Lexicalization of Probabilistic Grammars

Helmut Schmid
Institute for Computational Linguistics
University of Stuttgart
Azenbergstraße 12, 70174 Stuttgart, Germany
schmid@ims.uni-stuttgart.de

Abstract

Two general methods for the lexicalization of probabilistic grammars are presented which are modular, powerful and require only a small number of parameters. The first method multiplies the unlexicalized parse tree probability with the exponential of the mutual information terms of all word-governor pairs in the parse. The second lexicalization method accounts for the dependencies between the different arguments of a word. The model is based on a EM clustering model with word classes and selectional restrictions as hidden features. This model is useful for finding word classes, selectional restrictions and word sense probabilities.

1 Introduction

Unlexicalized probabilistic grammars like probabilistic context-free grammars (PCFGs) fail to disambiguate frequent syntactic ambiguities like PP-attachment and coordination ambiguities correctly, because they lack the necessary information about lexical dependencies. Compare e.g. the sentences *He ate the cake with a spoon* and *He ate the cake with apricot glaze*. A simple unlexicalized PCFG would assign the same structure to both sentences. Correct disambiguation of these sentences requires the knowledge that a *spoon* is a likely nominal head of *with* PPs modifying the verb *eat*, whereas *glaze* is a likely nominal head of *with* PPs modifying the noun *cake*.

Probabilistic grammars are usually lexicalized (Charniak, 1997; Collins, 1997; Carroll and Rooth, 1998) by replacing the grammar symbols with pairs consisting of a grammar symbol and a lexical head. The rule `VP` $\rightarrow$ `V NP` e.g. is replaced by a set of rules of the form $\langle$`VP,eat`$\rangle$ $\rightarrow$ $\langle$`V,eat`$\rangle$ $\langle$`NP,cake`$\rangle$. Because this results in a data sparseness problem, the lexical heads of the arguments are usually assumed to be independent given the left-hand side of the rule (or the whole rule). In the simplest case, a probabilistic grammar with two types of rules is obtained. Rules of the first type select a grammar rule given the lexical head. They are exemplified by the rule $\langle$`VP,eat`$\rangle$ $\rightarrow$ $\langle$`V,eat`$\rangle$ $\langle$`NP,VP,eat`$\rangle$. Rules of the second type choose the lexical head of an argument. They are exemplified by the rule $\langle$`NP,VP,eat`$\rangle$ $\rightarrow$ $\langle$`NP,cake`$\rangle$.

This type of lexicalization has the disadvantage that each unlexicalized parameter is replaced by a large number of lexicalized parameters.

This paper presents an alternative method where the lexicalized model is a modular extension of the unlexicalized model and where the number of parameters is easy to control. The lexicalized probability of a parse is obtained by multiplying its unlexicalized probability with the exponential of the pointwise mutual information of each word and its lexical governor. The number of parameters is small because individual parameters are only required for pairs whose frequency differs significantly from the expected frequency under the assumption of independence. The model assumes that the different arguments of a word are statistically independent given the word.

A second lexicalization method is described which captures the dependencies between the arguments of a word. It is based on a EM clustering model and is expected to have good smoothing properties.

2 Lexicalization With Governors

The first lexicalization method imposes the following rather general restrictions on a probabilistic grammar.

1. The grammar assigns an (unlexicalized) probability $p(T)$ to each parse tree T.
2. The grammar assigns to each word `w` in a parse tree `T` at most one governor `g`.

There are no other restrictions on the definition of the governor than its uniqueness (which will later be weakened). Governors may be words, lemmas or pairs consisting of a word/lemma and a grammatical function like subject, indirect object etc. For efficiency reasons, it is desirable that an efficient algorithm for the labeling of parse forests with governors is available.

In order to motivate our solution, assume for the moment that a PCFG is to be lexicalized and that the words are generated by unary lexical rules `C` $\rightarrow$ `w`. We see in eq. 1 that the lexicalized probability $p(C \rightarrow w|C,g) = p(w|C,g)$ is identical to the product of the unlexicalized probability $p(C \rightarrow w|C) = p(w|C)$ and $EMI(w,g|C)$, the exponential of the pointwise mutual information between `w` and `g` given `C`.

$$\begin{aligned} p(w|C,g) &= p(w|C)\frac{p(w,g|C)}{p(w|C)p(g|C)} \\ &= p(w|C)\, EMI(w,g|C) \end{aligned} \quad (1)$$

Therefore, the lexicalized probability of a parse tree is its unlexicalized probability multiplied by the EMI factors of the word-governor pairs. In order to simplify the model, a dummy governor is assigned to words without governor. The resulting formula for the computation of the lexicalized parse probability is shown in eq. 2, where n is the sentence length and w_i, C_i and g_i are the i-th word, its category and its governor, respectively.

$$P_{lex}(T) = P_{unlex}(T) \prod_{i=1}^{n} EMI(w_i, g_i|C_i) \quad (2)$$

2.1 Parameter Training

The EMI parameters are either estimated from unparsed corpora using the Expectation Maximization (EM) algorithm or from treebank data.

Treebank training requires a treebank which is annotated with governor information. If governor information is missing, it has to be added by running the annotation program whose availability is presumed. Given the annotated parse trees, the training algorithm counts how often a category C with governor g is expanded to some word w.

For each tuple $\langle w, C, g\rangle$ with $f(w,C,g) > 0$, the algorithm checks whether its frequency diverges significantly from the expected value under the assumption that w is independent of g given C. To this end, it computes the probability that a sample of size $n = \sum_{w'} f(w',C,g)$ which is randomly drawn from a Bernoulli distribution with probability $p = p(w|C)$ contains $m = f(w,C,g)$ or more "1" events. To this end, it sums the probabilities of the binomial distribution $b(r;n,p) = \binom{n}{r} p^r (1-p)^{n-r}$ for all $r \geq m$.[1] If this sum is below e.g. 5 % (the usual threshold for significance), the algorithm estimates a separate EMI parameter for this tuple according to eq 3.

$$\begin{aligned} f(g,C) &= \sum_w f(w,g,C) \\ f(w,C) &= \sum_g f(w,g,C) \\ f(C) &= \sum_w f(w,C) \\ p(w,g|C) &\approx \frac{\hat{f}(w,g,C)}{f(C)} \\ p(w|C) &\approx \frac{f(w,C)}{f(C)} \\ p(g|C) &\approx \frac{f(g,C)}{f(C)} \\ EMI(w,g|C) &= \frac{p(w,g|C)}{p(w|C)p(g|C)} \\ &= \frac{\hat{f}(w,g,C)f(C)}{f(w,C)f(g,C)} \end{aligned} \quad (3)$$

The frequencies $\hat{f}(w,g,C)$ are smoothed frequencies obtained by absolute discounting (Ney et al., 1994) or similar smoothing methods (see e.g. (Chen and Goodman, 1996)). All words w which are not in the set $W_s(g,C)$ of words with significant frequency, share a single parameter $EMI(w,g|C) = EMI(-,g|C)$ for a given context C and governor g. In order to obtain a probability distribution, this parameter has to

[1] Usually it is easier to compute the equivalent quantity $1 - b(< m; n, p)$.

be defined such that eq. 4 holds.

$$\sum_w p(w|c) EMI(w, g|C) = 1 \quad (4)$$

Solving this equation for the unknown parameter $EMI(-, g|C)$ gives the following formula for the computation of this parameter:

$$\frac{1 - \sum_{w \in W_s(g,C)} p(w|C) EMI(w, g|C)}{1 - \sum_{w \in W_s(g,C)} p(w|C)} \quad (5)$$

Because EMI parameters are only stored for word-governor pairs with significant frequency, the number of parameters remains small.

2.2 Multiple Governors

Linguistic grammars often analyze control verbs like `to promise` in such a way that an argument is governed by two verbs. The noun phrase `Peter` in the sentence `Peter promised to come` e.g. is governed by the two verbs `promised` and `come`.

The lexicalized probability of an argument which is governed by two words is derived in the following way: (For simplicity, we omit here the C on which all probabilities depend.)

$$\begin{aligned} p(w|g_1, g_2) &= p(w) \frac{p(w, g_1)}{p(w)p(g_1)} \frac{p(w, g_2)}{p(w)p(g_2)} \\ &\quad \frac{p(w, g_1, g_2)p(w)}{p(w, g_1)p(w, g_2)} \frac{p(g_1)p(g_2)}{p(g_1, g_2)} \\ &= p(w)\, EMI(w, g_1)\, EMI(w, g_2) \\ &\quad \frac{EMI(g_1, g_2|w)}{EMI(g_1, g_2)} \end{aligned} \quad (6)$$

We obtain an expression similar to the one in eq. 1, but with two additional terms. The first additional term is the EMI score of the second governor. The second term captures the fact that the relation between w, g_1 and g_2 is not fully described by the pair-wise dependencies.

How could the second parameter be estimated? Estimating a separate parameter for each triple consisting of an argument and a governor pair is difficult due to data sparseness problems, although one of the governors has to be a control verb. Again, it seems a good idea to estimate this parameter $\Theta(w, g_1, g_2) = EMI(g_1, g_2|w)/EMI(g_1, g_2)$ only for the small number of triples whose frequency diverges significantly from the expected value. To this end, we check whether $b(r; n, p) < 0.05$ for $r = f(w, g_1, g_2)$, $n = f(g_1, g_2)$ and $p = p(w)EMI(w, g_1)EMI(w, g_2)$.

The other words w use the same parameter $\Theta(w, g_1, g_2) = \Theta(-, g_1, g_2)$ whose value is chosen such that the following expression becomes 1:

$$\sum_w p(w)\ EMI(w, g_1)\ EMI(w, g_2)\ \Theta(w, g_1, g_2) \quad (7)$$

Solving this equation for the unknown parameter $\Theta(-, g_1, g_2)$ gives the expression in eq. 8 with $p^*(w) = p(w)EMI(w, g_1)EMI(w, g_2)$.

$$\frac{1 - \sum_{w \in W_s(g_1,g_2)} p^*(w)\Theta(w, g_1, g_2)}{1 - \sum_{w \in W_s(g_1,g_2)} p^*(w)} \quad (8)$$

For rare governor pairs, the value of $\Theta(-, g_1, g_2)$ has to be computed at run-time.

3 Lexicalization With Arguments

So far, only the dependence of words on their governors (knife – cut, bread – cut) was considered. The different arguments of a word (knife, bread) were implicitly assumed to be independent of each other given the word. This is an oversimplification. Knifes are likely to cut bread, but managers are more likely to cut jobs.

We drop the independence assumption and consider lexicalized models where a word w depends on its arguments $a_1, \ldots, a_n$. So, we are interested in the probability $p(w|a_1, \ldots, a_n, C)$ which is similar to the probability of words with multiple governors. However, the proposed solution for multiple governors would not work here, because multiple arguments are much more frequent and because verbs have up to three arguments.

3.1 Incorporation of a Clustering Model

Instead, the probability $p(w|a_1, \ldots, a_n, C)$ is directly estimated with an EM clustering model (EMC) (see e.g. (Rooth, 1998)) which decomposes the probability $p(w, a_1, \ldots, a_n, C)$ as follows:

$$\sum_c p(c)\ p(w|c)\ p(C|c) \prod_{i=1}^{n_C} \sum_r p(r|c, i, C)\ p(a_i|r) \quad (9)$$

We assume here that the number of arguments is fixed for each category C. The decomposition introduces the hidden features c and r.

The clusters c are interpretable as word classes and the r's are argument classes (= selectional restrictions). r might be e.g. a concept of a taxonomy like Wordnet (Miller et al., 1993). The probability of a word-argument tuple is obtained by summing over all clusters and all argument classes for each argument. The probability $p(w|a_1, \ldots, a_n, C)$ is computed as

$$p(w|a_1, \ldots, a_n, C) = \frac{p(w, a_1, \ldots, a_n, C)}{p(a_1, \ldots, a_n, C)} \quad (10)$$

and the probability $p(a_1, \ldots, a_n, C)$ is given by

$$\sum_c p(c)p(C|c) \prod_{i=1}^{n_C} \sum_r p(r|c, i, C)\ p(a_i|r) \quad (11)$$

A separate set of clusters could be used for each category C, or, alternatively, a single cluster set for each set of similar categories like e.g. the verbal categories. There are no other prior restrictions on the assignment of words and argument classes to clusters, i.e. on the probabilities $p(w|c)$ and $p(r|c, i, C)$. They are learned during training.

The dependencies between the different argument positions of a word are modeled by the clusters. Training with the EM algorithm makes the clusters of an EMC model homogeneous, i.e. the words, categories and argument classes with a high probability for a given cluster are similar to each other. So, each cluster has high probabilities for a certain group of words w, a certain group of categories C, and certain argument classes. Word-argument-tuples which fit the restrictions of a cluster well, will have a high probability. Words with multiple readings are likely to have large probabilities for different clusters modeling the different readings.

The distribution $p(a|r)$ of lexical heads a for a given argument class r is shared by all cluster sets. $p(a|r)$ is only positive if r dominates a in the taxonomy.

3.2 Advantages

The resulting probability distribution will be smooth because

- the dependencies between words and their arguments are modeled through the clusters which contain other words with similar arguments. An argument which occurred with one of the words is likely to occur with the others, too.
- words select argument classes rather than individual words as arguments. If a word occurs with some argument a, then it is likely to occur also with semantically similar arguments.

The introduction of hidden features reduces the number of parameters because

- the argument classes (= selectional restrictions) depend only on the cluster and not on the governor or the classes of the other arguments.
- the lexical heads of the arguments depend only on the argument class and are further restricted by the taxonomy.

The lexicalization based on the EMC model has the following advantages compared to the previous model:

- It models dependencies between arguments.
- It models multiple readings of a word.
- Word classes are learned.
- Selectional restrictions are learned.
- The resulting probability distribution is smooth.

The components of the model are useful for other applications:

- The $p(w|c)$ probabilities allow different readings of a word to be separated.
- $p(C|c)$ could be useful for finding frequent alternations like the causative-inchoative alternation.
- $p(r|c, i, C)$ yields selectional restrictions for arguments.
- $p(a|r)$ could be used to obtain probability estimates for the different senses of a word.

3.3 Interpolated Models

The EMC model smooths the probability distribution. For fixed expressions like *"kick the bucket"* or *"spill the beans"* the resulting probability distribution might actually be too smooth. Such phrases show very specific selectional properties

which are not shared by other words and therefore modeled badly. This problem can be solved with the following method:

1. Compare the observed frequency of each word-argument tuple with the expected value according to its model-based probability.
2. If there is a significant difference, add a new cluster which generates only this tuple.

The resulting model is basically a weighted mixture of an EMC model and a family of distributions $p(w|C, a_1, \ldots, a_n)$ which are non-zero for a small number of tuples. The latter parameters model fixed phrases whereas the EMC model assigns probabilities to the remaining data and to the literal meanings of phrases.

3.4 Training

Because of the hidden parameters, the EMC model requires iterative EM training when it is trained on a treebank. Based on (estimates of) the frequencies of the word-argument-tuples in a corpus, the expected frequencies $f(c, w)$, $f(c, C, i, r)$, and $f(a, r)$ are computed according to the current EMC model. $f(c, w)$, e.g., is computed as follows:

$$f(c,w) = \sum_{\langle w, a_1, \ldots, a_{n_C}, C \rangle} \frac{p(c, w, a_1, \ldots, a_{n_C}, C)}{\sum_{c'} p(c', w, a_1, \ldots, a_{n_C}, C)} \tag{12}$$

The model parameters are then reestimated based on these frequency estimates. $p(w|c)$ e.g. is computed as follows:

$$p(w|c) = \frac{f(c,w)}{\sum_{w'} f(c,w')} \tag{13}$$

The estimation of frequencies (E-step) and probabilities (M-step) is repeated until the likelihood of held-out data decreases.

The training algorithm has three parameters, namely the number of clusters c, the set of selectional restrictions r and the set of start probabilities. These parameters should be varied in order to find a good combination. The different models could be compared on the basis of the likelihood of held-out data.

4 Summary

Two methods for the lexicalization of probabilistic context-free grammars have been proposed. Both methods are modular extensions of unlexicalized models and both can be combined with a wide range of probabilistic grammars. They only require that an unlexicalized parse tree probability is defined and that the lexical governor/arguments of each word are defined and efficiently computable.

The first method defines the lexicalized probability of a parse tree as the product of its unlexicalized probability and the exponentials of the pointwise mutual information scores of all words and their governors. Governors may be arbitrarily defined. They might e.g. be words or lemmas or words plus semantic relations. Even multiple governors can be handled in an extension of the model. Mutual information scores are only explicitly represented for word-governor pairs whose frequencies differ significantly from the expected values for independent words. Therefore the number of parameters is small.

The second method is based on an EM clustering model which models the dependencies between the different arguments of a word. This model contains two hidden features, namely word classes and argument classes and requires iterative training with the EM algorithm. The model is improved by adding parameters which directly model frequent word-argument tuples which are not adequately modeled by the EMC model. This model is also useful for finding word classes, selectional restrictions and word sense probabilities.

References

Glenn Carroll and Mats Rooth. 1998. Valence induction with a head-lexicalized PCFG. In *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, Granada, Spain.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word Statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Menlo Parc.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco. Morgan Kaufmann Publishers.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, Madrid, Spain.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1993. Introduction to WordNet: An on-line lexical database. Technical report, Cognitive Science Laboratory, Princeton University.

Hermann Ney, U. Essen, and R. Kneser. 1994:. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Mats Rooth. 1998. Two-dimensional clusters in grammatical relations. In *Inducing Lexicons with the EM Algorithm*, volume 4 of *AIMS: Arbeitspapiere des Instituts für maschinelle Sprachverarbeitung*. University of Stuttgart, Germany.