaning that they only have lexical items on one 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 node to the foot node 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.
- 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.
- 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.

Two parse trees are 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.

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. There are five probability distributions:

1. The probability $p_I(\rho)$ that the derivation starts with the initial tree ρ, so that $\Sigma_{i=1}^{N} p_I(\rho_i) = 1$ for N initial trees.

2. The probability $p_S(n_s, \rho)$ that an initial tree ρ is substituted into a substitution node n_s, so that $\Sigma_{i=1}^{N} p_S(n_s, \rho) = 1$ for N initial trees.
3. The probability $p_L(n, \rho)$ that a left auxiliary tree ρ is inserted into the internal nonterminal node n, and the probability $p_{NL}(n)$ that n takes no left adjunction, so that $p_{NL}(n) + \Sigma_{i=1}^{N} p_L(n, \rho) = 1$ for N left auxiliary trees.
4. The probability $p_R(n, \rho)$ that a right auxiliary tree ρ is inserted into the internal nonterminal node n, and the probability $p_{NR}(n)$ that n takes no right adjunction, so that $p_{NR}(n) + \Sigma_{i=1}^{N} p_R(n, \rho) = 1$ for N right auxiliary trees.
5. The probability $p_{LR}(n)$ and the probability $p_{RL}(n)$ that a simultaneous adjunction into the internal nonterminal node n makes a left-adjunction first or a right-adjunction first respectively, so that $p_{LR}(n) + p_{RL}(n) = 1$.

All these restrictions means that strings can be parsed with respect to a PLTIG in time $O(n^3)$ instead of the $O(n^6)$ for PLTAGs.

3 Method

In the experiments, both language models are trained on an unannotated corpus, consisting of 10 000 sentences collected from the English section of Wikipedia. The trigram model could easily be read of the corpus by counting trigram frequencies and then using absolute discounting to account for unseen trigrams.

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 4 for an illustration).

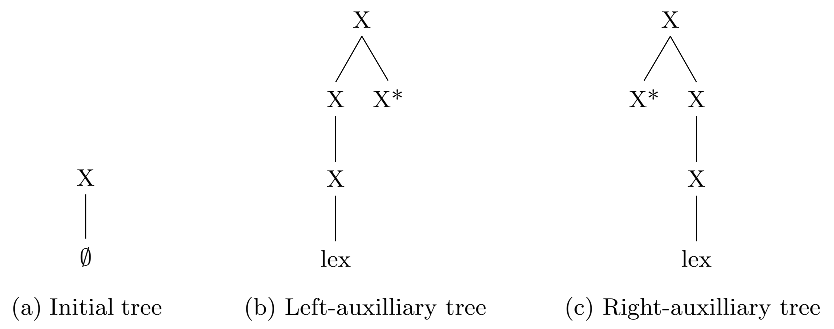

Fig. 4: The main elementary tree types. An initial tree (a) with a single empty lexical item, a left-auxiliary tree (b) and a right-auxiliary tree (c).

The PLTIG was then trained through an Expectation-Maximization (EM) process. The EM training algorithm [5] 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.
Expectation Compute the probability that each lexical tree is used when parsing the training sentences.
Maximization Update parameters based on the outcome of the E-step so that the probability of generating the training sentences is maximized.

The expectation and maximization steps are run repeatedly until it converges on a local maximum.

Data sparsity is also a problem for PLTIGs and can be mitigated using the same types of algorithms. We take after Hwa and use linear interpolation smoothing for PLTIGs due to its consistent performance with different configuration [9].

For the evaluation, a fresh set of sentences were randomly selected from the Wikipedia. Then, a separate corpus was created for each sentence s, containing approx. 3 200 similar sentences. Each alternative sentence was obtained by replacing a word in s by with a different word, at most two edits (i.e., letter insertions, deletions, or replacements) from the original. 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 indeed inferior.

The two language models were used to compute the probability of each original sentence and its alternatives. The probabilities of the alternatives where then compared against the probability of the original sentence. The number of alternative sentences that received a higher probability compared to the original sentence, and the number that received a lower probability were recorded.

4 Results

The outcome of our experiments was that the N-gram model assigned a higher probability to 43.1 percent of the alternative sentences, while the PLTIG assigned a higher probability to just 3.4 percent of them. The PLTIG based model must thus be said to outperform the N-gram model.

In the cases where either model assigned a higher probability to an alternative sentence, it is instructive to look at the word that differed between the two sentences, and which caused the alternative to receive a higher probability. Table 1 shows a listing of the most frequent of these words.

In the N-gram case, almost all the words in the list follow a common theme. They are all relatively short words that tend to be common in English. Some of the words in the top of the list also has a very high count, actually the summed proportion of the first seven words is more than 50 percent. Our interpretation

Table 1: The table illustrates the cases where and an alternative sentence was assigned a higher probability. The first column lists the words replaced most often, the second column shows the number of alternative sentences that received a higher likelihood than the original sentence, and the third column gives corresponding the percentage.

(a) N-gram

Word	Sentences	Percentage
of	22474	10.9
a	20824	10.1
the	18532	9.0
is	15995	7.7
in	13820	6.7
be	7397	3.6
as	7123	3.4
an	6422	3.1
at	5994	2.9
he	5153	2.5
i	4176	2.0
was	4129	2.0
it	3862	1.9
by	3284	1.6
for	3250	1.6
or	3174	1.5
my	2534	1.2
has	2152	1.0
had	1463	0.7
are	1259	0.6
his	1258	0.6
one	1173	0.6
and	1111	0.5
that	1040	0.5
do	976	0.5
on	957	0.5
this	912	0.4
who	850	0.4
9	830	0.4
all	824	0.4
11	781	0.4
any	768	0.4

(b) PLTIG

Word	Sentences	Percentage
au	678	4.2
bit	450	2.8
19	379	2.3
15	290	1.8
z	277	1.7
93	277	1.7
ani	268	1.6
87	245	1.5
62	240	1.5
jun	239	1.5
pit	233	1.4
cpr	232	1.4
25	204	1.3
on	175	1.1
27	157	1.0
boy	154	0.9
my	153	0.9
at	153	0.9
30	149	0.9
9	148	0.9
sons	146	0.9
nor	136	0.8
in	134	0.8
23	131	0.8
22	131	0.8
t	130	0.8
mono	129	0.8
fans	129	0.8
0	126	0.8
is	125	0.8
pov	123	0.8
4th	123	0.8

is that common words can be seen in many different contexts, and this makes it difficult for the model to predict what is to follow.

The same list for the PLTIG is appears more diverse, but several of the items are numbers. This suggests that since the each single number did not occur very often, the model had not enough data to learn how it was connected with other words. It may therefor be useful to pre-process the corpus buy grouping uncommon words into categories, i.e., numbers, colours, etc., and replacing instances of each such category by a token representing the entire category.

5 Conclusion and future work

It is not surprising that PLTIGs outperform N-grams, considering that it is after all a more expressive model. What to us is surprising is the extent to which it does so. It is also interesting to see that the two language models appear to make different kinds of mistakes. Whereas N-grams are thrown by frequently used words, PLTIGs stumble on rare words.

Encouraged by the outcome of the current study, we have begun work on an implementation of PLTIGs in the open-source ASR system Kaldi. The aim is to conduct practical experiments in the wild, to see if PLTIGs are efficient enough to be useful, and if the results we saw here carry over.

In this work, we have followed Hwa and used three adjunction sites in each elementary tree. It may be worthwhile to conduct a separate study in which the number of adjunction sites are varied, to see how their degree affects the accuracy and efficiency of the model.

References

[1] A. Abeillé et al. “Using Lexicalized Tags for Machine Translation”. In: *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*. COLING ’90. Stroudsburg, PA, USA: Association for Computational Linguistics, 1990, pp. 1–6.

[2] B. Allison et al. “Another look at the data sparsity problem”. In: *Text, Speech and Dialogue*. Springer Berlin Heidelberg, Sept. 2006, pp. 327–334.

[3] J. Baker. “The DRAGON system–An overview”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 23.1 (Feb. 1975), pp. 24–29.

[4] M. Banko et al. “Mitigating the Paucity of Data Problem”. In: Jan. 2001.

[5] A. P. Dempster et al. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1977), pp. 1–38.

[6] J. W. Forgie et al. “Results obtained from a vowel recognition computer program”. In: *The Journal of the Acoustical Society of America* 31.11 (1959), pp. 1480–1489.

[7] J. T. Goodman. “A bit of progress in language modeling”. In: *Computer Speech & Language* 15.4 (2001), pp. 403–434.

[8] R. Hwa. "An Empirical Evaluation of Probabilistic Lexicalized Tree Insertion Grammars". In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. ACL '98. Stroudsburg, PA, USA: Association for Computational Linguistics, 1998, pp. 557–563.

[9] R. Hwa. "Learning probabilistic lexicalized grammars for natural language processing". PhD thesis. Cambridge, MA: Harvard University, Mar. 2001.

[10] F. Jelinek. "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4 (Apr. 1976), pp. 532–556.

[11] A. K. Joshi et al. "Tree adjunct grammars". In: *Journal of computer and system sciences* 10.1 (1975), pp. 136–163.

[12] D. Klakow et al. "Log-Linear Interpolation of Language Models". In: *Proceedings of the International Symposium on Chinese Spoken Language Processing*. 1998, pp. 1695–1698.

[13] T. Marill. "Automatic recognition of speech". In: *IRE Transactions on Human Factors in Electronics* 1 (1961), pp. 34–38.

[14] J.-B. Michel et al. "Quantitative Analysis of Culture Using Millions of Digitized Books". In: *Science* (2010). URL: `http://www.sciencemag.org/content/331/6014/176.full`.

[15] J. R. Pierce. "Whither Speech Recognition?" In: *The Journal of the Acoustical Society of America* 46.4B (1969), pp. 1049–1051.

[16] P. Resnik. "Probabilistic Tree-adjoining Grammar As a Framework for Statistical Natural Language Processing". In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*. COLING '92. Stroudsburg, PA, USA: Association for Computational Linguistics, Aug. 1992, pp. 418–424.

[17] Y. Schabes et al. *Stochastic lexicalized context-free grammar*. Tech. rep. 12. Cambridge, MA: Mitsubishi Electric Research Laboratories, July 1993.

[18] Y. Schabes et al. *Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced*. Tech. rep. 13. Cambridge, MA: Mitsubishi Electric Research Laboratories, June 1994.

[19] S. M. Shieber et al. "Synchronous Tree-adjoining Grammars". In: *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*. COLING '90. Stroudsburg, PA, USA: Association for Computational Linguistics, 1990, pp. 253–258.

[20] C. P. Smith. "A Phoneme Detector". In: *The Journal of the Acoustical Society of America* 23.4 (1951), pp. 446–451.