UNIVERSITÀ
DI SIENA
1240

Dipartimento di ingegneria dell'informazione e scienze matematiche

Computer and Automation Engineering

Robotics and Automation

Language Processing Technologies

- lex/yacc select SQL parser
- PoS tagger stanford Library

Professor: **M. Maggini**

Student: **Francesco Vigni**

A.A. 2016-2017

Chapter 1

Lex/Yacc SQL parser

The main idea was to create a program which is able to recognize very simple select sql statements and notify whether or not the sentence provided in input is correct with sql language. Obviously we need some code to able to recognize the fixed words such as SELECT, WHERE, FROM .. etc, aka entities for lexical scanner and then we must be sure that the whole query makes sense.

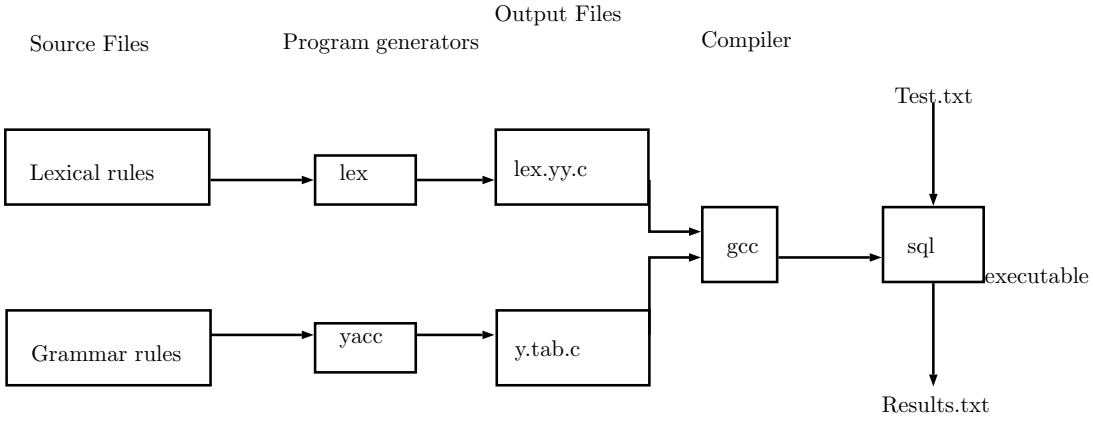


Figure 1.1: Compiling routine

The first job is done by the lexical scanner, lex. It recognize certain patterns of symbols using regular expressions, and assign them to tokens, that are available

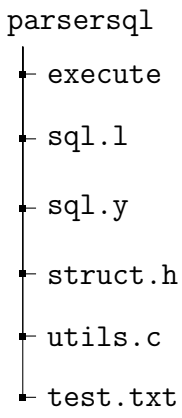
for the whole program. In this work we choose the alphanumeric RE ([a-z][A-Z0-9]*) for identifing the token IDENTIFIER, so we are able to correctly get tablenames with form (tab1, tab2, ..) or columns like col1, col2 etc.. for the other tokens we impose the same symbol pattern used by the token. The grammar task is done by yacc, in which we mainly specify the order in which we want the tokens.

1.1 Tokens

The lexical token we decide to use are: SELECT, WHERE, FROM, AND, OR, IDENTIFIER, CONST, NL. This is just an academic experiment to implement grammar and lexical rules in a real problem. The lexical analysis is done by giving lex sql.l to the command line, in doing so the lex program automatically generates a C file that follows the rules we provide in sql.l. In order to validate the sql statement, support files are required: struct.h, utils.c. These files define structure able to save in lists the columns and the tables of each statement if no error occurs. The file results.txt is populated in file utils.c and stores per each line according to the grammar, the tables and the columns. If the grammar is not satisfied it won't save that statement. In order to improve debugging a compilable file is provided under the name of: execute. it just calls the shell and execute all the commands in the right order.

1.2 Folder tree files

Here is provided the content of the folder inside the compressed file attached to this report.



Chapter 2

Part-of-Speech at Stanford

The main idea of the Part-of-Speech tagger at Stanford relies on

- explicit use of preceding and following tag context via a dependency network representation.
- use of lexical features.
- use of priors in conditional loglinear models.
- improved modeling of unknown word features.

Almost all approaches to sequence problems such as Part-of-Speech tagging take unidirectional approach to conditioning inference along the sequence, at Stanford the idea that investigating tag in both direction (L to R & R to L) came out with good results. Using dependency networks for representing preceding and following tag, along with local conditional loglinear models is a method to provide efficient bidirectional inference(Figure: 2.1).

The Part-of-Speech tagged data used in the experiments is the Wall Street Journal data from Penn Treebank III (Marcus et al., 1994), a very first result came out just considering the tag of the nearest words, the decision of assigning a tag to the word w_0 was influenced by the tags t_{-1} and t_{+1} of the words w_{-1} and

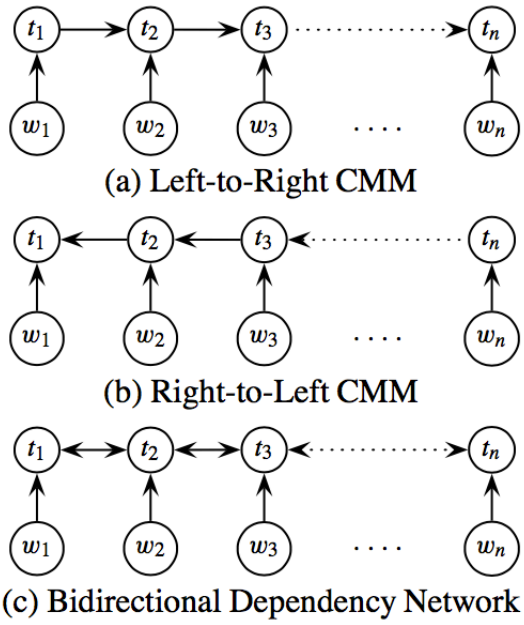


Figure 2.1: Dependency network directions

w_{+1}, the table 2.1 compress the overall result. The probability assigned to a

Model	**Features**	**Token**	**Unknown**
Baseline	56805	93,69%	82,61%
3Words	239767	96,57%	86,78%

Table 2.1: Accuracy improvement nearest tags

tagged sequence of words $x = \langle t, w \rangle$ is the product of local probabilities i.e. the bidirectional model, one from each time slice.

$$P(t, w) = \prod_i P(t_i | t_{i-1}, t_{+1}, w_i) \tag{2.1}$$

The local conditional loglinear models used for the experiment is the Conditional Markov Model (CMM) eq. 2.1 or Maximum Entropy Markov Model (MEMM),

for example we might populate a model for $P(t_0|t_{-1}, t_{+1}, w_0)$ with a maxent model of the form:

$$P_\lambda(t_0|t_{-1}, t_{+1}, w_0) = \frac{e^{\lambda_{\langle t_0, t_{-1}\rangle} + \lambda_{\langle t_0, t_{+1}\rangle} + \lambda_{\langle t_0, w_0\rangle}}}{\Sigma_{t_0'} e^{\lambda_{\langle t_0', t_{-1}\rangle} + \lambda_{\langle t_0', t_{+1}\rangle} + \lambda_{\langle t_0', w_0\rangle}}} \tag{2.2}$$

Where with the symbol $\langle t, w\rangle$ stands for current word features, and features in these models are defined using templates. The classifier makes a single decision at time, conditioned on evidence from observations and previous decisions as shown in Figure 2.2.

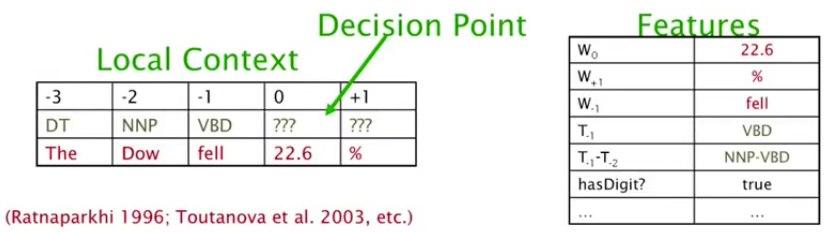


Figure 2.2: PoS tag

We have to define the score such as the product of local probabilities from a dependency network:

$$score(x) = \prod_i P(x_i|Pa(x_i)) \tag{2.3}$$

where $Pa(x_i)$ are the nodes with arcs to the node x_i, if the graph is acyclic this score will be the joint probability of the event $x, P(x)$. To summarize, we draw a dependency network in which each node has as neighbors all the other nodes that we would like to have influence it directly. Each node's neighborhood is then considered in isolation and a local model is trained to maximize the conditional likelihood over the training data of that node. At test time, the sequence with the highest product of local conditional scores is calculated and returned. We can always find the exact maximizing sequence, but only in the case of an acyclic net it is guaranteed to be the maximum likelihood sequence.

A large discussed problem of PoS taggers is the modeling of unknown words, many tools can be used in order to avoid assigning zero probability to words that can eventually appears in the corpus. At Stanford the smoothing model uses a regularized objective F:

$$F(\lambda) = \sum_{i} log(P_\lambda(t_i|w,t)) + \sum_{j=1}^{n} \frac{\lambda_j^2}{2\sigma^2} \tag{2.4}$$

aka quadratic regularization which resists high feature weights unless they produce great score gain. Empirical experiments shows that the parameter σ is optimized when $\sigma^2 = 0.5$, the above mentioned quadratic regularization is very effective in improving the generalization performance of tagging model. The Maximum Entropy (MaxEnt) tagger presented combines the advantages of both *Statistical Decision Tree* and Brill's *Transformation Based Learner*, adding rich feature representation (bidirectional dependency network), by it self changing from generative to discriminative model does not give a big boost to PoS tagging performance, the big boost comes from being able to put a lots of features to observation and combining them together in a new way. The higher accuracy of discriminative models come at the price of much slower training. The key advantage of MEMMs is that the use of feature vectors allows much richer representations than those used in HMMs. For example, the transition probability can be sensitive to any word in the input sequence. In addition, it is very easy to introduce features that are sensitive to spelling features (e.g., prefixes or suffixes) of the current word w_i, or of the surrounding words. These features are useful in many NLP applications, and are difficult to incorporate within HMMs in a clean way.

Conclusion

This project is for educational purposes only it spans from a very basic implementation of the select sql query to the analysis of the work done at Stanford[1]. The PoS tagger shows very good results, in particular configurations such as taking in consideration the tag of the previous and of the next word gives 97,24 % of token accuracy. We must remember that in PoS tagging the baseline rule: assign the most frequent tag to its word and *noun*[2] to unknown words is already giving 90% of performance accuracy so gaining few decimals is important since this task optimization can move to an upperbound of around 98% which seems to be the maximum human PoS tagging performance.

[1] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network

[2] each tag refers to Penn Treebank set