Natural Language Engineering **6** (2): 113–138. Printed in the United Kingdom

© 2000 Cambridge University Press

A lightweight dependency analyzer for partial parsing

BANGALORE SRINIVAS
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932, USA

(*Received 5 November 1999*)

Abstract

In this paper, we present a novel approach to partial parsing that produces dependency links between words of a sentence. The partial parser called a *lightweight dependency analyzer* uses information encoded in supertags and hence can produce constituency-based as well as dependency-based analyses. The lightweight dependency analyzer has been used for text chunking, including noun and verb group chunking. We also present a proposal for a general framework for parser evaluation that is applicable for evaluating both constituency-based and dependency-based, partial and complete parsers. The performance results of the lightweight dependency analyzer on *Wall Street Journal* and Brown corpus using the proposed evaluation metrics are discussed.

1 Introduction

Robust parsing is crucial for applications that exploit the syntactic structure of language. Applications such as Machine Translation have long relied on the syntactic information of the source and target languages. The use of syntactic information has also made way into applications such as Information retrieval and extraction (Strzalkowski, Lin, Perez-Carballo and Wang 1997) and for language modeling (Lafferty, Sleator and Temperley 1992; Chelba and Jelinek 1998) in speech processing. There have been two main paradigms to robust parsing: finite-state grammar and statistical approaches.

Finite State Grammar-based approaches to parsing, exemplified by several authors (Joshi 1960; Abney 1990; Appelt, Hobbs, Bear, Israel and Tyson 1993; Roche 1993; Grishman 1995; Hobbs, Appelt, Bear *et al.* 1997; Joshi and Hopely 1997; Karttunen, Chanod, Grefenstette and Schiller 1997), use hand-built grammars represented as cascaded finite-state regular expression recognizers, with each recognizer providing locally optimal output. The output of these systems is mostly in the form of noun groups and verb groups with no clausal-level or modifier attachments, often called a *shallow parse*. These parsers always produce one output, since they use the longest match heuristic to resolve cases of ambiguity when more than one regular expression matches the input string at a given position. At present, none of these systems use any statistical information to resolve ambiguity.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Pioneered by the IBM natural language group (Fujisaki, Cocke, Black and Nishino 1989) and later pursued by others (Schabes, Roth and Osborne 1993; Jelinek, Lafferty, Magerman *et al.* 1994; Magerman 1995; Collins 1996; Collins 1997; Ratnaparkhi 1997), the statistical approach decouples the issue of well-formedness of an input string from the problem of assigning a structure to it. The rules to assign a structure to an input are extracted automatically from hand-annotated parses of large corpora, which are then subjected to smoothing to obtain reasonable coverage of the language. The resultant set of rules is not linguistically transparent, and are not easily modifiable. Lexical and structural ambiguity is resolved using probability information that is encoded in the rules. This allows the system to assign the most-likely structure to each input. The output of these systems consists of constituent analysis, the degree of detail of which is dependent on the detail of annotation present in the treebank that is used to train the system.

We have presented (Joshi and Srinivas 1994; Srinivas 1997a; Bangalore and Joshi 1999) an alternate robust parsing approach called *supertagging* that integrates the flexibility of linguistically motivated lexical descriptions with the robustness of statistical techniques. The idea underlying the approach is that the computation of linguistic structure can be localized if lexical items are associated with rich descriptions (*supertags*) that impose complex constraints in a local context. This makes the number of different descriptions for each lexical item much larger, than when the descriptions are less complex; thus increasing the local ambiguity for a parser. However, this local ambiguity can be resolved by using statistical distributions of supertag co-occurrences collected from a corpus of parses. Supertag disambiguation results in a representation that is effectively a parse (*almost parse*). In this paper, we show how the output of the supertagger can be combined to produce a dependency analysis.

The layout of the paper is as follows. In section 2, we briefly describe the task of supertagging and illustrate it with an example in section 3. Some of the recent work on supertagging is presented in section 4. In section 5, we illustrate the use of the supertagged output for dependency analysis. Issues concerning the applicability of existing metrics for evaluation of partial parsers are discussed in section 6, and our proposal for a general framework for parser evaluation is presented in section 7. In section 8, we evaluate the supertagger and LDA system for the purpose of text chunking and partial parsing. The supertagger and LDA system have been used in a number of applications, which are mentioned in section 9.

2 Supertagging

Part-of-speech disambiguation techniques (POS taggers) (Church 1988; Weischedel, Schwartz, Palmucci, Meteer and Ramshaw 1993; Brill 1993) are often used prior to parsing to eliminate (or substantially reduce) the part-of-speech ambiguity. The POS taggers are all local in the sense that they use information from a limited context in deciding which tag(s) to choose for each word. As is well known, these taggers are quite successful.

In a lexicalized grammar such as the Lexicalized Tree-Adjoining Grammar

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

(LTAG), each lexical item is associated with at least one elementary structure (tree). The elementary structures of LTAG localize dependencies, including long distance dependencies, by requiring that all and only the dependent elements be present within the same structure. As a result of this localization, a lexical item may be (and, in general, almost always is) associated with more than one elementary structure. We will call these elementary structures *supertags*, in order to distinguish them from the standard part-of-speech tags. Note that even when a word has a unique standard part-of-speech, say a verb (V), there will usually be more than one supertag associated with this word. Since there is only one supertag for each word (assuming there is no global ambiguity) when the parse is complete, an LTAG parser (Schabes, Abeillé and Joshi 1988) needs to search a large space of supertags to select the right one for each word before combining them for the parse of a sentence.

Since LTAGs are lexicalized, we are presented with a novel opportunity to eliminate or substantially reduce the supertag assignment ambiguity by using local information such as local lexical dependencies, prior to parsing. As in standard part-of-speech disambiguation, we can use local statistical information in the form of N-gram models based on the distribution of supertags in a LTAG parsed corpus. Moreover, since the supertags encode dependency information, we can also use information about the distribution of distances between a given supertag and its dependent supertags.

Note that as in standard part-of-speech disambiguation, supertag disambiguation could have been done by a parser. However, carrying out part-of-speech disambiguation prior to parsing makes the job of the parser much easier and therefore speeds it up. Supertag disambiguation reduces the work of the parser even further. After supertag disambiguation, we would have effectively completed the parse and the parser need 'only' combine the individual structures; hence the term *almost parsing*. This method can also be used to parse sentence fragments and in cases where the supertag sequence after disambiguation may not combine into a single structure.

3 Example of supertagging

LTAGs associate with each lexical item, one elementary tree for each syntactic environment that the lexical item may appear in. As a result, each lexical item is invariably associated with more than one elementary tree. We call the elementary structures associated with each lexical item super parts-of-speech (super POS) or **supertags**[1]. A supertag is said to be anchored by the word associated with it. Figure 1 illustrates a few elementary trees associated with each word of the sentence, *market professionals said London has several attractions.*

Supertags are of two types: initial supertags (αs) and auxiliary supertags (βs). Initial supertags represent non-recursive constructs and auxiliary supertags represent recursive constructs in a language. The non-terminals appearing as leaves of an initial supertags are marked for substitution($\downarrow$). An auxiliary supertag contains a distinctive

[1] For the purpose of this paper, we suppress the features associated with the supertags.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

non-terminal as a leaf node which is marked as *foot node* (∗). In addition to the foot node, an auxiliary supertag may contain non-terminals that are marked for substitution. Typically, substitution nodes are filled in by complements of the anchor, while a foot node is filled in by a modifee that the anchor modifies.

The example in Figure 2 illustrates the initial set of supertags assigned to each word of the sentence *market professionals said London has several attractions*. The order of the supertags for each lexical item in the example is not relevant. Figure 2 also shows the final supertag sequence assigned by the supertagger which picks the best supertag sequence using statistical information about individual supertags and their dependencies on other supertags.

We have previously presented a trigram-based as well as a dependency-based supertag disambiguation model (Srinivas 1997b; Bangalore and Joshi 1999). Trained on one million words of *Wall Street Journal* data and tested on 50K words, the trigram model assigns the correct supertag to 92.2% of words.

4 Recent advances in supertagging

In recent work (Chen, Srinivas and Vijay-Shanker 1999), a variety of novel models have been explored that improve the performance of supertagging. These models can be categorized as contextual models and class-based models. The contextual models improve supertag performance by combining a variety of features from the context and exploiting the distinction between initial and auxiliary supertags. These contextual models have been combined using voting strategies to provide an error reduction of 9.5% over trigram models. In contrast to contextual models, class-based models improve supertagging performance by assigning words with sets of supertags, thus trading supertag ambiguity for improved accuracy. Different classes of supertags have been derived based on structural properties and also based on analysis of confusion pairs produced by the trigram model. With a slight increase in supertag ambiguity, 1.3 supertags per word, an error reduction of 45% has been achieved over the trigram model and with 3.8 supertags per word, the error reduction is 65% over a trigram model.

We have also implemented a weighted finite-state transducer version of an n-gram model for supertagging. The construction of such a transducer follows naturally from the n-gram model which is expressed as that sequence of supertags that maximizes the product of emit probability and the contextual probability as shown in Equation (1):

$$Pr(T) = argmax_T \prod Pr(w_i \mid t_i) * Pr(t_i \mid t_{i-1} \ldots t_{i-n+1}) \quad (1)$$

For the purpose of finite-state transducer implementation, the part of the emit probability model is represented as a weighted lexicon (`LEX`) as shown in Figure 3. The lexicon transducer, `LEX`, consists of one state which is both the initial and final state and each arc is a self loop on the initial state. A word and its supertag are represented as the input and output symbols of an arc. The probability, $Pr(w_i \mid t_i)$, is represented as the weight of the arc.

The contextual probability model is represented by a finite-state acceptor (`CONTEXT`).

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

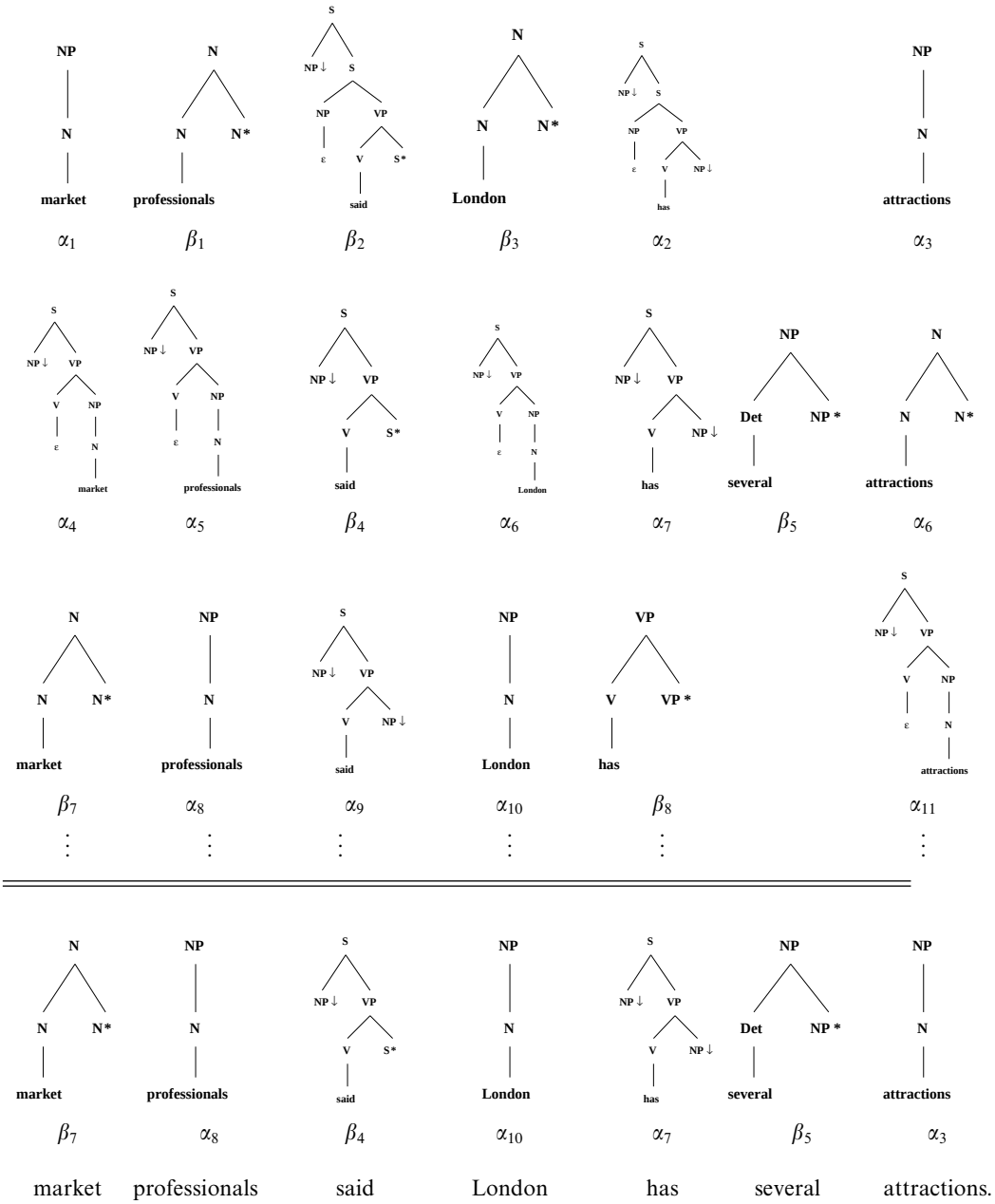


Fig. 1. A selection of the supertags associated with each word of the sentence *market professionals said London has several attractions.*

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Sent:	market	professionals	said	London	has	several	attractions.
Initial	α_1	β_1	β_2	β_3	α_2		α_3
Assig.	α_4	α_5	β_4	α_6	α_7	β_5	β_6
	β_7	α_8	α_9	α_{10}	β_8		α_{11}
	⋮	⋮	⋮	⋮	⋮		⋮
Final Assig.	β_7	α_8	β_4	α_{10}	α_7	β_5	α_3

Fig. 2. Supertag disambiguation for the sentence *market professionals said London has several attractions.*

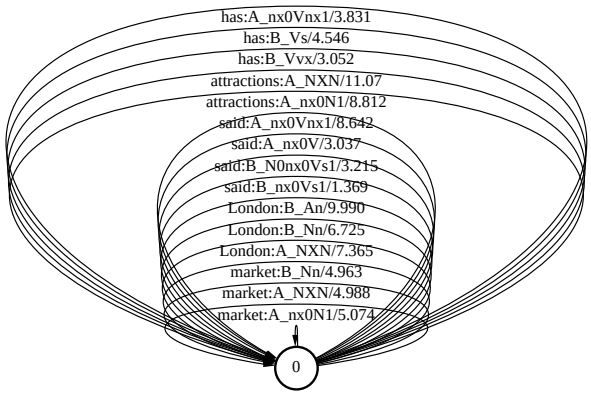


Fig. 3. Sample entries from LEX. A word and its supertag are represented as input-output pairs of an arc and the associated (minus logarithm) probability is represented as the weight on the arc.

Each n-gram conditioning context is represented as a state in this FSM. The possible supertags appearing in a given context are represented as arcs starting from the state representing that context. The conditional probability values are represented as weights on the arcs. Furthermore, a back-off model for unseen contexts is also included in the FSM. The back-off model contributes additional arcs and states for less specific contexts as shown in Figure 4 for a trigram contextual model. The resulting FSM can be determinized and minimized for purposes of efficiency.

The overall finite-state transducer implementation of the n-gram supertagger is shown in Equation 2. Given a sentence (`SENTENCE`) which is represented as an FSM, we compose it with the `LEX` transducer and the `CONTEXT` transducer. Then we search for the best path (most probable path) through the resulting weighted transducer. The sequence of output symbols associated with the best path correspond to the supertag sequence for the input sentence.

(2) $$T = \texttt{Output(}\ \texttt{BestPath}\ \texttt{(SENTENCE o LEX o CONTEXT))}$$

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

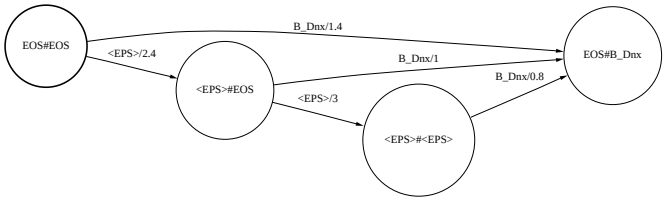


Fig. 4. Representation of a trigram with its back-off parameters represented as a finite-state machine. Supertags are labels on the arcs and the n-gram contextual (minus logarithm) probability is represented as the weight on the arc. ⟨EPS⟩ is the empty symbol.

We have implemented the finite-state transducer model of supertagging using the FSM toolkit available from AT&T Labs-Research (Mohri, Pereira and Riley 1998) (`http://www.research.att.com/sw/tools/fsm`). The resulting `LEXICON` transducer has one state and 128,000 arcs while the `CONTEXTUAL` transducer has 5,417 states and 42,511 arcs. The supertagger tags at 50 words per second.

Notice that the above method can also be applied when the input is a lattice instead of a sentence. We intend to use this flexibility to supertag word lattices output from a speech recognizer.

5 Lightweight dependency analyzer

Supertagging associates each word of a sentence with a unique supertag. The sequence of supertags selected by the supertagger can be provided as input to an LTAG parser (Schabes 1990) which combines them using substitution and adjunction operations to produce two structures called the derived tree and the derivation tree. The derived tree shown, in Figure 6, is a phrase structure tree with constituency information. The derivation tree, shown in Figure 7, records the process in which the supertags are combined to produce the phrase structure tree. The nodes of the derivation tree are the supertags that a word is assigned and the links connecting the nodes represent substitution or adjunction operations. The derivation tree can also be interpreted as a dependency tree with a parent node representing a head word and its children nodes representing its dependents. Also, substitution links represent complementation and adjunction links represent modification.

Instead of employing an LTAG parser to establish dependency links among words of a sentence, we choose to do so directly using the dependency requirements encoded in the supertags. As mentioned earlier, substitution nodes and foot nodes in supertags serve as slots that must be filled by the arguments of the anchor of the supertag. A substitution node of a supertag is filled by the complements of the anchor while the foot node of a modifier supertag is filled by a word that is being modified by the supertag. These argument nodes have a polarity value reflecting their positional orientation with respect to the anchor of the supertag. Also associated with a supertag is a list of internal nodes (including the root node) that appear in the supertag. Encoding of a few supertags in this scheme is shown in

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Supertag	Root category	Anchor category	Internal nodes	External nodes
β_1 (noun modifier)	N	N	N	+N*
β_2 (Subject Wh) (Sentence Complement)	S	V	S NP VP	−NP. +S*
β_4 (sentence complement)	S	V	S VP	−NP. +S*
β_5 (determiner)	NP	D	NP	+NP*
β_8 (auxiliary verb)	VP	V	VP	+VP*
α_1 (head noun)	NP	N	NP	–
α_2 (transitive subject Wh)	S	V	S VP NP	−NP. +NP.
α_4 (Predicative Noun)	S	N	S VP NP	−NP.
α_7 (transitive verb)	S	V	S VP	−NP. +NP.

Fig. 5. Illustration of the encoding of a few supertags used by the lightweight dependency analyzer. This also the list of supertags that will be referred to in examples in this paper.

Figure 5[2]. A dot indicates a complement while an asterisk indicates a modifee. Using the structural information coupled with the argument requirements of a supertag, a simple algorithm such as the one shown in Figure 8 provides a method for annotating the sentence with dependency links.

An example illustrating the output from this algorithm is shown in Table 1. The first column lists the word positions in the input, the second column lists the words, the third lists the names of the supertags assigned to each word by a supertagger. The node requirement of each supertag is shown in column four and the dependency links among the words, computed by the above algorithm, is shown in the seventh column. The two pass process for dependency linking is shown in the fifth and the

[2] This representation is a variant of categorial grammar representation.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

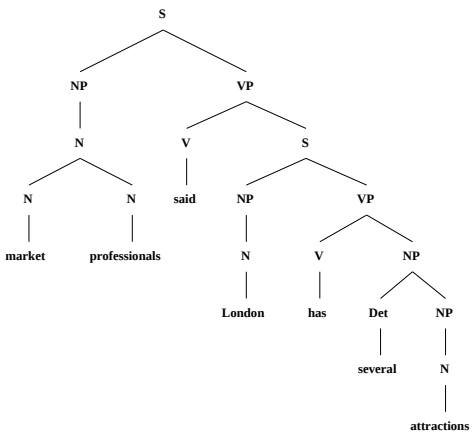


Fig. 6. The parse tree for the sentence *market professionals said London has several attractions.*

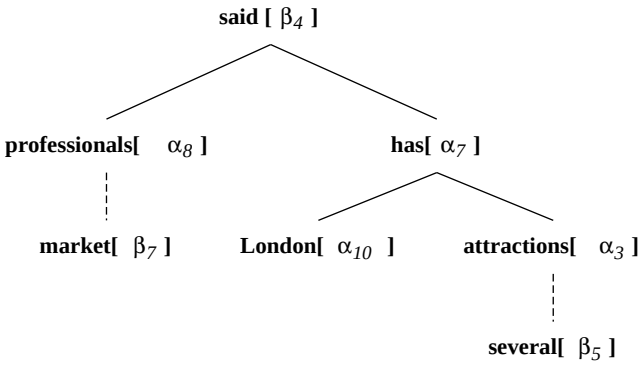


Fig. 7. The derivation tree for the sentence *market professionals said London has several attractions.*

Step 1: For each modifier supertag s in the sentence
Compute the dependencies for s
Mark the words serving as complements as unavailable
for step 2.
Step 2: For the non-modifier supertags s in the sentence
Compute the dependencies for s

Compute Dependencies for s_i of w_i:
For each external node d_{ij} in s_i do
Connect word w_i to the nearest word w_k to the left or right of w_i
depending on the direction of d_{ij}, such that label(d_{ij}) $\in$ internal_nodes(s_k)
skipping over marked supertags, if any.

Fig. 8. Two pass lightweight dependency analyzer.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 1. *An example sentence with the supertags assigned to each word and dependency links among words*

Pos	Word	Supertag	Node req.	Pass 1	Pass 2	Dep Links
0	Market	β_1	+N*	3*	–	3*
1	professionals	α_8	+N*	2*		2*
2	said	β_4	−NP. +S*	3*		3*
3	London	α_{10}	–	–	–	–
4	has	α_7	−NP* +NP.	3* 6.		3* 6.
5	several	β_5	+NP*	6*	–	6*
6	attractions	α_3	–	–	–	–

sixth columns. The * and the . beside a number indicate the type of the dependency relation, * for modifier relation and . for complement relation.

Lightweight Dependency Analyzer (LDA) is a heuristic-based, linear time, deterministic algorithm which is not forced to produce dependency linkages spanning the entire sentence. LDA can produce a number of partial linkages since it is driven primarily by the need to satisfy local constraints without being forced to construct a single dependency linkage that spans the entire input. This, in fact, contributes to the robustness of LDA and promises to be a useful tool for parsing sentence fragments that are rampant in speech utterances exemplified by the Switchboard corpus.

Due to the fact that LDA produces a dependency annotated sentence as the output of the parsing process, parsing evaluation metrics that measure the performance of constituent bracketing such as Parseval (Harrison, Abney, Fleckenger *et al.* 1991) are unsuitable for evaluating the performance of LDA. Although supertags contain constituent information, and it is possible to convert a dependency linkage into a constituency based parse, the Parseval metric does not have provision for evaluating unrooted parse trees since the output of LDA can result in disconnected dependency linkages.

5.1 Discussion

In this section, we discuss a few points about the supertag and LDA system. It is often a topic of tension to balance the domain specificity versus the portability of a natural language grammar. The more domain specific a grammar is, the better is its performance on that domain but the less portable it is to other domains. On the other hand, a wide coverage grammar is portable, but needs to be specialized to exploit the idiosyncrasies of the domain to be efficient. Given a new domain either the parsing system or the heuristics employed by the grammar have to be retrained. In contrast, the modularization of the supertag and LDA system achieves the balance of domain specificity versus portability quite elegantly. The domain dependent aspects are factored out and associated with the supertagger while the LDA continues to be the domain independent unit of the system. The probability

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 2. *An example illustrating the working of LDA on a center-embedded sentence*

Pos	Word	Supertag	Node req.	Pass 1	Pass 2	Dep Links
0	The	β_1	+NP*	1*		1*
1	rat	α_2	–	–	–	–
2	the	β_1	+NP*	3*		3*
3	cat	α_2	–	–	–	–
4	the	β_1	+NP*	5*		5*
5	dog	α_2	–	–	–	–
6	chased	β_5	−NP. −NP*	5. 3*		5. 3*
7	bit	β_5	−NP. +NP*	3. 1*		3. 1*
8	ate	α_{11}	−NP. +NP.		1. 10.	1. 10.
9	the	β_1	+NP*	10*		10*
10	cheese	α_2	–	–		–

Table 3. *An example illustrating the working of LDA on a sentence with long distance extraction*

Pos	Word	Supertag	Node req.	Pass 1	Pass 2	Dep Links
0	who	α_2	–	–	–	–
1	do	β_6	+S*	3*		3*
2	you	α_2	–	–	–	–
3	think	β_4	−NP. +S*	2. 5*		2. 5*
4	Bill	α_2	–	–	–	–
5	believes	β_4	−NP. +S*	4. 7*		4. 7*
6	John	α_2	–	–	–	–
7	loves	α_9	−NP. −NP.		6. 0.	6. 0.

model of the supertagger models the domain specific idiosyncrasies while the LDA uses a wide-coverage grammar and linguistically motivated domain independent structures to compute the dependency linkage. Each time this system is ported to a new domain, the only module that needs retraining is the supertagger. The data used by LDA remains unchanged since this module is not trained on any particular domain.

It is interesting to observe that LDA works by first identifying the adjunct structures using the distinction between recursive and non-recursive supertags. Once the arguments of recursive structures are identified, they are considered to be skippable for establishing the arguments of non-recursive structures. The two-pass strategy of LDA simulates a stack. Further, LDA relies on the locality of the arguments modulo recursive structures to establish the dependency links. This is made possible due to the two properties of LTAGs, extended domain of locality and factoring of recursion. The effectiveness of LDA on some examples involving relative clauses and sentence complement are shown in Tables 2 and 3.

It must be noted that the correctness of the parse produced by the LDA is dependent on the accuracy of the supertagger in assigning supertags to words. As mentioned earlier, the supertagger performs at an accuracy of 92.2% on WSJ

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

corpus. However, if the supertagger were to produce multiple supertags per word, the possibility of a word being associated with the correct supertag increases dramatically. With an average of about three supertags per word, the accuracy of the supertagger increases to 97%. The output of an N-best supertagger could be used by an extended LDA in order to both select the correct supertag as well as assign a dependency structure to words. However, we believe such an extension to the LDA would result in it being similar to a conventional parser such as a CKY parser and the elegance of the LDA being a simple, robust stapler of supertags would be lost.

6 Limitations of Parseval

The Parseval scheme served its purpose of providing a cross-representational evaluation metric using which the performance of various parsers could be quantified and compared – a hither-to-fore impossible task. However, there have been several objections to the Parseval scheme. In this section, we summarize some of the objections and limitations of this scheme.

First, it was observed that crossing brackets measure penalizes mis-attachments more than once. A detailed description is presented in (Lin 1995). Furthermore, the precision measure penalizes a parser for inserting extra, possibly correct, brackets, if annotation in the treebank is skeletal (Srinivas, Doran and Kulick 1995). Due to these limitations of the Parseval metric, it is unclear as to how the score on this metric relates to success in parsing. It is also not clear if this metric can be used for comparing parsers with different degrees of structural fineness since the score on this metric is tightly related to the degree of structural detail in the gold standard.

A significant limitation of the Parseval metric is that it has not been designed to evaluate parsers that produce partial parses. A partial parser could potentially produce constituents/chunks that may not be connected in a complete tree. In cases where an attachment of a constituent is hard to make, it might be equally useful to leave the constituent unattached than to force the parser to always attach it to some node. However, the bracket representation used by Parseval is inadequate to represent disconnected trees. In the bracket notation, there is an implicit assumption that the unattached constituents are attached to the root of the tree.

Another significant limitation of the Parseval metric is that it does not compare analyses to parsing tasks or applications. Parseval metric computes crossing brackets, recall and precision as a way of approximating for the exact match criterion (identity of the output parse with the gold standard parse). Performance using the exact match criterion has a direct interpretation in terms of performance on tasks that a parser may be put to use. However, a measure that is an approximation to the exact match criterion, such as Parseval, has a less direct interpretation since the definition of approximation varies from task to task. Applications that use a parser may be interested in specific structures, Noun Phrases, Appositives, Predicative Nominatives, Subject-Verb-Object relations and so on. Parseval metric is not fine-grained enough to evaluate parses with respect to specific syntactic phenomena. It divorces the parser from the application the parser is embedded in.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

7 Our proposal

We present a proposal for parser evaluation that overcomes the limitations of the Parseval scheme and is also applicable for evaluating partial parsers. It is also intended to serve as a metric for evaluating parsers that are embedded in applications. We recommend two types of evaluation: Application Independent Evaluation and Application Dependent Evaluation.

7.1 Application independent evaluation

The objective of this evaluation is to provide a measure that can be used to compare parsers irrespective of their output representation. Parsers are either constituency-based or dependency-based systems. Constituency-based parsers produce a hierarchically organized constituent structure, while dependency-based parsers produce a set of dependency linkages between the words of a sentence. Furthermore, parsers could produce full parses that span the entire input string or partial parses that are a set of locally consistent parses for non-overlapping substrings of the input. Also, in terms of the detail of a parse, statistical constituency-based parsers produce relatively flat structures for Noun Phrases and Verb groups, since the treebanks they are trained off of do not annotate for the internal structures of these elements. Hence, parsers that do produce internal structure for these elements need to be normalized first so that non-recursive phrasal constituents are flattened to chunks such as noun chunks, verb chunks, preposition chunks.

Also, hierarchically organized constituency notation does not have the provision to represent partial parses since an 'unattached constituent' is implicitly assumed to attach to the highest constituent. Hence to facilitate the comparison of partial and full parsers, we propose that the accuracy of hierarchical structures be measured in terms of the accuracy of the relations between chunks that are expressed by dependency links between certain words (typically heads) of the chunks. Partial parses are thus penalized by their not being able to express certain dependency links due to unattached constituents.

Thus we propose a normalized representation that is based on chunks and dependency links. Evaluating this representation amounts to first evaluating the flat phrasal structures such as noun chunks, verb groups, preposition phrases (disregarding the attachment location) and then evaluating the correctness of hierarchical structures using dependencies between words (typically, heads) of the two chunks. If the dependency links are labeled then evaluation could be performed with and without including the labels. In contrast to our proposal (Lin 1995), proposed using a dependency linkage model for the complete sentence. We believe that a model that combines chunks and dependency links is more appropriate since dependency links in certain parts of a sentence such as noun chunks cannot be annotated with high degree of annotator agreement.

7.2 Application dependent evaluation

The objective of this evaluation is to serve as an adequacy evaluation (see Cole *et al.*, 1996) of a parser. Adequacy Evaluation involves determining the suitability

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

- Let x R y represent that x is in a relation R with respect to y.
- Let S_{gold} be the relation, R, expressed in the key (annotated corpus).
- Let S_{out} be the relation, R, expressed in the output of the parser.
- Recall = $(S_{gold} \cap S_{out})/S_{gold}$.
- Precision = $(S_{gold} \cap S_{out})/S_{out}$.
- F-Measure = $((\beta^2 + 1) * P * R)/(\beta^2 * P + R)$ where β is the relative importance of Recall and Precision.

Fig. 9. Summary of the relation model of parser evaluation.

of a system for a task. It provides users (application developers) information to determine whether a parser is adequate for their particular task, and if so, whether it is more suited than others. The result of this evaluation is a report which provides comparative information of the parsers and does not necessarily identify the best parser, thus allowing the user to make an informed choice.

One of the dimensions of adequacy evaluation is to evaluate parsers from the point of view of specific grammatical constructions such as minimal and maximal Noun phrases, Appositives, Preposition Phrase modifiers, Predicative constructions, Relative Clauses, Parentheticals and Predicate-Argument relations. The degree of difficulty and accuracy of identifying these grammatical constructions, given a parse, depends on the representation adopted by the grammar underlying the parser. The appropriateness of the representation for a task can only be evaluated by evaluating the accuracy with which these grammatical constructions can be identified. Hence we recommend that parsers be evaluated based on their performance in identifying specific grammatical constructions.

7.3 A general framework for parser evaluation

In the preceding sections, we have identified two kinds of parser evaluations – application independent evaluation and application dependent evaluation. In this section, we propose a general framework called the Relation Model of parser evaluation that allows for the two kinds of evaluations to be instantiated in it. In this framework, a parse is viewed in multiple dimensions (e.g. constituency, dependency, chunks, attachments) with each dimension expressing a relation R. A parser can be evaluated along any one (or all) of these dimensions. A gold standard used for evaluation is viewed as expressing one particular relation. Performance of the parser in expressing the relation R is measured in terms of recall, precision and F-measure. F-measure provides a parameter β that can be set accordingly, so as to measure the performance of a system depending on the relative importance of recall to precision. A summary of the evaluation framework is shown in Figure 9.

A few instantiations of this general framework are as follows: For evaluating chunks of type t where t could be a noun chunk, verb chunk, preposition phrase and so on; R is defined as the relation: x *starts* and y *ends* the chunk type t. A similar instantiation could be used to evaluate the parser on specific constructions. For evaluating dependencies, the relation R is defined as x *depends on* y. The dependency links could be evaluated further based on their labels.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

8 Evaluation of Supertag and LDA system

In this section, we discuss the performance of supertagging for text chunking and performance of LDA for dependency analysis on a range of corpora.

8.1 Performance of supertagging for text chunking

We have applied the supertag and Lightweight Dependency Analyzer (LDA) system for text chunking. Text chunking, proposed by Abney (1991) involves partitioning a sentence into a set of non-overlapping segments. Text chunking serves as a useful and relatively tractable precursor to full parsing. Text chunks primarily consist of non-recursive noun and verb groups. The chunks are assumed to be minimal and hence non-recursive in structure.

8.1.1 Noun Chunking

Supertag based text chunking is performed by the local application of functor-argument information encoded in supertags. Once the head noun of the noun chunk is identified, the prenominal modifiers that are functors of the head noun or functors of the functors of the noun are included in the noun chunk. The head NP is identified as the noun with a initial (non recursive) supertag (α_1 or α_8 in Figure 1). Thus, the following simple algorithm identifies noun chunks, for example, shown in statements (3), (4) and (5).

Algorithm *Scan right to left starting with the noun initial supertag and collect all functors of a noun or a noun modifier.*

Some example noun chunks identified by this algorithm from supertagged *Wall Street Journal* (WSJ) texts are shown in below.

(3) New Jersey Turnpike Authority

(4) its increasingly rebellious citizens

(5) two $400 million real estate mortgage investment conduits

Ramshaw and Marcus (1995) present a transformation-based noun chunker using three specialized tags (I, O, B). Each word of the training corpus is annotated to indicate if the word is included in a noun chunk (I), outside a noun chunk (O), or at the boundary of two noun chunks (B). A set of transformation rules are then learned using an error-driven transformation-based learning scheme presented in Brill (1993). The performance of this scheme using rules defined on parts of speech and rules incorporating lexical information is shown in Table 4.

Table 4 also shows the performance of the supertag based noun chunking, trained and tested on the same texts as the transformation-based noun chunker. The supertag-based noun chunker performed better than transformation-based noun chunker with lexical templates, even though the supertag-based noun chunker does not use lexical context in determining the extent of a noun chunk. The supertagger uses lexical information on a per word basis only to pick an initial set of supertags

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 4. *Performance comparison of the transformation based noun chunker and the supertag based noun chunker*

System	Training size	Recall	Precision
Ramshaw & Marcus	Baseline	81.9%	78.2%
Ramshaw & Marcus (with lexical information)	200,000	92.3%	91.8%
Ramshaw & Marcus (with lexical information)	950,000	93.5%	93.1%
Supertags	Baseline	74.0%	58.4%
Supertags	200,000	93.0%	91.8%
Supertags	1,000,000	93.8%	92.5%

for a given word. Moreover, the supertag-based noun chunking not only identifies the extents of the noun chunks but by the virtue of the functor-argument information, provides internal structure to the noun chunks. This internal structure of the noun chunks could be utilized during subsequent linguistic processing.

8.1.2 Verb chunking

We also performed an experiment similar to noun chunking to identify verb chunks. We treat a sequence of verbs and verbal modifiers, including auxiliaries, adverbs, modals as constituting a verb group. Similar to the noun chunk identification algorithm discussed above, the verb chunking based on supertags can be performed using local functor-argument information encoded in supertags. However, for verb chunks, the scan is made from left to right starting with a verbal modifier supertag, either an auxiliary verb or an adverbial supertag and including all functors of a verb or a verb modifier. Thus the following simple algorithm identifies verb chunks, for example, shown in statements (6), (7) and (8).

Algorithm *Scan left to right starting with the verb or verbal modifier supertag and collect all functors of a verb or a verb modifier.*

(6) would not have been stymied

(7) didn't even care

(8) just beginning to collect

Table 5 shows the performance of the transformation based verb chunker and the supertag based verb chunker, both trained and tested on the same sentences of the Wall Street Journal Corpus. The supertag-based verb chunker had higher precision than the transformation based verb chunker with lexical information but had lower recall. The supertag-based verb chunking not only identifies the extents of the verb

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 5. *Performance comparison of the transformation based verb chunker and the supertag based verb chunker*

System	Training size	Recall	Precision
Ramshaw & Marcus	Baseline	60.0%	47.8%
Ramshaw & Marcus (without lexical information)	200,000	83.6%	83.5%
Ramshaw & Marcus (with lexical information)	200,000	88.5%	87.7%
Supertags	Baseline	76.3%	67.9%
Supertags	200,000	86.5%	91.4%
Supertags	1,000,000	87.2%	92.2%

chunks but by the virtue of the subcategorization information, provides internal structure to the verb chunks. This internal structure of the verb chunks could be utilized during subsequent linguistic processing.

8.1.3 Preposition phrase attachment

Supertags distinguish a noun-attached preposition from a verb-attached preposition in a sentence, as illustrated by the two different supertags in statements (9) and (10). Due to this distinction, the supertagging algorithm can be evaluated based on its ability to select the correct supertag for the prepositions in a text.

(9) sell 500 railcar platforms to/B_vxPnx Trailer Train Co. of/B_nxPnx Chicago

(10) begin delivery of/B_nxPnx goods in/B_vxPnx the first quarter

The supertagger makes the preposition disambiguation (by choosing between B_vxPnx and B_nxPnx) based on a trigram context. It performs at 81.1% accuracy. More often than not, the preposition is not within the same trigram window as the noun and the verb due to modifiers of the noun and the verb. Furthermore, the preposition disambiguation is carried out in an on-line fashion as part of the supertag disambiguation process as opposed to the classifier-based solutions proposed in the literature (Hindle and Rooth 1991; Ratnaparkhi, Reynar and Roukos 1994; Brill and Resnik 1994; Collins and Brook 1995), where the classifier is trained on at least the verb, noun and preposition triples.

8.1.4 Other constructions

In this section, we summarize the performance of the supertagger in identifying constructions such as appositives, parentheticals, relative clauses and coordinating conjunctions.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Appositives In the XTAG grammar[3], the appositive construction has been analyzed using a Noun Phrase modifying supertag that is anchored by a comma which takes the appositive Noun Phrase as an argument, for example, the comma in statement (11) and the second comma in statement (12) anchor appositive supertags. Thus, a comma in a sentence could either anchor an appositive supertag or a set of coordination supertags, one supertag for each type of coordination. Further, a comma also anchors supertags that mark parentheticals as in statement (13). The task of the supertagger is to disambiguate among the various supertags and assign the appropriate supertag to the comma, in the appositive context. In Table 6, we present the performance results of the supertagger on such a task. The baseline of assigning the most likely supertag to comma results in zero percent recall since the most likely supertag for a comma is the co-ordination supertag.

Parentheticals Propositional attitude verbs such as *say* and *believe* can not only appear in the canonical word order of subject-verb-complement, but can also appear at other positions in a sentence, as in statements (12) and (13). Also, the relative order of the subject and the verb can also be reversed as in statement (12). The XTAG grammar distinguishes such adverbial constructions from the sentence complement construction by assigning different supertags to the verb. A detailed analysis of this construction is presented in Doran (1996). The task of the supertagger is to disambiguate among the sentence complement supertag and the various adverbial supertags for the verb given the context of the sentence. Table 6 presents the performance results of the supertagger on this task. Once again the baseline of selecting the most likely supertag of the verb results in zero percent recall.

(11) Van Pell, 44

(12) "The U.S. underestimated Noriega all along," *says* Ambler Moss, a former Ambassador to Panama

(13) Mr. Noriega's relationship to American intelligence agencies became contractual in either 1966 or 1967, intelligence officials *say*.

Coordination Conjunctions In Table 6, we also present the performance of the supertagger in identifying coordination constructions. Coordination conjunctions anchor a supertag that requires two conjuncts, one on either side of the anchor. There is one supertag for every possible pair of conjunct types. We not only have supertags that coordinate like types but we also include supertags that coordinate unlike types amounting to about 30 different supertags for coordination.

Relative Clauses Due to extended domain of locality of LTAGs, verbs are associated with one relative clause supertag for each of the arguments of the verb. This is unlike CCG, where the relative pronoun is associated with multiple supertags. A more detailed discussion about the differences in the distribution of ambiguity between the CCG and LTAG is presented in (Doran and Srinivas 1994). The task

[3] A wide coverage grammar being developed at University of Pennsylvania (XTAG-Group 1995).

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 6. *Performance of the trigram supertagger on appositive, parentheticals, coordination and relative clause constructions*

Construction	# of occurrences	# identified	# correct	Recall	Precision
Appositive	362	491	306	84.5%	62.3%
Parentheticals	225	200	170	75.5%	85%
Coordination	1886	1750	1329	70.5%	75.9%
Relative Clauses	297	269	116	39.0%	43.2%
Relative Clauses (ignoring valency)	297	269	156	52.5%	58%

of the supertagger in LTAG is not only to identify that the verb is in a relative clause structure, but also to identify the valency of the verb and the argument that is being relativized. The performance of the supertagger with and without the valency information being taken into account is present in Table 6.

We are unaware of any other quantitative results on WSJ data for identifying these constructions, for comparative evaluation.[4] It is interesting to note that the performance of the supertagger in identifying appositives and parenthetical construction is better than for coordination conjunctions and relative clause constructions. We believe that this might in part be due to the fact that Appositives and Parentheticals can mostly be disambiguated using relatively local contexts. In contrast, disambiguation of Coordinate constructions and Relative clauses typically require large contexts that are not available for a trigram supertagger.

8.2 Performance of supertag and LDA system

In this section, we present results from two experiments using the supertagger in conjunction with the LDA system as a dependency parser. In order to evaluate the performance of this system, we need a dependency annotated corpus. However, the only annotated corpora we are aware of, the Penn Treebank (Marcus, Santorini and Marcinkiewicz 1993) and the SUSANNE corpus (Sampson 1994), annotate sentences with constituency trees. In the interest of time and the need for a dependency annotated corpus, we decided to transform the constituency trees of the two corpora into dependency trees using some rules and heuristics. It must be noted that the resulting corpora are only but an approximation to a manually annotated dependency corpus. However, although the annotation resulting from the transformation does not conform to a standard dependency annotation, it

[4] Lin (1996) provides results for some of these constructions on the Brown corpus.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

is nevertheless an invaluable resource for performance evaluation of dependency parsers.

The WSJ corpus from the Penn Treebank and the subset of the Brown corpus in SUSANNE were converted by two related but different mechanisms. The conversion process of the Penn Treebank proceeded as follows. For each constituent of a parse a head word is associated using the head percolation table introduced by Jelinek *et al.* (1994) and Magerman (1995). The head percolation table associates with each possible constituent label an ordered list of possible children of the constituent whose head word is passed to the parent. The annotation of a parse tree with head word information proceeds bottom up. At any constituent, the percolation information is consulted to determine which of the constituent's children would pass the head word over to their parent. Once the head words are annotated, the dependency notation is generated by making the head words of non-head constituents to be dependent on the head of the head constituent. An example of this procedure is illustrated in Figure 10.

In a similar manner to the WSJ conversion process, we converted the subset of the LOB annotation of the SUSANNE corpus into a dependency notation. However, in this conversion process we could not use the head information table used in the WSJ conversion process since the annotation labels are different. Instead, we regard a constituent label to be a projection of one of its children, which was regarded as the head constituent. Once the heads are annotated the dependency notation is generated by making the head words of non-head constituents to be dependent on the head word of the head constituent.

In the first experiment we use the dependency versions of the Penn Treebank annotation of the WSJ corpus and the LOB annotation of the SUSANNE corpus as gold standards. However, in the second experiment, we use derivation structures of WSJ sentences that were parsed using the XTAG system as the gold standard. The derivation structures serve as dependency structures that are closest in conventions to those assumed by the LDA system.

Experiment 1

The trigram supertagger trained on 200,000 words of the WSJ corpus was used as in conjunction with the LDA to provide a dependency analysis for 2,000 sentences of Section 20 of the WSJ corpus. The Penn Treebank parses for these sentences were converted into dependency notation which was used as the gold standard. A dependency link produced by the LDA was regarded to be correct if the words being related by the link were also present in the gold standard. However, to account for the differences in the annotation, the dependency relations among words in a noun group were treated as equivalent to each other. So also, dependency relations among words in a verb group were treated as equivalent to each other. There were 47,333 dependency links in the gold standard and the LDA produced 38,480 dependency links correctly, resulting in a recall score of 82.3%. Also, a total of 41,009 dependency links were produced by the LDA, resulting in a precision score of 93.8%.

We conducted a similar experiment using the SUSANNE corpus. Using the

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))

1 Pierre 2 Vinken
2 Vinken  8 will
3 , 2 Vinken
4 61 5 years
5 years 6 old
6 old 2 Vinken
7 , Vinken
8 will 8 will
9 join 8 will
10 the 11 board
11 board 9 join
12 as 9 join
13 a 15 director
14 nonexecutive 15 director
15 director 12 as
16 Nov. 9 join
17 29 16 Nov.
18 . will
```

Fig. 10. The phrase structure tree and the dependency linkage obtained from the phrase structure tree for the WSJ sentence *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.*

same trigram supertagger trained on the 200,000 words of the WSJ corpus in conjunction with the LDA, we provided a dependency analysis for the sentences in the SUSANNE corpus (125,000 words). The gold standard was created by converting the phrase structure annotations for these sentences into dependency notation using the procedure described above. As in the case of WSJ corpus, to account for the differences in the annotation, the dependency relations among words in a noun group were treated as equivalent to each other. So also, dependency relations among words in a verb group were treated as equivalent to each other. There were a total of 126,493 dependency links in the output of the LDA of which, 112,420 also present in the gold standard, resulting in a precision score of 88.8%. There were a total of 140,280 links in the gold standard, resulting a recall of 80.1%. It is interesting to note

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 7. *Comparative evaluation of LDA on the Wall Street Journal and Brown Corpus*

Corpus	# of	# produced dependency links	# correct by LDA	Recall	Precision
Brown	140,280	126,493	112,420	80.1%	88.8%
WSJ	47,333	41,009	38,480	82.3%	93.8%

that although the trigram model was trained on the WSJ corpus, the performance of the LDA on SUSANNE is comparable to the performance of the LDA on the WSJ corpus.

On analyzing the errors, we discovered that several of the errors were due to the approximate nature of the dependency corpus created by the conversion process. A second source of errors was due to the differences in the notion of dependency produced by the LDA system and that resulting from the conversion of the Treebank. This is largely due to a lack of a standard dependency notation.

Experiment 2

In this experiment, we used the derivation structures produced by XTAG for 1350 WSJ sentences (of length less than 16 words) as the gold standard. The correct derivation was hand picked for each of the 1350 WSJ sentences. These sentences were supertagged using the supertagger trained on 8,000 sentences of WSJ and dependency annotated using the LDA system. The metric for evaluation was Recall and Precision computed based on the number of dependency links present in both the gold standard and the output produced by the system. Table 8 shows the performance of the system. Although the derivation trees produced by the XTAG system are closest in terms of conventions used in the dependency output of the LDA system, there are a few points of divergences, a major one being the annotation for sentence complement verbs. While in the LDA system, the embedded verb depends on the matrix verb the reverse is the case in an LTAG derivation structure (Rambow, Weir and Vijay-Shanker 1995). Also, since the XTAG system, as it is implemented does not permit two auxiliary trees to be adjoined at the same node, certain valid derivation structures are not possible in XTAG. A precise formulation of possible derivation structures in LTAG framework is presented in (Schabes and Shieber 1992).

Next, we present the performance of the LDA in terms of its accuracy at producing a dependency linkage at the level of a sentence, when evaluated against an LTAG derivation tree. Table 9 tabulates the percentage of sentences that have zero, one, two and three dependency link errors. In contrast to evaluation against a skeletally bracketed treebank, evaluation against LTAG derivation trees is much more strict. For example, an LTAG derivation contains detailed annotation in terms of the internal structure of the nominal modifiers in Noun Phrases and verbal modifiers in

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Table 8. *Performance of LDA system compared against the XTAG derivation structures*

Training size (words)	Test size (words)	Recall	Precision
200,000	12,000	83.6%	83.5%

Table 9. *Percentage of sentences with zero, one, two and three dependency link errors*

% sentences with 0 errors	% sentences with $\leq$1 error	% sentences with $\leq$2 errors	%sentences with $\leq$3 errors
35%	60.3%	78%	89.8%

Verb groups. Also, a derivation structure has a stricter imposition of the argument-adjunct distinction and distinguishes readings that have similar phrase structure trees such as predicative and equative readings and idiomatic and non-idiom readings of a sentence. Further, the derivation structure is much more closer to semantic interpretation of a sentence than the phrase structure. Hence, the performance figures that are shown in Tables 8 and 9 are more strict and hence more significant than the crossing bracket, precision and recall figures measured against skeletally bracketed corpora.

9 Applications of supertagger and LDA

The outputs of the supertagger and LDA have been used in a variety of applications. In information retrieval applications (Chandrasekar and Srinivas 1997b; Chandrasekar and Srinivas 1997c; Chandrasekar and Srinivas 1997d) supertagged sentences from relevant documents are used to create patterns which are used to match against supertagged sentences from documents whose relevance is to be determined. In information extraction applications (Doran, Niv, Baldwin, Reynar, and Srinivas 1997), patterns based on supertag and LDA output have been used to extract fillers for the fields of information templates. The output of the supertagger distinguishes modifiers from complements and the output of LDA specifies the scope of the modifier clause. In text simplification applications (Chandrasekar, Doran and Srinivas 1996; Chandrasekar and Srinivas 1997a), the distinction between main clauses and modifier clauses has been used to simplify sentences. We have also used the output of the supertagger in class-based language modeling experiments (Srinivas 1996).

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

10 Summary

In this paper, we have presented an alternate method to robust parsing that uses the output of the supertagger to produce dependency analysis. Further, we have discussed the evaluation metrics for parsers that are prevalent and indicated their limitations for comparing parsers that produce different representations. We have proposed an alternate evaluation metric based on relation model of evaluation that overcomes the limitations of the existing evaluation metrics. Using these metrics, we provide extensive evaluation results of the supertagger and the LDA system on a range of tasks and compared its performance against the performance of other systems on those tasks.

Acknowledgments

This work was done when the first author was at University of Pennsylvania. It was partially supported by NSF grant NSF-STC SBR 8920230, ARPA grant N00014-94 and ARO grant DAAH04-94-G0426. We would like to thank Aravind Joshi, Steven Abney, Raman Chandrasekar, Christine Doran, Beth Ann Hockey, Mark Liberman, Mitch Marcus and Mark Steedman for useful comments and discussions which have helped shape this work. We also thank the reviewers for their insightful comments and suggestions to improve an earlier version of this paper.

References

Abney, S. (1990) Rapid Incremental parsing with repair. *Proc. 6th New OED Conference: Electronic Text Research*, pp. 1–9. Waterloo, Canada.

Abney, S. (1991) Parsing by chunks. In: R. Berwick, S. Abney and C. Tenny, editors, *Principle-based Parsing*. Kluwer.

Appelt, D., Hobbs, J., Bear, J., Israel, D. J. and Tyson, M. (1993) FASTUS: a finite-state processor for information extraction from real-world text. *Proc. IJCAI-93*, Chambery, France.

Bangalore, S. and Aravind Joshi (1999) Supertagging: An approach to almost parsing. *Computational Linguistics* **25**(2).

Brill, E. and Resnik, P. (1994) A rule-based approach to prepositional phrase attachment disambiguation. *Proc. Int. Conf. on Computational Linguistics (COLING '94)*. Kyoto, Japan.

Brill, E. (1993) Automatic grammar induction and parsing free text: A transformation-based approach. *Proc. 31st Annual Meeting of the Association for Computational Linguistics*. Columbus, OH.

Chandrasekar, R., Doran, C. and Srinivas, B. (1996) Motivations and methods for text simplification. *Proc. 16th Int. Conf. on Computational Linguistics (COLING '96)*. Copenhagen, Denmark.

Chandrasekar, R. and Srinivas, B. (1997a) Automatic induction of rules for text simplification. *Knowledge-Based Systems* **10**: 183–190.

Chandrasekar, R. and Srinivas, B. (1997b) Gleaning information from the web: Using syntax to filter out irrelevant information. *Proc. AAAI 1997 Spring Symposium on NLP on the World Wide Web*.

Chandrasekar, R. and Srinivas, B. (1997c) Using supertags in document filtering: The effect of increased context on information retrieval effectiveness. *Proc. Recent Advances in NLP (RANLP) '97*. Tzigov Chark.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Chandrasekar, R. and Srinivas, B. (1997d) Using syntactic information in document filtering: A comparative study of part-of-speech tagging and supertagging. *Proc. RIAO'97*. Montreal, Canada.

Chelba, C. and Jelinek, F. (1998) Exploiting Syntactic Structure for Language Modeling. *Proc. ACL '98*, pp. 225–231. Montreal, Canada.

Church, K. W. (1988) A stochastic parts program and noun phrase parser for unrestricted text. *2nd Applied Natural Language Processing Conference*. Austin, TX.

Cole, R. A., Mariani, J., Uszkoreit, H., Zaenen, A. and Zue, V. (1996) Survey of the state of the art in human language technology. `http://www.cse.ogi.edu/CSLU/HLTsurvey/`.

Collins, M. (1996) A New Statistical Parser Based on Bigram Lexical Dependencies. *Proc. 34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, NM.

Collins, M. (1997) Three Generative, Lexicalised Models for Statistical Parsing. *Proc. 35th Annual Meeting of the Association for Computational Linguistics*. Madrid, Spain.

Collins, M. and Brook, J. (1995) Prepositional phrase attachment through a backed-off model. *Proc. 3rd Workshop on Very Large Corpora*. Cambridge, Boston, MA.

Doran, C., Niv, M., Baldwin, B., Reynar, J. and Srinivas, B. (1997) Mother of Perl: A Multi-tier Pattern Description Language. *Proc. Int. Workshop on Lexically Driven Information Extraction*. Frascati, Italy.

Doran, C. and Srinivas, B. (1994) A wide-coverage CCG parser. *Proc. 3rd TAG+ Conference*. Paris, France.

Doran, C. (1996) Punctuation in quoted speech. *Proc. SIGPARSE96*. Santa Cruz, CA,

Fujisaki, T., Cocke, J. J., Black, E. and Nishino, T. (1989) A probabilistic parsing method for sentence disambiguation. *Proc. 1st Annual International Workshop of Parsing Technologies*. Pittsburgh.

Grishman, R. (1995) Where's the Syntax? The New York University MUC-6 System. *Proc. 6th Message Understanding Conf*. Columbia, MD.

Harrison, P., Abney, S., Fleckenger, D., Gdaniec, C., Grishman, R., Hindle, D., Ingria, B., Marcus, M., Santorini, B. and Strzalkowski, T. (1991) Evaluating syntax performance of parser/grammars of English. *Proc. Workshop on Evaluating Natural Language Processing Systems*. ACL.

Hindle, D. and Rooth, M. (1991) Structural ambiguity and lexical relations. *29th Meeting of the Association for Computational Linguistics*. Berkeley, CA.

Hobbs, J. R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M. and Tyson, M. (1997) FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In: E. Roche and Schabes Y., editors, *Finite State Devices for Natural Language Processing*. MIT Press.

Jelinek, F., Lafferty, J., Magerman, D. M. Mercer, R., Ratnaparkhi, A. and Roukos, S. (1994) Decision tree parsing using a hidden derivation model. *Proc. ARPA Workshop on Human Language Technology Workshop*, March.

Chen, J., Srinivas, B. and Vijay-Shanker, K. (1999) New models for improving supertag disambiguation. *Proc. 9th Conf. European Chapter of Association for Computational Linguistics*. Bergen, Norway.

Joshi, A. and Hopely, P. (1997) A parser from antiquity. *Natural Lang. Eng.* **2**(4).

Joshi, A. K. (1960) Computation of Syntactic Structure. In: *Advances in Documentation and Library Science*, Vol. III, Part 2. Wiley Interscience.

Joshi, A. K. and Srinivas, B. (1994) Disambiguation of super parts of speech (or supertags): Almost parsing. *Proc. 17th Int. Conf. on Computational Linguistics (COLING '94)*. Kyoto, Japan.

Karttunen, L., Chanod, J.-P., Grefenstette, G. and Schiller, A. (1997) Regular expressions for language engineering. *Natural Lang. Eng.* **2**(4).

Lafferty, J., Sleator, D. and Temperley, D. (1992) Grammatical Trigrams: A Probabilistic Model of Link Grammar. *Technical Report CMU-CS-92-181*, School of Computer Science, Carnegie Mellon University.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press

Lin, D. (1995) A dependency-based method for evaluating broad-coverage parsers. *Proc. IJCAI-96*, Montreal, Canada, August.

Lin, D. (1996) Evaluation of principar with the Susanne corpus. *Proc. Workshop on Robust Parsing at European Summer School in Logic, Language and Information*. Prague.

Magerman, D. M. (1995) Statistical decision-tree models for parsing. *Proc. 33rd Annual Meeting of the Association for Computational Linguistics*.

Marcus, M. M., Santorini, B. and Marcinkiewicz, M. A. (1993) Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19**(2):313–330.

Mohri, M., Pereira, F. C. N. and Riley, M. (1998) *A rational design for a weighted finite-state transducer library*. *Lecture Notes in Computer Science* **1436**. Springer-Verlag.

Rambow, O., Weir, D. and Vijay-Shanker, K. (1995) D-Tree Grammars. *Proc. 33rd Conference of Association of Computational Linguistics*.

Ramshaw, L. and Marcus, M. P. (1995) Text chunking using transformation-based learning. *Proc. 3rd Workshop on Very Large Corpora*. Cambridge, Boston, MA.

Ratnaparkhi, A., Reynar, J. and Roukos, S. (1994) A maximum entropy model for prepositional phrase attachment. *Proc. ARPA Workshop on Human Language Technology*. Plainsboro, NJ.

Ratnaparkhi, A. (1997) A linear observed time statistical parser based on maximum entropy models. *Proc. Empirical Methods in Natural Language Processing*. New Providence.

Roche, E. (1993) *Analyse syntaxique transformationelle du francais par transducteurs et lexique-grammaire*. PhD thesis, Universite Paris 7.

Sampson, G. (1994) SUSANNE: a Doomsday book of English grammar. *Corpus-based Research into Language*. Rodopi, Amsterdam.

Schabes, Y., Roth, M. and Osborne, R. (1993) Parsing the Wall Street Journal with the Inside-Outside Algorithm. *Proc. European ACL*.

Schabes, Y. (1990) *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania.

Schabes, Y., Abeillé, A. and Joshi, A. K. (1988) Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. *Proc. 12th Int. Conf. on Computational Linguistics (COLING'88)*. Budapest, Hungary.

Schabes, Y. and Shieber, S. (1992) An alternative conception of tree-adjoining derivation. *Proc. 20th Meeting of the Association for Computational Linguistics*.

Srinivas, B. (1996) "Almost Parsing" Technique for Language Modeling. *Proc. ICSLP96 Conference*. Philadelphia, USA.

Srinivas, B. (1997a) *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Srinivas, B. (1997b) Performance evaluation of supertagging for partial parsing. *Proc. Fifth Int. Workshop on Parsing Technology*. Boston, USA.

Srinivas, B., Doran, C. and Kulick, S. (1995) Heuristics and parse ranking. *Proc. 4th Annual International Workshop on Parsing Technologies*. Prague.

Strzalkowski, T., Lin, F., Perez-Carballo, J. and Wang, W. (1997) Building effective queries in natural language information retrieval. *Proc. Applied NLP (ANLP) '97*. Washington, DC.

Weischedel, R., Schwartz, R., Palmucci, J., Meteer, M. and Ramshaw, J. (1993) Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics* **19**(2):359–382.

XTAG-Group. (1995) A Lexicalized Tree Adjoining Grammar for English. *Technical Report IRCS 95-03*, University of Pennsylvania. Updated version available at `http://www.cis.upenn.edu/ xtag/tr/tech-report.html`.

https://doi.org/10.1017/S1351324900002345 Published online by Cambridge University Press