SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

Dependency Based Logical Form Transformations

Stephen Anthony and Jon Patrick
School of Information Technologies
The University of Sydney
Sydney, Australia 2006
{stephen,jonpat}@it.usyd.edu.au

Abstract

This paper describes a system developed for the transformation of English sentences into a first order logical form representation. The methodology is centered on the use of a dependency grammar based parser. We demonstrate the suitability of applying a dependency parser based solution to the given task and in turn explain some of the limitations and challenges involved when using such an approach. The efficiencies and deficiencies of our approach are discussed as well as considerations for further enhancements.

1 Introduction

In addition to the well-known *all words* and *lexical sample* tasks deployed in previous Senseval workshops a number of new tasks have been included in this sense evaluation. These new tasks include identification of semantic roles as in FrameNet (Gildea and Jurafsky 2002), disambiguation of WordNet glosses (Miller 1990; Fellbaum 1998; Harabagiu, Miller et al. 1999), automatic acquisition of subcategorisation frames (Korhonen 2002; Preiss and Korhonen 2002), and Logical Form Identification (LFI) (Rus 2002; Rus and Moldovan 2002). This paper discusses a solution developed for the LFI task. The approach used here employs a functional dependency parser (Järvinen and Tapanainen 1997; Tapanainen and Järvinen 1997) and uses a limited number of additional resources. This contribution is intended to demonstrate the suitability of a dependency parser to the given task and also explain some of the limitations and challenges involved when using such an approach.

1.1 Motivation

Part of the initial step towards the interpretation of a sentence as postulated by Hobbs et al. (1993) involves the proof of the logical form of a sentence. This statement entails the transformation of a sentence into a logical form as a fundamental building block towards *sentence interpretation*.

Advantages specifically related to the utilisation of logical forms in language processing include a simplified interface between syntax and semantics, a natural and easily exploitable representation of syntactic arguments, and the potential for formation of *conceptual predicates* (Rus 2002) if predicates are disambiguated with respect to a general ontology such as WordNet.

1.2 Task Description

The Logical Form (LF) employed in this task is a flat, scope-free first order logic representation that embeds lexical and syntactic information. A predicate is generated for every nominal, verbal, adjectival, and adverbial content word. The name of the predicate is a concatenation of the lemmatised word form and part-of-speech category. The sentence below is followed by its corresponding target *logical form* representation.

Some students like to study in the mornings.

student:n_ (x3) like:v_ (e4, x3, e6) to (e5, e6)
study:v_ (e6, x3, x9) in (e7, x9) morning:n_ (x9).

Relationships between predicates are shared through their arguments. The two types of arguments used are events (e) and entities (x). Using the transformation shown above as an example, the event predicate ‘like’ is labeled as e4 and has subject argument x3 which corresponds to ‘student’ and grammatical object argument e6 which corresponds to the ‘study’ event predicate.

The remainder of the argument slots are reserved for indirect and prepositional objects. Determiners, plurals, negation, auxiliaries, verb tenses, and punctuation are excluded from the final representation.

2 Methodology

The system is built using a highly modular design and is intended to be as generic and reusable as possible. The basic data structure is a flat list-like representation with generic property slots attached to each element. This structure maximises compatibility with the final representation and allows for greater flexibility in the types of information that may be

associated with each predicate. Figure 1 illustrates the major processing modules available and the work flow.

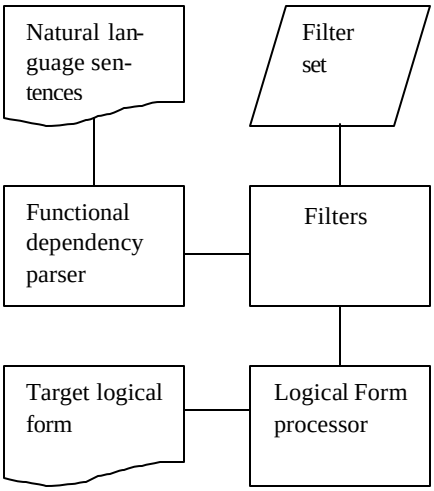

Figure 1: Logical form identification work flow

A syntactic parse including functional dependencies is produced on a per sentence basis. Definitions of the properties associated with each token are presented in Table 1.

Attribute	Value
Word ID	Integer sentence position
Head ID	Integer position of head dependency
Text	The original word form
Lemma	Lemmatised word form
Morpho	Morphological function tags. Parts of speech and sub-features
Syntax	Surface syntactic tags
Depend	Dependency functions
MAIN	Main element
SUBJ	Position of syntactic subject
OBJ	Syntactic object position
I-OBJ	Indirect object position
COMP	Position of syntactic complement
PCOMP	Prepositional complement position
DET	Determiner dependent
ATTR	Attributive nominal
CC	Coordinating conjunction
GOAL	Position of goal
OC	Object complement

Table 1: Linguistic information stored for each token

The resultant parse is transformed into a linear data structure indexed by word position. This is illustrated in Table 2 using the example sentence 'Some students like to study in the mornings'. The original token text is stored, as is the lemmatised form.

WordID	Lemma	HeadID	Depend	Text	Morpho	Syntax
2	some	3	det	Some	DET	>N
3	student	4	subj	students	N NOM PL	NH
4	like	1	main	like	V PRES	VA
5	to	6	pm	to	INF MARK	AUX
6	study	4	obj	study	V INF	VA
7	in	6	tmp	in	PREP	EH
8	the	9	det	the	DET	>N
9	morning	7	pcomp	mornings	N NOM PL	NH

Table 2: Example syntactic parse

Head and dependency type are the most important class of information used by the system. The dependency type and head of the token is often directly, if not indirectly, translatable into a predicate argument. Examples of the types of dependency functions employed include subject, object, prepositional complement, agent, subject and object complements, indirect object, goal, and coordinating conjunctions. Determiner and negator functions are also of interest because they are excluded from the final representation.

The filter module moderates the presence or absence of tokens using stop lists or pass lists or a combination of both. Stop lists are used to specify content to be excluded from the token stream and pass lists specify elements that should remain. Tokens may be filtered from the stream based on any attribute type and value listed in Table 1. This information is provided in the filter set. The principal types of information filtered in this system are determiners based on morphological tags and auxiliaries based on syntactic tag information. For example 'some' and 'the' are filtered as a consequence of a morpho property equals 'DET' stop list rule.

When the token stream has been annotated with the necessary information and has passed through the filter, the tokens that remain are passed through the logical form processor (LFP). The main function of the LFP is to build an inverted index identifying all dependent tokens. Once grammatical dependencies are assigned and the inverted index is built the logical form representation may be constructed. Each predicate is constructed from the token stream in turn based on the part-of-speech category of the token. The base form of the token is concatenated with the part-of-speech tag. A mapping table is used to transform the part-of-speech information pro-

duced by the parse into the coarser grained WordNet tags.

Entities are the simplest type of predicate to construct as they contain only a single argument, for which the word identifier attribute value is used. Noun tokens 'student' and 'morning' from the example are transformed into the predicates student:n_(x3) and morning:n_(x9). Pronouns, prepositional complements, and coordinating conjunctions are dealt with individually using their respective dependency function values.

Adjectives are constructed using the head dependency value as the argument unless the dependent is marked with a subject. In this case the argument becomes the head of the subject. Adverbs are created primarily using the dependency function alone.

Verbal predicates are constructed using SUBJ, OBJ, GOAL, OC, IOBJ, COMP, and PCOMP dependencies in the specified order. A special case exists for verbs that have object complement dependencies. In these cases attributive nominals are identified and assigned as arguments independently.

The main verb 'like' in our example is transformed into the predicate like:v_(e4, x3, e6) as a result of subject (SUBJ) and object (OBJ) dependencies found in 'student' and 'study' respectively. Given the fact that we are dealing with the main verb, the LFP inverts the subject and object dependencies, inserts them into the head verb token property slot and assigns their respective word identifier values. The inverted properties augment the token slot for 'like' which has word identifier four in Table 2. The additional elements of the inverted index used to build the predicate are listed in Table 3.

Attribute	Value
OBJ	6
SUBJ	3
depend head	main 1
lemma	like
morpho	V PRES
syntax	VA
text	like

Table 3: Augmented token slot for 'like'

Verbal predicates which also serve as grammatical objects also warrant special treatment. The token 'study' is an example of this as it serves as the object of the head verb 'like'. A cache is used to store the sentential head, prepositional complements, subjects, and coordinating conjunctions. The cache is used in this instance to assign the subject and prepositional complement arguments in order to form the predicate study:v_(e6, x3, x9). Notice from Table 2 word identifier three matches the grammatical subject token 'students' and word identifier nine matches the head of the prepositional phrase 'in the mornings'.

Once all tokens are processed the logical form transformation is complete and the final representation is presented in the aforementioned notation.

3 Evaluation

Argument, predicate, and sentence level precision and recall measures are used to evaluate performance of the system as compared to a gold-standard. The system was trained on a set of 50 sentences with corresponding logical forms. Final testing was performed on a set of 300 LF-sentence pairs.

3.1 Argument Level

Precision at the argument level is defined to be the number of correctly identified arguments divided by the number of all identified arguments. Recall is defined to be the number of correctly identified arguments divided by the real number of arguments that should be present in the target transformation.

3.2 Predicate Level

Predicates must identify all arguments correctly to be counted as a correct predicate. Precision is defined to be the number of correctly identified predicates divided by the number of all attempted predicates. Recall is defined as the number of correctly identified predicates divided by the real number of predicates that were supposed to be identified in the target transformation.

3.3 Sentence Level

Various other sentence level measures are also used. Sentence-argument is defined as the number of sentences that have all arguments correctly identified divided by the number of sentences attempted. Sentence-predicate is similar except conditioned on predicates. Sentence-argument-predicate is defined to be the number of sentences that have all arguments correctly identified divided by the number of sentences which have all predicates correctly identified. Sentence-argument-predicate-sentences refers to the number of sentences that have all arguments and all predicates correctly identified divided by the number of sentences attempted.

4 Results

As stated earlier the final evaluation was conducted on a set of 300 sentence-LF pairs. Table 4 lists the evaluation precision and recall results using

the measures discussed in section 3 which have been converted into percentages.

Evaluation Measure	Score
Argument Precision	76.4
Argument Recall	65.6
Predicate Precision	84.0
Predicate Recall	85.0
Sentence-Argument	16.0
Sentence-Predicate	35.3
Sentence-Argument-Predicate	38.7
Sentence-Argument-Predicate-Sentences	13.7

Table 4: Evaluation results as percentages

The major source of error in terms of arguments originated from the parser's inappropriate handling of coordinating conjunctions. Another common source of error arose from poor handling of nominal group complexes. With regard to predicate performance, the decision to forfeit the use of the available multi-word item list proved costly.

5 Future Work

Harabagiu et al. (1999) proposed a scheme for attaching sense tags to predicates within the framework of transforming WordNet glosses into a logical form. In this way conceptual predicates may be formed to manipulate a meaning representation in more significant ways. Naturally the sense inventory must be sensitive enough to allow for a meaningful and representative mutation to be applied to the meaning representation.

6 Conclusions

Dependency grammars provide a natural and intuitive solution to the task of logical form identification. We have managed to demonstrate relatively good overall performance on the given task with minimal additional processing and a very small amount of training data.

It is argued that a dependency grammar based parse provides a rich source of knowledge that is suitable for the transformation of English sentences into a logical form. It would appear that there is to a large extent enough information embedded within the parser's output to achieve the desired outcome. It is however apparent that other types of information could further improve the solution. These types of information include named entity recognition and multi-word phrase detection.

References

Fellbaum, C. (1998). *WordNet : An Electronic Lexical Database*. Cambridge, Massachusetts; London, MIT Press.

Gildea, D. and D. Jurafsky (2002). "Automatic Labeling of Semantic Roles." *Computational Linguistics* 28(3): 245-288.

Harabagiu, S. M., G. A. Miller, et al. (1999). *WordNet 2 - A Morphologically and Semantically Enhanced Resource*. SIGLEX.

Hobbs, J. R., M. E. Stickel, et al. (1993). "Interpretation as Abduction." *Artificial Intelligence* 63: 69-142.

Järvinen, T. and P. Tapanainen (1997). *Dependency Parser for English*. Helsinki, University of Helsinki, Department of General Linguistics.

Korhonen, A. (2002). *Subcategorization Acquisition*. Ph.D. Dissertation. Computer Laboratory, University of Cambridge.

Miller, G. (1990). "WordNet: An Online Lexical Database." *International Journal of Lexicography* 3(4).

Preiss, J. and A. Korhonen (2002). *Improving Subcategorization Acquisition with WSD*. In Proceedings of the ACL Workshop on Word Sense Disambiguation: Recent Successes and Future Directions.

Rus, V. (2002). *Logic Form For WordNet Glosses*. Ph.D. Dissertation. Computer Science Department, School of Engineering, Southern Methodist University.

Rus, V. and D. I. Moldovan (2002). "High Precision Logic Form Transformation." *International Journal on Artificial Intelligence Tools* 11(3).

Tapanainen, P. and T. Järvinen (1997). *A Non-Projective Dependency Parser*. In Proceedings of the 5th Conference on Applied Natural Language Processing, Association for Computational Linguistics, Washington D.C.