LOGICON: A System for Extracting Semantic Structure using Partial Parsing

Kais Dukes
School of Computing, University of Leeds
LS2 9JT, United Kingdom
sckd@leeds.ac.uk

Abstract

Partial parsing is an established NLP technique used to perform syntactic analysis without generating a full constituent parse tree. This paper presents LOGICON, an end-to-end system using partial parsing, which assigns novel semantic structures to natural language text. Evaluating against a test set of 500 previously unseen sentences, the system has an accuracy of 62.4% as measured by exact matching against the expected semantic output. Since partial parsing is used, the system is robust and will assign partial semantic structure to sentences it may not fully understand. As stochastic methods are not used, the system is deterministic and fast. A syntactic tagging scheme is proposed which is closely aligned to the corresponding semantics. The system was developed as part of a PhD research project, and was written to evaluate partial parsing as the first step to creating a full natural language question-answering system.

Keywords

natural language processing, partial parsing, annotated corpora, constituent structure, thematic relations, semantic role labeling, syntax parse trees, part-of-speech tagging

1. Introduction

The LOGICON system was developed as part of an ongoing PhD research project, with the aim of using partial parsing to extract semantic structures from natural language. LOGICON contrasts with PARASITE, a system which produces formal semantics for unrestricted text [9], since partial parsing [1][2] is used instead of deep parsing, and semantic roles are used for representing meaning as opposed to logic statements with variables and quantifiers.

For example, given a simple sentence focusing around an event, LOGICON attempts to identify roles for the *actor* (who did the event), the *action* (what the event was) and the *target* (what entity the actor performed the action on). The system employs simple partial parsing techniques as described by Abney [1], [2]. A syntax-driven approach is then used to derive semantic roles through recursion.

2. Semantic Structures

Thematic relations are an intuitive approach to assigning meaning to the constituents of a sentence. A typical problem is what role to assign to a noun phrase. Traditionally thematic relations include Agent, Patient, Theme, Location, etc. However, there is no definitive list of roles, and in some cases which role to use is not immediately clear: in "the key opened the door" is the key the agent or the instrument? In order to produce a practical system, the research focused on working with a small set of well-defined roles:

Table 1. Semantic relations describing an event

Relation	Description
ACTION	the action, or main verb
ACTOR	the doer of the action
TARGET	what the action was performed on
LOCATION	where the action was performed
TIME	when the action was performed

The ACTOR role is typically assigned to the syntactic subject of the sentence, and the TARGET is typically assigned to the object. Together, the roles are grouped into a semantic structure called an EVENT. For the simple sentence "Jack helped John", the corresponding semantic structure produced by LOGICON would be:

EVENT:
ACTOR: Jack
ACTION: help
TIME: PAST
TARGET: John

A semantic structure called a LINK is used to represent sentences that use a copula to link a subject (the SOURCE) to its predicate (the TARGET). For example, "The apple is red" would have the following semantic structure:

LINK:
SOURCE: apple
TARGET: red

Table 2 below gives a brief summary of the important keywords used in the semantic structures found in the annotated corpus:

Student Research Workshop, RANLP 2009 - Borovets, Bulgaria, pages 18–22

Table 2. Keywords in LOGICON semantic structures

Keyword	Description
SPEAKER	represents the first person
LISTENER	represents the second person
OTHER	represents the third person
OF	used in a possessive construction
CONFIRM	"*Is* the sky blue?"
EXPLAIN	"*Why* is the sky blue?"
QUANTIFY	"How much" / "How many"
PARAMETER	second object (ditransitive verbs)
LOCATION	"Jack put the book *on the table*"
SPECIFIC	represents the definite article
GENERAL	represents an indefinite article
CONCEPT	an isolated noun phrase
POSSIBLE	"Jack *might* eat"
EXPECTED	"Jack *will* eat"
RECOMMENDED	"Jack *should* eat"
MODIFIER	"Jack ate *quickly*" (adverb)
NOT	used to negate a structure
AND	"Jack *and* Mary are clever"
OR	"Eat the apple *or* the orange"

Special handling is given to pronouns and to possessive constructions, using the first 4 keywords listed in table 2. As an example, LOGICON translates the sentence "You broke my car" into the corresponding semantic structure:

EVENT:
ACTOR: LISTENER
ACTION: break
TIME: PAST
TARGET: car OF SPEAKER

The aim of the system is to produce enough semantic detail to enable effective question-answering. Since partial parsing, and not deep parsing is used, some structures are not dissected. For example the internals of noun phrases are not handled directly.

In this respect, the semantic structures can be considered a form of semantic role labeling. The structures are general-purpose so that it would be more accurate to say that they are an intermediate form between the frame semantics found in FrameNet [3] and the verb-argument annotations found in the PropBank corpus [8].

2.1 Semantic recursion

Since natural language is inherently recursive, it is not unreasonable to expect that any corresponding semantic structures should show similar recursion. The semantic structures generated by LOGICON are syntactically-driven, that is to say they are derived directly from parse trees constructed via partial parsing. Since these parse trees are recursive structures, so are the corresponding semantics.

A simple example would be the sentence "Who said time is money?". In the corresponding semantic structure, this is analyzed as a LINK (a subject/predicate construction) embedded within an EVENT (an action in time or space):

EVENT:
ACTOR: UNKNOWN
ACTION: say
TIME: PAST
TARGET: LINK:
SOURCE: time
TARGET: money

The actor (the doer of the action) is UNKNOWN ("who?"). The UNKNOWN keyword is used as a placeholder for thematic roles in sentences which use interrogative pronouns. In the event structure above, the target of the action (what was said) is itself another semantic structure, a link between a subject and its predicate: "time *is* money".

3. Partial Parsing

3.1 Abney's partial parsing scheme

The partial parsing scheme introduced by Abney [1] and implemented in the Cass partial parser [2], successively builds a parse tree bottom-up by using a cascade of finite state transducers. Customizable patterns are used to define the regular expressions used to parse at each level. These patterns are specified in a human readable format (similar to Backus-Naur form) and are then complied into a unified finite state transducer automatically. A distinguishing feature of the original scheme is that there is no definite top-level node representing the entire sentence. The system is more like a chunking analyzer as opposed to a full syntactic parser.

The main advantages of the Cass partial parsing scheme is that it is robust (it will not fail to produce a partial analysis given input it may not fully understand), it is fast (orders of magnitude faster than stochastic parsers) and relatively easy to implement.

3.2 The annotated corpus

An annotated corpus was constructed at the start of the project. By adopting a corpus-driven methodology, the effectiveness of potential parser rules was decided by available corpus evidence. The annotations were produced as follows:

1. A set of 2000 sentences was collected.
2. For each sentence, the semantic structure expected to be produced by the system was manually annotated.
3. From the expected semantic structure, a syntactic parse tree was also annotated that would provide the suitable semantic skeleton from which to derive the semantics.

After annotation the corpus was divided into two sets: a training set of 1500 sentences, and an evaluation set of 500 sentences. The training set would be used as a reference when building the system, in order to test the effectiveness of the parser during its construction, and to try out various partial parsing rules. An example annotated sentence is shown below:

"Who wrote 'The Moon is a Harsh Mistress?'"

```
(EV
    (C Who) (V wrote)
    (LN
        (C (Q The) (C Moon))
        (AUX is)
        (C (Q a) (C Harsh Mistress))))

EVENT:
    ACTOR: UNKNOWN
    ACTION: write
    TIME: PAST
    TARGET: LINK:
                SOURCE: SPECIFIC Moon
                TARGET: GENERAL Harsh Mistress
```

The following resources were used to construct corpus:

1. Example sentences from the Link Grammar Parser [10].
2. Sentences based on patterns in A.L.I.C.E. [11].
3. Questions from the TREC-10 QA track [6].
4. Sentences from novels in Project Gutenberg.
5. News headlines from news.google.com.

Different sources were used so that a wide-coverage parser could be constructed. Focusing on a particular genre – such as newspaper text – might have resulted in a more limited parser. A large proportion of the data was derived from the question templates found in the A.L.I.C.E. chat program, which is relevant to question-answering because this data was formed after studying the most frequent inputs given to a popular chat system.

3.3 Partial parsing in LOGICON

Three possible parsing schemes were considered at the outset of the project: using a stochastic parser such as Bikel's parser [4], using a dependency parser such the Link Grammar Parser [10], or using a partial parser. An existing stochastic parser was not suitable for use in LOGICON, because either these are pre-trained on a different tagset or need to be trained using a large corpus. It was felt that converting the output from the Link Grammar Parser would be too time consuming, so it was decided to construct a partial parser which matched the syntax in the annotated corpus. A Brill tagger using transformation-based machine learning was first applied to the training set [5].

With an effective part-of-speech tagger in place, Abney's original partial parsing scheme was then adapted. Initially, this yielded encouraging results. Out of the 1500 sentences in the training corpus, 7 simple rules resulted in partial syntax trees which had an accuracy of 90.84% as measured by the number of nodes parsed and connected to the correct constituent nodes. A total of 35 rules were finally used.

3.4 The partial parsing algorithm

The partial parsing algorithm used by the LOGICON system is described as follows[1]. The parser constructs a parse tree bottom-up. At each stage of its operation, there is a set of top-level nodes, which are grouped together to form new top-level nodes at the next iteration. The result of the algorithm is a partial parse tree, defined as a set of one or more final top-level nodes, each of which is a complete parse tree:

1. Construct a node for each word, using part-of-speech tags from the tagger.
2. For each parser rule R, apply R to the top-level nodes.
3. If at least one rule did apply, repeat step 2 until no rules apply, and no new top-level nodes can be created.

Figure 1. Rule in partial parser specifying a new node

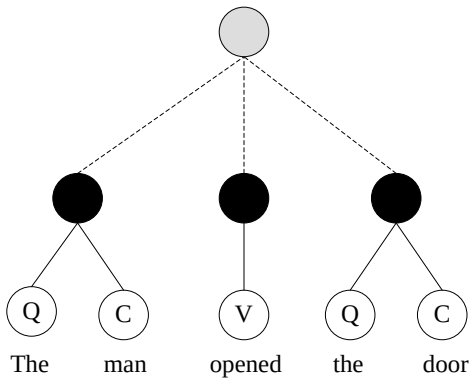


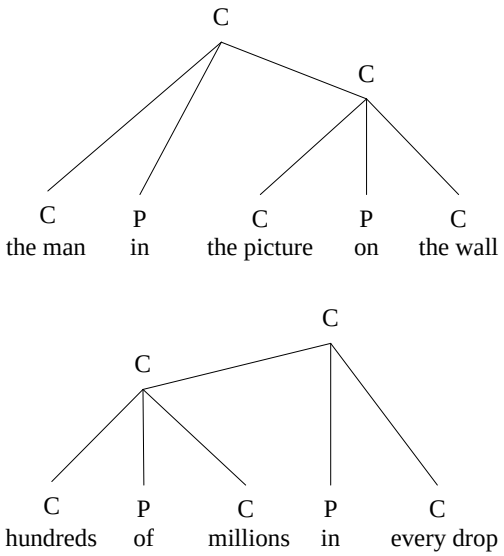
Figure 1 above shows a syntactic structure constructed by the parser during its analysis. The current top-level nodes are shown in black. At this stage of operation, the three top-level nodes represent a noun-phrase, followed by a verb, followed by a second noun phrase. These would be recognized by a parser rule as a subject-verb-object construction, and these 3 nodes would be collected into a new top-level node shown in gray.

[1] See the appendix for a description of the partial parsing algorithm in pseudo-code

The ordering of rules in the parser is crucial, as this represents rule precedence. The rules which operate on lower parts of the parse tree are listed first. Rules which operate on similar syntactic structures are listed together. However they are ordered so that the rule which is more applicable in general will operate first, to create new top-level nodes with the correct precedence.

Figure 2 below shows two sentences with the same part-of-speech tags, but with different constituent structure. In the annotated corpus, two noun phrases separated by a preposition (e.g. 'the picture *on* the wall') are grouped together into a symmetric compound noun phrase. This analysis is used to support the expected semantic structure. Ambiguity arises when the parser is faced with the sequence C + P + C + P + C. As figure 2 indicates, this can be analyzed in one of two ways.

Figure 2. Ambiguity in preposition and noun phrase structure



The first right-to-left grouping was found to be more common in the corpus. The rule which deals with prepositions builds this structure by default. The second grouping also occurs, and in this case the behavior is overridden by adding lexical information to the parser rule.

3.5 The tagging scheme

It was decided to keep the tagging scheme used by LOGICON to be as close as possible to the final semantic structure. A sample sentence found in the training corpus is: "Who was the lead singer for the Commodores?". This is analyzed as a copula link, with the subject 'who'. The predicate is a compound phrase with a preposition linking two noun phrases ('the lead singer' and 'the Commodores'). The syntactic structure constructed via partial parsing is:

```
(LN
    (C Who) (AUX was)
    (C
        (C (Q the) (C lead singer))
        (P for)
        (C (Q the) (C Commodores))))
```

A reduced tagset of 18 tags is used by the partial parser:

Table 3. Semantically-aligned tagset used for syntax tree nodes

Tag	Description or example
N	noun
V	verb
AUX	auxiliary verb
P	preposition
Q	quantifier / determiner
SYM	symbol / punctuation
NEG	negation ("no, not")
POS	possessive ("Jack's house")
CONJ	conjunction ("and, or")
T	time phrase ("2pm")
LOC	location phrase ("in London")
COMP	complementizer / relative pronoun
C	concept / noun phrase
EV	subject-verb-object event
LN	subject-copula-predicate link
XL	explanation question ("why")
MOD	modifier (adverb)
OP	mathematical operation ("2 + 2")

3.6 Mapping partial parsing to semantics

The translation algorithm is a recursive map. The translator accepts as input the results of the partial parser and performs a recursive algorithm which visits each node in turn, bottom-up. A sequence of semantic translation rules are applied, with each rule operating on a pre-defined syntactic tag. Thematic relations are deduced directly from the syntax, designed with semantic analysis in mind. For example, rules will map a verb tag to an ACTION role, and a time tag to a TIME role. Since a parent node's children will already have semantics attached, these structures can be used to construct the next-level of analysis, and so on, until the entire partial tree has semantics attached to each node. The final output of the LOGICON system is the resulting semantic structure attached to the top-level nodes.

4. Evaluation

4.1 Evaluation against the annotated corpus

The annotated corpus was divided into a training set of 1500 sentences and an evaluation set of 500 sentences. When applied to the evaluation set, the system produced the exact expected semantic output for 312 of the 500 sentences, giving an accuracy score of 62.4%.

Table 4. Evaluation matching against exact expected semantics

Matched	Not matched	Total	%
312	188	500	62.4

A qualitative analysis of the errors indicated that most of the inaccuracies were due to the part-of-speech tagger. The evaluation set contained words not previously seen by the Brill tagger which resulted in incorrect parts-of-speech.

4.2 Parsing speed

Despite the lower than expected accuracy, the system did demonstrate a good trade-off between speed versus depth of analysis. The algorithm presented is wholly deterministic, making no use of stochastic techniques or backtracking. The entire 2000 sentences took 0.75 seconds to process. This measurement was the average of several runs on an Lenovo T61 Laptop, running two Intel Core Duo processors, at 2.4 GHz.

5. Conclusions and Future Work

In this paper the LOGICON system was presented and a novel set of semantic structures were described, driven by syntax. The system employs partial parsing techniques using a tagging scheme in which syntax and semantics are closely aligned. With a few partial parsing rules, reasonable accuracy was obtained. Future work will involve refining the parser using a more accurate part-of-speech tagger such as SVMTool [7] and then applying the system to the question-answering domain.

6. Appendix: Parsing Algorithm

The algorithm used by the LOGICON partial parser is shown in pseudo-code below:

- create initial nodes from part-of-speech tags
- repeat until no rule applies:
  - for each parser rule R:
    - repeat while R applies to existing top-level nodes:
      - use R to create a new top-level node

In the training corpus of 1500 sentences, there were a total of 6722 constituent nodes. The second column in table 5 shows the number of nodes generated by each parser rule. The fourth column shows the cumulative percentage. With a few general rules reasonable accuracy can be achieved, but producing increased accuracy from the parser requires writing a larger number of specialized rules.

Table 5. Top 10 most common partial parsing rules

Parser rule	Nodes	%	Cum. %
C → N+	2104	31.30	31.30
C → C + Q	1104	16.42	47.72
C → named entity	1049	15.61	63.33
EV → contains V	761	11.32	74.65
LN → contains AUX + C	550	8.18	82.83
C → C + P + C	362	5.39	88.22
V → P + V	176	2.62	90.84
LN → contains AUX	138	2.05	92.89
C → C + POS + C	96	1.43	94.32
AUX → AUX + NEG	81	1.20	95.52

7. References

[1] S. Abney. Partial Parsing via Finite-State Cascades, in Workshop on Robust Parsing (ESSLLI), 1996.

[2] S. Abney. The SCOL Manual, http://www.vinartus.net/spa, 1997.

[3] C. Baker, C. Fillmore, J. Lowe. The Berkeley FrameNet project, in Proceedings of COLING/ACL, 1998.

[4] D. Bikel. Intricacies of Collins' Parsing Model, Computational Linguistics, 30(4), 2004.

[5] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging, Computational linguistics, 1995.

[6] J. Chen, A. Diekema, et al. Question answering: CNLP at the TREC-10 question answering track. In Proceedings of the Tenth Text REtrieval Conference, 2001.

[7] J. Gimenez, L. Marquez. SVMTool: A general POS tagger generator based on Support Vector Machines. In Proceedings of the 4th International Conference on Language Resources and Evaluation, 2004.

[8] M. Palmer, G. Daniel, K, Paul. The proposition bank: An annotated corpus of semantic roles, Computational Linguistics, 31(1), 2005.

[9] A. Ramsay, H. Seville. Models and Discourse Models, Journal of Language and Computation, 1(2), 2000.

[10] D. Sleator, D. Temperley. Parsing English with a Link Grammar, Third International Workshop on Parsing Technologies, 1993.

[11] R. Wallace. The Annotated A.L.I.C.E. AIML, http://www.alicebot.org/aiml/aaa, 2007