Probabilistic Parsing with Finite State Machines

Vassilis Pitsikalis

National Technical University of Athens, School of ECE, Zografou, Athens 15773, Greece.
vpitsik@cs.ntua.gr

1. Introduction

In this work we consider the problem of alternative parsing methods. The current standard approaches are either fully statisticall or generative. These suffer from drawbacks like the need for either huge amounts of labeled training data or expert grammar design, corespondingly. Thus the goal is to build a module that can either be used for syntactic or semantic parsing that will not require large amounts of training data, or expert-designed grammars.

The need for a robust parser is even higher since spontaneous speech may be 'far away' from speech consistent to a restricted vocabulary or may contain disfluencies and ungrammatical sentencies. Finite State Transducers have been used extensively in Natural Language Processing and Automatic Speech Recognition problems. For instance the compilation of morphological or phonological rules and representation of large scale dictionaries. They have recently been applied succesfully in parsing using context free grammars. An important reason for the succesfull and increasing use of FST's is that complex structures can be built over simpler ones since they obey on a standard set of algebraic operators.

We consider a statistical syntactic parser module that uses two different originated information sources to disambiguate between rules. These are rule weights that capture vertical relationships among different levels in the parse tree, and n-grams representing horizontal dependencies. By this way, the first model takes under consideration the hierarchical dependencies whereas the later one, provides complementary language model information to disambiguate between parses. Then the two models are combined by using appropriate weights so that the error is minimised on the held-out data. Both the rules and the n-grams are modeled as FST's. The resulting transducer is then recursively composed with a FSA that encodes all possible sequences produced by the acoustic decoder until a fixed point is reached. The corpus on which the model will be applied is the WSJ database.

2. Background

A finite-state transducer is a finite automaton whose state transitions are labeled with both input and output symbols. A path through the transducer encodes a mapping from an input symbol sequence to an output symbol sequence. By putting weights on transitions it is made fisible to encode probabilities along paths eg. search paths in a space of parse trees. The abstract representation of weighted finite-state transducers is the probability semiring. Thus algebraic operations or atributes may hold.

Basic operations (algorithms) of WFST are: composition and intersection, determinization, minimization. The composition of two transducers represents their relational composition by composing input symbols of the first transducer to output symbols of the second. Composition is used to combine different levels of representation, e.g. composition of pronunciation lexicon with word level grammar to generate a phone to word transducer whose word sequences are restricted to the grammar. A transducer is said to be deterministic if each of its states has at most one transition with each given input symbol. Determinization is the operation of making a transducer deterministic, though not every transducer can be transformed. Minimization of WFST is the procedure of creating a new equivalent transducer which has the least number of states and the least number of transitions among all WFST's. Other important operations include the projection either to the input or output to an FSA.

The derivation of a string of words after applying some rules expansions can be represented by a parse tree. For finite state automata syntactic parsing can be thought of as the procedure of searching through the space of all possible parse trees to find the correct parse tree of the input. Common generative parse methods are the top-down and bottom-up approaches. Though both have disadvantages such as generating parse trees that are either irellevant with the data, or ones that have no chance leading to a correct according to the rules tree. There are evolutions of the basic algorithms e.g. by use of methods like dynamic programming.

One of the major problems to resolve in parsing is ambiguity i.e. choosing the correct – in a predifined way – parse tree out a multitude of possible parses.

3. Probabilistic Parsing

In statistical parsing probabilities are assigned to the different parse trees, in order to disambiguate between them. The proposed model constitutes of two parts. For a given syntactic/semantic parse tree (or branch/subtree) T the standard probability by expanding each node of the tree is constructed by the product of probabilities of all the rules expanded. This model referred hereafter as the 'vertical' λ_1 model because of its conceptual and geometrical nature in the tree, captures the vertical dependencies in the parse tree ; e.g. figure 1 presents a simple parse tree in which the probability of the tree is:

$$\begin{aligned} P(T|\lambda_1) &= P(A)P(A \to BC|A)P(B \to D|B)P(C \to EF|C) \\ &= P(A)\frac{P(A \to BC)}{P(A)}\frac{P(B \to D)}{P(B)}\frac{P(C \to EF)}{P(C)} \end{aligned} \tag{1}$$

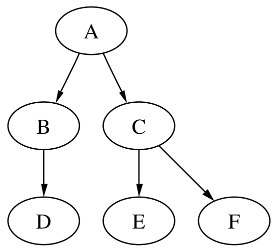


Figure 1: Simple parse tree

assumming that the rules are independent in the tree.

On the other hand another model λ_2, the 'horizontal' one shall be used to capture the relations/dependencies on each level of the parse tree accross rules. Independency accross tree levels is assumed again. The probabilities can be approximated by trigrams. The coresponding tree probability according to the λ_2 model, will be the product of the probabilities for each level:

$$\begin{aligned} P(T|\lambda_2) &= P(A)P(BC)P(DEF) \\ &= P(A)P(B)P(BC)P(D)P(E|D)P(F|DE) \end{aligned} \tag{2}$$

We consider that the different kind of dependencies (vertical & horizontal) that are captured by each model, are complementary and shall provide more information in order to disambiguate between parse trees. The tree probabilities $P(T|\lambda_1)$ and $P(T|\lambda_2)$ shall be used in a combined statistical model λ via an exponentially weighted formulae as:

$$P(T|\lambda) = P(T|\lambda_1)^{\gamma_1} P(T|\lambda_1)^{\gamma_1} \tag{3}$$

in which the weights γ_1, γ_2 can be computed under constraints of the error minimization.

4. Recursive Composition

Rules are coded as Finite State Transducers (using AT&T Bell Labs Finite State Machine library [?]). Weights were assigned according to the computed tree probabilities. Left-hand sides of the rules are mapped to the input alphabet of the FST and right-hand sides to the output alphabet. The rule transducer is produced by taking the closure of each rule transducer and the union of all. The n-gram model is coded as an FSA too, and then composed with the rule transducer to produce the final rule transducer which represents all alternative rules.

The syntactic decoding of a wordgraph (encoded as an automaton) is done by recursively composing it with the final rule transducer until a fixed point i.e convergence, is reached.

$$T_n = (W \circ^n R) = ((W \circ R) \dots) \circ R) \tag{4}$$

An easy to implement convergence criterion is the equivalence of the best paths $bestpath(T_n) = B_n$ of T_n instead of the FST's equivalence which is computationally intensive since it requires minimization of transducers too. After finding the best path of the fixed point of the algorithm, the full parse tree can be constructed recursively as:

$$T_{n-1} = (P_{n-1})_{proj-out} \bigcap (R \circ B_n)_{proj-in} \tag{5}$$

where $(T)_{proj-*}$ stands for the projection of the FST T input or output, and $\cap$ as the intersection of two acceptors.

5. Experiment – Case Study

For an input sentence drawn from the database, e.g. "THE LUXURY AUTO MAKER LAST YEAR SOLD 1,214 CARS IN THE U.S." the corresponding wordgraph automaton is shown in figure 2. Selected local grammar rules are shown before taking the union among all of them in figure 3. The first two steps of the recursive composition with the final rule transducer are shown in figures 4. Finally the resulting tree after backtracking as the union of the bestpaths at each iteration is shown in figure 5.

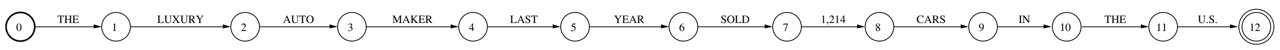


Figure 2: Word graph input automaton

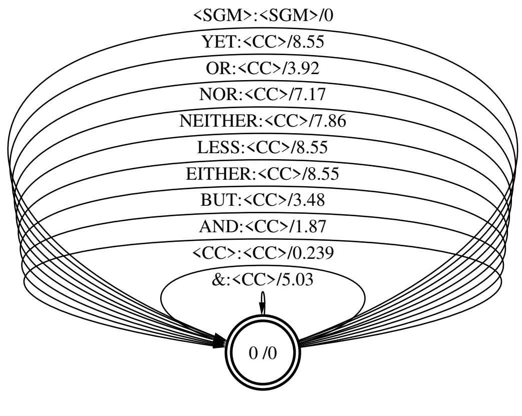
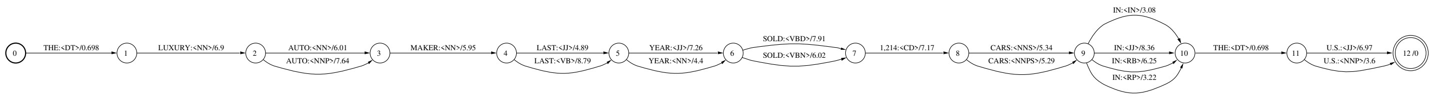
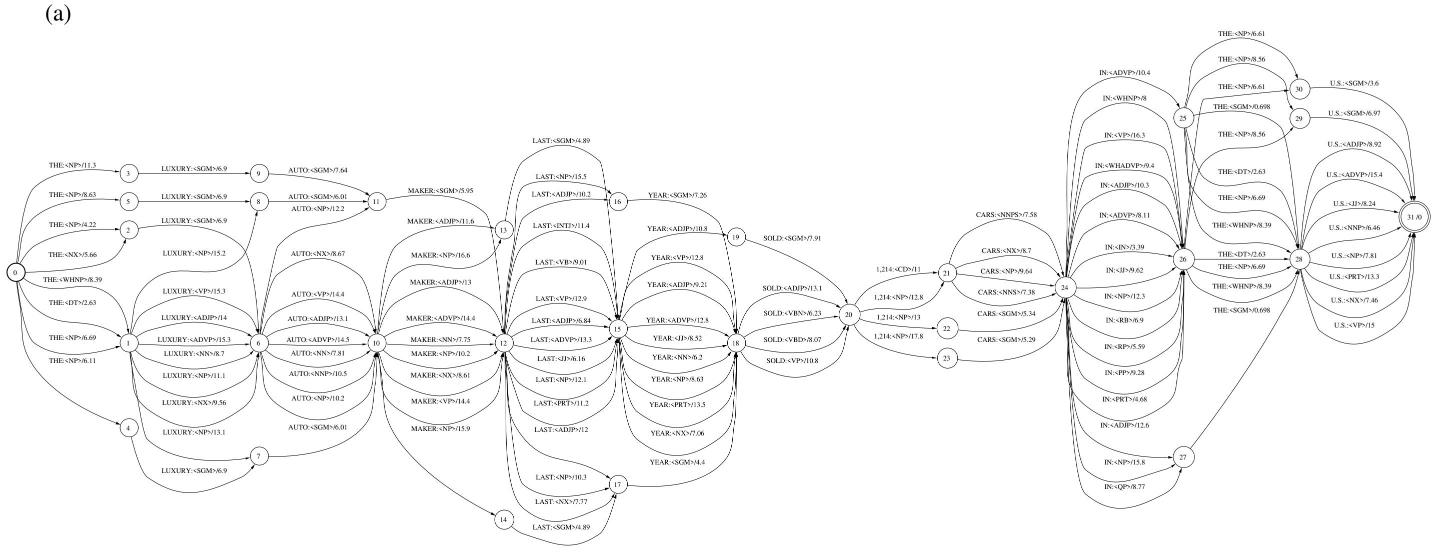


Figure 3: A simple FST that represents rules e.g. CC.



(a)



(b)

Figure 4: Recursive composition after (a) 1, (b) 2 iterations.

6. Discussion

The hybrid combination of vertical – hierarchical dependencies with n-gram language models that restricts dramatically the possible parse trees adds great robustness to the parse procedure. Furthermore the encoding of the whole grammar as a single FST adds compactness to the system and can be integrated easily with othe modules. Thought there are matters that have to be resolved like the efficiency of the system by use of more sophisticated pruning techniques.

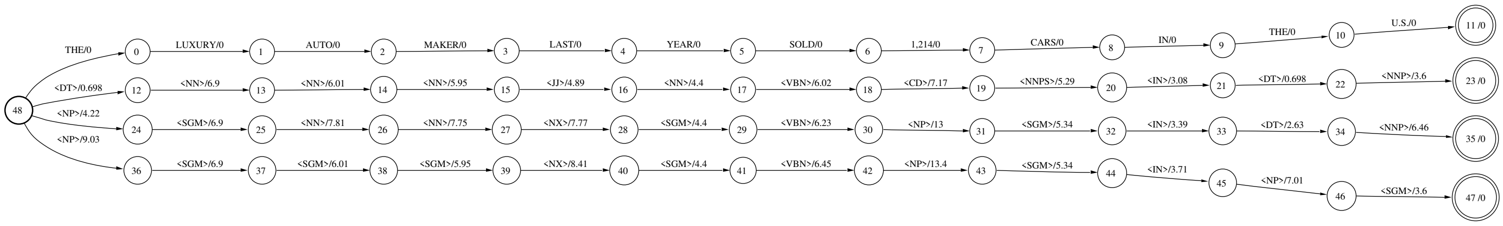


Figure 5: Union of backtracking of the best paths .

7. Acknowledgemnts

This work was based on tools and data that were provided by Alex Potamianos whom I would like to thank; especially for insightfull discussions about his work [6] which was actually extended in this short study.

8. References

[1] Collins, M., *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1999.

[2] Manning, C.D. and Schutze, H., *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[3] Jurafsky, D. and Martin, J. H. *Speech and Language Processing*, Prentice-Hall, 2000

[4] Kaiser, E. C., Johnston, M., and Heeman, P. A. *PROFER: Predictive, Robust Finite-State Parsing for Spoken Language*, in Proc. ICASSP'2000 (Istanbul, Turkey), Jun. 2000.

[5] Roche, M., *Parsing with finite state transducers*, In E. Roche and Y. Schabes, ed., *Finite–State Language Processing*, MIT Press, 1997

[6] Potamianos, A. and Kuo, H-K. J., *Statistical recursive finite state machine parsing for speech understanding*, in Proceedings of the International Conference on Spoken Language Processing, (Beijing, China), Oct. 2000.

[7] Mohri, M., *On Some Applications of Finite-State Automata Theory to Natural Language Processing*, Journal of Natural Language Engineering, 2, 1996.