A SYNTACTIC LANGUAGE MODEL BASED ON INCREMENTAL CCG PARSING

Hany Hassan *Khalil Sima'an** *Andy Way*

School of Computing
Dublin City University
Dublin 9, Ireland

*Language and Computations
Universiteit van Amsterdam
Amsterdam, Netherlands

ABSTRACT

Syntactically-enriched language models (parsers) constitute a promising component in applications such as machine translation and speech-recognition. To maintain a useful level of accuracy, existing parsers are non-incremental and must span a combinatorially growing space of possible structures as every input word is processed. This prohibits their incorporation into standard linear-time decoders. In this paper, we present an incremental, linear-time dependency parser based on Combinatory Categorial Grammar (CCG) and classification techniques. We devise a deterministic transform of CCGbank canonical derivations into incremental ones, and train our parser on this data. We discover that a cascaded, incremental version provides an appealing balance between efficiency and accuracy.

Index Terms— Natural languages, Language Modeling, Grammar

1. INTRODUCTION

As it processes an input sentence left-to-right (for a language like English), word-by-word, an *incremental* parser builds for each prefix of the input sentence a partial parse that is a subgraph of the partial parse that it builds for a longer prefix. Incremental parsers may have access only to a fixed, limited window of lookahead words.[1] Besides being cognitively plausible, an incremental parser is more appealing for applications if its time and space (worst-case) complexities are linear in input length. An incremental, linear-time parser should constitute a natural match for the word-by-word decoding and pruning schemes used within phrase-based statistical machine translation and speech recognition.

Combinatory Categorial Grammar (CCG) [1] is a theory that assumes a lexicalized grammar consisting of a lexicon and a small set of combinatory operators. The operators assemble lexical entries together into parse-trees. The lexical entries consist of syntactic constructs (called 'supertags') that describe such lexical information as the POS tag of the word, its subcategorization information and the hierarchy of phrase categories that the word may project upwards in the parse-tree.

[1]In other words, an incremental parser may not delay decisions indefinitely.

The CCGbank [2] was obtained by transforming the parse trees in the Penn Wall Street Journal (WSJ) Treebank into normal form CCG derivations, exhibiting head-dependency annotations over a wide-coverage lexicon of supertags. The CCGbank has been used to train wide-coverage CCG parsers, e.g. [3]. These parsers assign supertags to the words in an input sentence based on the probability of a word–supertag pair given their local context [4], and assemble these supertags together into parse trees by deciding on the suitable combinatory operators. Due to ambiguity regarding the choice and order of operators over the sequence of supertags, existing CCG parsers remain cubic-time (worst-case complexity) and non-incremental.

In this paper we present the design of a linear-time parser that builds exactly a single parse incrementaly, as it processes every input word from left to right. This parser is obtained by transforming CCGbank derivations into incremental ones, that exhibit the incremental parse decisions and states, and then training a classifier to make such incremental decisions for a novel input. We study different kinds of architectures for incremental parsing and various sets of classification features and arrive at a incremental linear time, parser that exhibits useful accuracy levels.

2. RELATED WORK

A number of researchers have introduced approaches that incorporate syntactic information into language models. The Structured Language Model [5] constitutes an incremental shift-reduce parser which conditions the probability of words on previous lexical heads, rather than previous words as in *n*-gram language models. The probability of the word is the weighted sum of its conditional probabilities from possible parses. [6] proposed an incremental top-down and left-corner parsing that generates conditional word probabilities. He deployed parse probabilities directly to calculate the string probabilities. Charniak [7] proposes a head-driven parsing approach that directly uses generative Probabilistic Context-Free Grammar (PCFG) models as language models which made use of a non-incremental, head-driven statistical parser to produce string probabilities. [8] proposes a language model based on a constraint dependency grammar and a tagger derived from that grammar to exploit syntactic dependencies.

All previous approaches depend on non-deterministic techniques to grow a huge number of partial derivations which is unmanageable for large-scale applications such as MT or large-scale speech recognition. This has limited the usability of these approaches to very small tasks and/or re-ranking of systems outputs. Another major aspect is that the previous approaches deploy PCFG techniques, that cannot handle non-constituent constructions.

3. DEPENDENCY LANGUAGE MODEL

We present a dynamic model of syntax construction based on the theoretical concepts proposed in [9]. In our model the syntactic process is represented by a sequence of transitions between adjacent syntactic states. The syntactic representation is built step-by-step from left-to-right while traversing the input string as shown in (1). The syntactic state is supposed to summarize all the syntactic information about fragments that have already been processed so far. The parser produces fully connected intermediate structures while moving from one word to the next.

We devise an incremental parser based on CCG as the grammatical representation of the syntactic states and the transition actions that lead from a state to another. Following the approach taken in (1), each word w_i is associated with a lexical syntactic/semantic descriptor st_i. At each transition, a parsing action o_i is associated with that transition, which transforms the current parsing state S_i to the next state S_{i+1} which, in turn, represents a new partial syntactic derivation. When the last word is encountered, a final state S_n represents the final syntactic structure for the given sequence of words. Such a sequence of parsing actions constructs the parsing derivation step-by-step.

$$S_0 \xrightarrow[w_1,st_1]{o_1} S_1 \xrightarrow[w_2,st_2]{o_2} S_2 \cdots\cdots S_i \xrightarrow[w_i,st_i]{o_i} S_{i+1} \cdots\cdots S_n \qquad (1)$$

In our language model, which employs CCG as grammatical representation: the lexical descriptor st_i is represented by a CCG supertag, the parsing action o_i is represented by a CCG Combinatory Operator with the state S_i being a composite CCG category.

The probability $P(W, S)$ of a word sequence $W = w_1^n$ and associated final parse state sequence $S = s_1^n$, which represents a possible derivation, can be described as in Eqn 2:

$$\begin{aligned} P(W,S) &= \prod_{i=1}^{n} P(w_i|W_{i-1}S_{i-1}) \\ &\quad P(st_i|W_i)P(o_i|W_i,S_{i-1},ST_i) \end{aligned} \qquad (2)$$

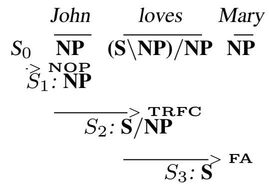


Fig. 1. *A sentence and possible supertag-, operator- and state-sequences. NOP: No Operation; BC: Backward Composition; FA: Forward Application.*

- $P(w_i|W_{i-1}S_{i-1})$ is the probability of w_i given the previous sequence of words W_{i-1} and the previous sequence of states S_{i-1}.
- $P(st_i|W_i)$: is the lexical descriptor (supertag st_i) probability given the word sequence W_i up to the current position. This is represented by a sequence tagger (supertagger) in our CCG incremental parser.
- $P(o_i|W_i, S_{i-1}, ST_i)$ represents the parsing action (operator o_i) probability given the previous words, supertags and state sequences up to the current position. This is represented by a sequence operator tagger in our CCG incremental parser.

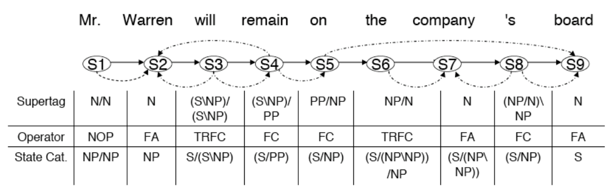
It is worth noting that the proposed language model is deterministic, in the sense that it maintains a single parsing state, that represents the possible parsing decision at each word position. This characteristic is very important for incorporating our dependency language model into large-scale MT and speech recognition systems. In the remainder of this paper, we will discuss the development and evaluation of this incremental parser based on CCG.

4. LINEAR-TIME, INCREMENTAL CCG PARSING

As it processes the sentence left-to-right, word-by-word, our parser specifies for every word a supertag and a combinatory operator, and maintains a parse-state (henceforth 'state'). Each state is represented by a composite CCG category. This composite CCG category is the result of applying the combinatory operator to the preceding state and current supertag. In terms of CCG representations, a CCG composite category specifies a functor and the arguments that are expected to the right of the current word.

Crucially, given a sentence and its state sequence, the dependency structure can be retrieved unambiguously. At each state the partial dependency structure can be represented as a directed graph with nodes representing words and arcs representing dependency relations. Figure 1 illustrates the workings of this incremental parser.

Our parser consists of two parts: (i) a Supertag-Operator Tagger which proposes a supertag–operator pair for the current word, and (ii) a State-Realizer, which realizes the cur-



	Mr.	Warren	will	remain	on	the	company	's	board
	S1	S2	S3	S4	S5	S6	S7	S8	S9
Supertag	N/N	N	(S\NP)/(S\NP)	(S\NP)/PP	PP/NP	NP/N	N	(NP/N)\NP	N
Operator	NOP	FA	TRFC	FC	FC	TRFC	FA	FC	FA
State Cat.	NP/NP	NP	S/(S\NP)	(S/PP)	(S/NP)	(S/(NP\NP))/NP	(S/(NP\NP))	(S/NP)	S

Fig. 2. Illustration of the CCGbank transformation process into incremental derivations.

rent state and dependencies by applying the current operator to the previous state and the current supertag. In this work, the State-Realizer is a deterministic function, whereas the supertag-operator tagger is a statistical one trained on our own incremental version of the CCGbank. While this conceptual view describes a baseline, fully/no-lookahead incremental version, we will trade off some aspects of this architecture for accuracy, by employing lookahead in predicting supertag–operator pairs.

To train the statistical components, we transform the CCGbank normal form derivations into strictly left-to-right derivations, with operators specifically chosen to allow incrementality while satisfying the dependencies in the CCGbank. In the next section we briefly describe our transformation technique developed to obtain the appropriate training data.

4.1. Obtaining left-to-right derivations from CCGbank

The goal of the transformation is to obtain training data for our incremental parsing approach. The result of the transform is an incremental CCGbank where sentences are annotated with supertags as well as combinatory operators that allow left-to-right, incremental building of a parse while satisfying the dependencies specified in the CCGbank. Figure 2 illustrates the transformation process, step-by-step, on a sentence of the CCGbank. At the beginning of the process, we start with the words, the associated supertags and the dependency relations, indicated by curved dotted arrows in the figure. The purpose of the transformation process is to induce the state sequence and the operator sequence. These sequences should be able to reproduce the given dependency relations.

The same procedure applies during parsing, i.e. if we have the supertag and the operator sequences, then we are able to construct both the incremental states and the dependency graph. The details of this linguistic transformation process is beyond the scope of this short paper.

4.2. Implementation Detail

After POS tagging, the parser works its way through the sentence, left-to-right, assigning for every word a supertag–operator pair, and deciding on a state using the deterministic state-realizer. We describe the state-realizer before delving

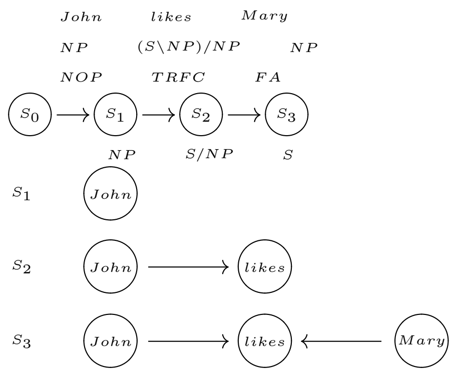


Fig. 3. Illustration of the operation of the incremental parse-state realizer and the associated intermediate dependency graphs at each state.

into the implementation of different versions of the supertag-operator tagger.

Parse-State Realizer: After assigning supertag–operator pairs for the words of the input sentence (described in the next section), the *state-realizer* deterministically realizes the parse-states as well as the intermediate dependency graphs between words using the CCG incremental operators (as defined in our incremental version of the CCGbank). Figure 3 illustrates the realizer operation along with the incrementally constructed partial dependency graphs at each state.

Supertag-Operator Taggers: We build different linear-time models for assigning supertag–operator pairs to words in order to explore the effect of the different gradations of incrementality on parsing accuracy. All models present in this paper are based on MaxEnt classifiers [10]. A MaxEnt classifier selects the class that gives the highest conditional probability of any class given a set of features of the input, where the probability is expressed as a log-linear interpolation of weights of features. The weights are trained in order to maximize the likelihood of the given training data.

5. EXPERIMENTS AND RESULTS

This section details a number of experiments carried out to test the effectiveness of the supertagger, the operator tagger, and our ability to capture the necessary dependencies using a range of incremental parsers. We used the same data split as in [3]. Sections 02–21 were used for training, section 00 for dev-testing of intermediate taggers, and section 23 for testing dependencies.

Given our introduction of new supertags for coordination, apposition, interruption, and WH-movement, we used section 00 to evaluate our supertagger's accuracy compared to the standard CCGbank set. Although our supertags are more complex, we obtain an F-score of 91.7 which compares favourably with the supertagger of [3].

Architecture	Dependency Accuracy	Supertagging Accuracy	Operator Accuracy
Joint	83.20	85.02	
Cascade	86.70	91.70	90.90
No look ahead	59.01	68.11	76.19

Table 1. Accuracy Results for Joint and Cascaded systems

In Table 1 we also present the results for our Operator tagger. This displays a very high accuracy of 90.9%. We also present the results for unlabelled dependency accuracy using our method. We use the same evaluation criteria as [3] by comparing the dependency output of the incremental parser with the predicate-argument dependencies in the CCGbank. Testing on section 23 of the WSJ, we obtain an F-score of 86.7. The score with the gold standard POS in the input is 87.5. Clearly, this result is considerably below the result reported in [3] (91.65% unlabelled dependency F-score), who use a non-incremental, chart parser (qubic-time in input length). While we aim for maximum accuracy, we accept the fact that a linear-time, incremental parser will generally perform less well than a qubic-time non-incremental parser. More relevant to language modeling, [3] observe that on section 23 of the WSJ, their parser takes 1.9 minutes. By contrast, our parser takes just 11 seconds, a speed-up of around ten times, on the same specification machine.

The results reported above demonstrate the accuracy of the cascaded approach using two cascaded taggers: the first for supertags, and the second the operator tagger followed by the deterministic state-realizer. We compared the cascaded model with a joint model, where we train a single classifier that produces the supertags and operators simultaneously in the same step. In Table 1 we give the unlabeled dependency results for section 23 for the cascaded and joint models side-by-side for comparative purposes. The significantly inferior result of the joint model exemplifies the hardness of CCG incremental parsing.

The present parser is just two words of lookahead away from being fully incremental. We also examined the effect of lookahead features on the supertagger, operator tagger and dependency results. As shown in Table 1, huge improvements are to be seen when the parser avails of lookahead. Clearly, full incrementality at this stage comes at a high cost in accuracy.

6. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a novel dependency language model based on wide-coverage CCG incremental parser. Our empirical results show useful dependency accuracy even when compared with bottom-up parsing while obtaining far more efficient incremental parsing. This speedup is, we feel, particularly attractive for applications that incorporate in the decoder a word-prediction (or language) model, since this semi-incremental parser works in a fashion similar to such language models, i.e. the possible states are built on-the-fly from the training data, just like any other non-parametric method. Work on an improved dependency accuracy is ongoing as well as an investigation of the effect of using the proposed language model in machine translation.

7. ACKNOWLEDGEMENTS

We would like to thank Salim Roukos for fruitful discussions. This work is partially funded by SFI Award 05/IN/1732.

8. REFERENCES

[1] Mark Steedman, *The Syntactic Process*, MIT Press, Cambridge, MA, 2000.

[2] Julia Hockenmaier and Mark Steedman, “Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank,” *Computational Linguistics*, vol. **33**, no. 4, pp. 355–396, 2007.

[3] S. Clark and J. Curran, “Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models,” *Computational Linguistics*, vol. **33**, no. 1, pp. 439–552, 2007.

[4] S. Bangalore and A. Joshi, “Supertagging: An Approach to Almost Parsing,” *Computational Linguistics*, vol. **25**, no. 2, pp. 237–265, 1999.

[5] C. Chelba, *Exploiting Syntactic Structure for Natural Language Modeling*, Ph.D. thesis, Johns Hopkins University, Baltimore, MD, 2000.

[6] B. Roark, “Probabilistic top-down parsing and language modeling,” *Computational Linguistics*, vol. **27**, no. 2, pp. 249–276, 2001.

[7] E. Charniak, “Immediate-head parsing for language models,” in *39th Meeting of the Association for Computational Linguistics - (ACL’01)*, Toulouse, France, 2001, pp. 124–131.

[8] W. Wang, Andreas Stolcke, and M. Harper, “The use of a linguistically motivated language model in conversational speech recognition,” in *Proceedings of Acoustics, Speech, and Signal Processing(ICASSP)*, Montreal, Canada, 2004.

[9] D. Milward, “Dynamic Dependency Grammar,” *Linguistics and Philosophy*, vol. **17**, pp. 561–605, 1994.

[10] Adam Berger, Vincent Della-Pietra, and Stephen Della-Pietra, “A Maximum Entropy approach to natural language processing,” *Computational Linguistics*, vol. **22**, no. 1, pp. 39–71, 1996.