Challenge Problem 5 DRAFT: Probabilistic Context-Free Grammars with Latent Annotation

Chad Scherrer, Tom Dietterich

November 24, 2014

1 Introduction

Context-free grammars (CFGs) provide a simple model for the structure of language and are widely-applied in natural language processing systems. A CFG consists of a set of non-terminal symbols N, a set of terminal symbols T, and designated start symbol S, and a set of production rules of the form $N \to R$, where R is a sequence of terminals or non-terminals. For each nonterminal $n \in N$, the set of rules having n on their left-hand-side are the "rules for n", which we will denote as $Rules(n)$. A sentence is generated by beginning with the start symbol S, choosing one of the rewrite rules in $Rules(S)$ and replace S by the right-hand-side of the rule. This is repeated recursively, each time expanding one of the non-terminals in the emerging sentence until no non-terminals remain.

Probabilistic context-free grammars (PCFGs) extend CFGs to define a probability distribution over the sentences and parse-trees generated by the grammar by specifying a separate multinomial distribution for each non-terminal n over $Rules(n)$. These probability distributions can be learned from data consisting of sentences and their parse trees ("treebanks"). The most famous treebank is the Wall Street Journal (WSJ) treebank developed at the University of Pennsylvania. It consists of 23 "sections", and it is available at no cost from the Linguistic Data Consortium (LDC) at the University of Pennsylvania.

A weakness of both CFGs and PCFGs is that each time a non-terminal is expanded using one of its rules, the choice is made independently of all other choices. This independence prevents PCFGs from capturing many important linguistic regularities. One way to address this problem is to replace very general non-terminals such as NP (noun phrase) and N (noun) with an expanded set of symbols. For example, we could have NP-animate, NP-inanimate, N-animate, and N-inanimate, to separately model noun phrases and nouns referring to animate versus inanimate objects. These are sometimes referred to as annotations, because one can imagine manually annotating each N and NP in a treebank with this additional information. However, there are no large annotated treebanks, and it is also not clear what annotations should be introduced.

One way to avoid manual annotation is to take the extreme approach known as lexicalization in which a separate "annotation" is defined for each word in the lexicon. For example, we would have N-car, N-truck, N-bus and so on. This leads to an immense grammar, and learning the parameters for the probability distributions requires introducing some form of "smoothing" so that rules for similar words are given similar probabilities.

In this challenge problem, we will investigate a different approach pioneered by Matsuzaki et al. [7] in which the annotations are "latent". That is, we could view the parse trees in the treebank as having "missing" annotations. We specify that each non-terminal can have up to k annotations, and it is the job of the learning algorithm to determine what those annotations should be and which sentences should use which annotations. In statistical modeling terms, we replace each nonterminal by a mixture of distributions over its child non-terminals (which are in turn, recursively, mixtures over their children). Recent work on this model includes Petrov et al. [9] and Cohen et al. [1].

2 Problem Overview

This problem will be given in three stages.

Stage 1: Fit a standard PCFG to a subset of the WSJ corpus.

Stage 2: Fit a latent PCFG to a subset of the WSJ corpus. The number of latent annotations will be fixed.

Stage 3: Fit a latent PCFG to a subset of the WSJ corpus. The number of latent annotations will be flexible and modeled via a Hierarchical Dirichlet Process.

Training data will consist of a text corpus (from the WSJ section of OntoNotes 5.0) with associated constituency-based parse trees (*i.e.*, one terminal associated with each nonterminal leaf node)

A good introduction to the field that only assumes general statistical modeling expertise can be found in the introduction section of Liang et al. [4].

3 Probability model

The following subsections describe iterations of the models to be expressed. Where appropriate, particular algorithms are mentioned. This is intended to be informative for the teams, but **there is no requirement to implement these particular algorithms.** The primary focus is on expressing the model in a natural way.

3.1 July 2015: Probabilistic Context-Free Grammar

The initial model considered for this challenge problem is that of a probabilistic context-free grammar (PCFG). This model is most commonly trained using the *inside-outside algorithm*. Once the production probabilities are estimated, parsing is usually performed using the *CYK algorithm*. More details on PCFGs are available from Collins [2].

3.2 January 2016: PCFG with latent annotations

The PCFG-LA model is an extension of PCFG in which each nonterminal is assumed to have a latent annotation. Thus, a particular node in the training data might be labeled as N (noun), but the model assumes this is realized as an instance of one of N_1 through N_K for some fixed K. After training the model, these latent subcategories are often seen to form clusters with some semantic interpretation. For example, N_1 might consist primarily of proper names of places, while N_2 might consist of types of animals.

Details of the model definition can be found in Matsuzaki et al. [7], and an excellent brief overview has been produced by Manning [6].

The PCFG-LA model can be trained using the EM algorithm, but this depends strongly on initialization due to local optima in the posterior density. Matsuzaki et al. handle initialization by providing annotations of the identity of the parent and sibling.

Petrov et al. [9] instead starts with a minimal grammar, and extends it with new latent subcategories using a split/merge heuristic.

Prediction

The prediction problem is to assign to a given sentence a parse tree (or distribution over parse trees, in a fully Bayesian approach). Unfortunately, as Matsuzaki et al point out, the problem of optimizing the posterior density of a PCFG-LA model is NP-hard. Matsuzaki and Petrov thus both rely on heuristics for this computation; see the respective publications for details.

3.3 July 2016: Hierarchical Dirichlet Process PCFG

The final stage for this challenge problem is a "stretch" goal. It will not be a formal requirement, but Galois will evaluate implementations that are submitted.

The Hierarchical Dirichlet Process PCFG is a version of the latent annotation model with the finite mixture of annotations replaced with a Dirichlet process. For details, see Liang et al. [3, 4].

3.4 Opportunities for Probabilistic Programming

Existing implementations of these models are very complex software systems. For example, Petrov's system requires 41240 lines of code (including comments). Probabilistic Programming languages hold the promise of making it much easier to write these programs. A second opportunity is to demonstrate that different probabilistic model components (finite mixtures, HDP mixtures) can be implemented as modular components and combined flexibly.

4 Training and test data

For training and test data, we plan to use Ontonotes 5.0, which is available at no charge from Linguistic Data Consortium [5]. From LDC's description:

> The goal of the project was to annotate a large corpus comprising various genres of text (news, conversational telephone speech, weblogs, usenet newsgroups, broadcast, talk shows) in three languages (English, Chinese, and Arabic) with structural information (syntax and predicate argument structure) and shallow semantics (word sense linked to an ontology and coreference).

The English portion of OntoNotes consists of 1.4 million words from a variety of sources (News, BN, BC, Web, Tele, and Pivot).

The directory `ontonotes-release-5.0/data/files/data/english/annotations/nw/wsj` contains directories 00 through 24. Within each of these directories you'll find (among other files) some files with names of the form `wsj_*.parse`. These `.parse` files contain the parse trees.

Petrov describes the "standard setup" for evaluation:

> We ran our experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank using the standard setup: we trained on sections 2 to 21, and we used section 1 as a validation set for tuning model hyperparameters. Section 22 was used as development set for intermediate results. All of section 23 was reserved for the final test. We used the EVALB parseval reference implementation, available from Sekine and Collins (1997), for scoring. All reported development set results are averages over four runs. For the final test we selected the grammar that performed best on the development set.

We will follow the standard setup as described by Petrov, but will train and test on smaller subsets of the data. Details of this are yet to be determined.

5 Evaluation

5.1 Quantitative evaluation

Quantitative evaluation will be in terms of the following:

1. Program size (lines of code)
2. Computational cost: execution time and memory
3. Parsing accuracy: the standard F_1 metric, as computed using the Evalb program of Sekine and Collins [10]

In the context of parsing, *precision* is defined as the proportion of predicted nodes (over all levels of the parse tree) that are also correctly parsed, and *recall* is the proportion of true nodes that are correctly parsed. The F_1 metric is then computed as the harmonic mean of precision and recall.

The Berkeley parser of Petrov [8] will be used as a gold standard. It is understood that this parser is hand-tuned code that is the result of a substantial development effort. The challenge is to approach to the performance of this state-of-the-art parser with some very small fraction of the development effort.

5.2 Qualitative evaluation

In addition to the usual considerations, a `diff` of the PCFG and PCFG-LA submissions should reveal strong similarities, with modifications in the code corresponding as closely as possible to the conceptual modification to the model.

References

[1] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. Experiments with Spectral Learning of Latent-Variable PCFGs. 2012.

[2] Michael Collins. Michael Collins web page. URL `http://www.cs.columbia.edu/~mcollins/`.

[3] Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. The Infinite PCFG using Hierarchical Dirichlet Processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–697, 2007.

[4] Percy Liang, Michael I Jordan, and Dan Klein. Probabilistic Grammars and Hierarchical Dirichlet Processes. In Tony O'Hagan and Mike West, editors, *The Handbook of Applied Bayesian Analysis*. Oxford University Press, 2009. URL `http://www.cs.berkeley.edu/~jordan/papers/liang-jordan-klein-haba.pdf`.

[5] Linguistic Data Consortium. OntoNotes Release 5.0, 2013. URL `https://catalog.ldc.upenn.edu/LDC2013T19`.

[6] Christopher Manning. Latent Variable PCFGs, 2012. URL `https://class.coursera.org/nlp/lecture/174`.

[7] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with latent annotations. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics ACL 05*, 05pages:75–82, 2005. doi: $\mathrm{doi}{:}10.3115/1219840.1219850$. URL `http://portal.acm.org/citation.cfm?doid=1219840.1219850`.

[8] Slav Petrov. berkeleyparser: A natural language parser from UC Berkeley. URL `https://code.google.com/p/berkeleyparser/`.

[9] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, number July, pages 433–440, 2006.

[10] Satoshi Sekine and Michael Collins. Evalb - Bracket scoring program, 1997. URL `http://nlp.cs.nyu.edu/evalb/`.