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

Senseval 3 Logic Forms: A System and Possible Improvements

Altaf Mohammed, Dan Moldovan, and Paul Parker
Language Computer Corporation
Richardson, TX 75080
{altaf, moldovan, parker}@languagecomputer.com

Abstract

Logic Forms, particular powerful logic representations presented in Moldovan and Rus (2001), are simple yet highly effective. In this paper, the structure of Logic Forms and their generation from input text are described. The results of an evaluation comparing the Logic Forms generated by hand with those generated automatically are also reported. Finally, we suggest some improvements to the representation used in the LFI task based on our results.

1 Introduction

Logic Forms are first order logic representations of natural language text. The notation is very close to the natural language. A Logic Form is a collection of predicate instances derived from text. A detailed description of the notation is presented in Moldovan and Rus (2001).

Logic Forms can be utilized by a wide variety of applications. A Logic Prover (Rus, 2002; Moldovan et al., 2003) utilizing the axioms generated by the Logic Form generation system boosts the performance of the Question Answering system. The Prover essentially takes as input the Logic Forms of the question and one or more answers and then proceeds to justify (and rank) the answers based on (i) world knowledge axioms, and (ii) NLP axioms. The Logic Prover developed at Language Computer Corporation has increased the performance of the QA system by 30%.

2 Automatic Generation of Logic Forms

2.1 Parse Tree Construction

Logic Forms are derived from the output of a syntactic parser. The first step is the identification of word collocations (based on those identified by WordNet (Miller, 1995)). The parser then proceeds to identify (i) parts of speech of individual words, and (ii) syntactic structure of the text, based on grammar rules. It also differentiates active verb constructs from passive ones. The output is a parse tree. We also include in the parse tree (i) word senses, based on WordNet, and (ii) named-entity tags (Surdeanu and Harabagiu, 2002). Named-entity tags are tags associated with a word or group of words, indicating that they belong to a particular category, for instance, currency, time, date, place, human, etc. The word senses from the parse tree are simply included in the Logic Form as is (for subsequent use in applications), while the named-entity tags are additionally used in the generation of Logic Forms.

2.2 Logic Form Generation

First we identify independent arguments (those arguments which are generated anew for certain predicates). These include arguments for nouns,

verbs (action/eventuality only), and compound nouns and coordinating conjunctions (for both of these, only the result argument (or the first one)). Independent arguments are also generated for certain adjectival phrases and/or determiners in the rare cases when they do not qualify a noun phrase, but stand all by themselves. The independent arguments, once generated, are propagated up the parse tree (as predicates form the leaves of the parse tree). In this way, heads of phrases are marked.

Next comes the identification of dependent arguments (those which are derived from other independent and/or dependent arguments). These may include arguments for modifiers (adjectives, adverbs), secondary verb slots (all except the first) and secondary coordinating conjunction slots (all except the resulting argument), or linking words (prepositions, subordinating conjunctions, etc). The derivation of these arguments follows from a slot-filling approach and is based on the interpretation of the parse tree structure and the associated transformation rule (Moldovan and Rus, 2001). This is a rule that says how a particular parse tree structure must be handled, for instance, 'S -> NP VP' says that the subject of the main/action verb of VP is the head of phrase of NP.

3 Dealing with Ambiguous Structures

Named-entity tags are helpful when parsing certain ambiguous structures. Take, for instance, the following two sentences:

(i) They gave the visiting team a heavy loss.
(ii) They played football every evening.

The grammar rule for the verb phrase in both sentences is 'VP -> VB NP NP'.

Whereas, in sentence (i), the first NP is the indirect object and the second one the direct, exactly the converse is true for sentence (ii). Upon closer examination, it is found that 'every evening' being an indicator of time, does not qualify for the position of the direct object.

The named-entity recognition system marks all such indicators of time/date. This enables us to disqualify these noun phrases as candidates for the position of the direct object of a verb.

The aforementioned is just one kind of ambiguity that we have addressed. Another kind of ambiguity that we encountered (but have not implemented a solution for yet) is the reduction of certain words to base forms. Consider, for instance,

(i) John **found** the key. *('find' in VBN form)*
(ii) The King promised to **found** a similar institution. *('found' in VB form)*

A possible solution we considered was looking at part-of-speech tags (VBN vs. VB) to resolve ambiguity, but have not pursued this further in the (current) absence of a database that maintains a mapping from inflected word and part-of-speech pairs to the corresponding base forms.

Another approach considered was choosing the base form whose frequency of occurrence in the Brown corpus, as reflected in WordNet, was highest.

4 Changes/Improvements for Senseval 3

Since no complete specification was given for the proper formation of logic forms for many special cases, we chose to model our Senseval 3 Logic Form system on the provided examples. The LF system was updated to model the Senseval 3 behavior in the following ways.

4.1 Adverbs Modifying Adjectives

These adverbs are assigned the same argument as the adjective they modify (Mohammed, 2003). For instance, *"the extremely fast athlete"* is represented as *"extremely:r_ **(x1)** fast:a_ **(x1)** athlete:n_ (x1)"*.

4.2 Variable Slots for Verbs

The verbs are now given a variable number of arguments (minimum two: the action/eventuality

and the subject). They get arguments for all verb objects, including prepositional attachments.

Previously, we had a fixed slot allocation mechanism for verbs, specifying always the action, the subject, and the direct object. These slots were filled with dummy arguments in the absence of proper arguments.

Example:
S: John gave Mary the book on Saturday.

LF (previous notation)
John:n_ (x1) give:v_ (e1, x1, x3) Mary:n_ (x2) book:n_ (x3) on (e1, x4) Saturday:n_ (x4)

LF (Senseval 3 notation)
John:n_ (x1) give:v_ (e1, x1, x3, ***x2, x4****) Mary:n_ (x2) book:n_ (x3) on (e1, x4) Saturday:n_ (x4)*

4.3 Subordinating Conjunctions

These conjunctions are given two arguments. The second argument is the main/action verb of the subordinate clause. The first argument is assigned as follows: (i) if the clause attaches to a sentence (or a verb phrase), then the main/action verb of this sentence (or verb phrase), (ii) if the clause attaches to a noun phrase, then the head of the noun phrase. Additional details are presented in Mohammed (2003).

Example:
If you heat ice, it melts.

LF:
If ***(e2, e1)*** *you (x1) heat:v_ (****e1****, x1, x2) ice:n_ (x2) it (x3) melt:v_ (****e2****, x3)*

5 Impact of Parse Tree Accuracy on Logic Forms

The Logic Forms are derived directly from the parse trees. This makes the generation of accurate parse trees extremely important. We have analyzed the performance of automatically generated Logic Forms based on both the machine-generated (hence necessarily somewhat erroneous) parse trees and parse trees generated by human annotators.

The following results are based on the set of 300 test sentences provided for the Logic Forms Identification task at Senseval 3. The number of sentences with all predicates correctly identified has increased from 155 to 191, an improvement of 23.2%. The number of sentences with all correct arguments and all correct predicates (in other words, 100% correct Logic Forms) has increased from 65 to 98, a 50.7% improvement over Logic Forms derived from machine-generated parse trees. The results are presented in Table 1. Note that the row captioned "Predicates" indicates the number of sentences for which all predicates were correctly identified, while the row captioned "Entire LF" indicates those for which all predicates as well as all associated arguments were correctly identified.

	Machine Parse	***Hand Parse***	***Improvement***
Predicates	155 / 300	191 / 300	23.2%
Entire LF	65 / 300	98 / 300	50.7%

Table 1: Performance of LF Generation System

6 Recommendations

The sample data provided for the LFI task was used as the model for expected system behavior. A few recommendations that we believe would be improvements are below. Note that neither space nor time permitted an extensive consideration of other alternatives.

6.1 Possessive Pronouns

Possessive pronouns are treated as mere adjectives in the sample data. A better way would be to handle these as any other possession indicator, and thus treat them as two-argument predicates.

Example:
S: John drives his car.

LF (Sample data):
John:n_ (x1) drive:v_ (e1, x1, x2) ***his (x2)*** *car:n_ (x2)*

LF (recommendation):
John:n_ (x1) drive:v_ (e1, x1, x2) ***his (x2, x1)*** *car:n_ (x2)*

6.2 Hybrid Verb Slot Representation

A verb's slots are supposed to signify their relation to the verb. The first slot is always reserved for the action/eventuality expressed by the verb. The second, then, is always for the subject of the verb. Now, the third slot should be *reserved* for the direct object of the verb (if any), and then, additional slots should be filled if and only if there are indirect objects associated with the verb.

We propose using either dummy or null arguments for certain slots. For instance, for a verb that has only indirect objects (apart from a subject), the representation for the verb can be, for instance,

run:v_ (e1, x3, ***0****, x4, x6, ...)*, or
run:v_ (e1, x3, ***x9****, x4, x6, ...)*, where 'x9' is a dummy argument that is not referenced anywhere else in the Logic Form.

Moreover, the inclusion of prepositional attachments in the verb slots is a kind of redundancy that should be avoided. The following examples will make this proposal clear.

S: *John plays at the park.*
LF: *John:n_ (x1) play:v_ (e1, x1) at (e1, x2) park:n_ (x2)*

S: *John plays every day at the park.*
LF: *John:n_ (x1) play:v_ (e1, x1, 0, x2) every:a_ (x2) day:n_ (x2) at (e1, x3) park:n_ (x3)*

S: *John plays tennis every day at the park.*
LF: *John:n_ (x1) play:v_ (e1, x1, x2, x3) tennis:n_ (x2) every:a_ (x3) day:n_ (x3) at (e1, x4) park:n_ (x4)*

S: *John gives Mary the book every evening in the library.*
LF: *John:n_ (x1) give:v_ (e1, x1, x3, x2, x4) Mary:n_ (x2) book:n_ (x3) every:a_ (x4) evening:n_ (x4) in (e1, x5) library:n_ (x5)*

Note that none of the examples include the arguments for 'park' or 'library' in the slots for verbs. Since these predicates are already connected to the verbs via prepositions, it is unnecessary to also include them in verb slots.

References

George A. Miller. 1995. *WordNet: A lexical database for English.* In Communications of the ACM, pages 39-41.

Altaf Mohammed. 2003. *Logic Form Transformation of WordNet Glosses.* Master's dissertation, University of Texas at Dallas, Richardson, TX.

Dan I. Moldovan and Vasile Rus. 2001. *Logic Form Transformation of WordNet and its Applicability to Question Answering.* In Proceedings of the ACL 2001 Conference, July 2001, Toulouse, France.

Dan Moldovan et al. 2003. *COGEX: A Logic Prover for Question Answering.* In Proceedings of the Human Language Technology Conference.

Vasile Rus. 2002. *Logic Forms for WordNet Glosses.* Ph.D. dissertation, Southern Methodist University, Dallas, TX.

Mihai Surdeanu and Sanda Harabagiu. 2002. *Infrastructure for Open-Domain Information Extraction.* In Proceedings of the Human Language Technology Conference (HLT 2002): 325-330.

Well-formed Formulas (WFFs) of Propositional Logic. Propositional logic uses a symbolic â€œlanguageâ€ to represent the logical structure, or form, of a compound proposition. Like any language, this symbolic language has rules of syntaxâ€”grammatical rules for putting symbols together in the right way. Any expression that obeys the syntactic rules of propositional logic is called a well-formed formula, or WFF. Fortunately, the syntax of propositional logic is easy to learn. It has only three rules: Any capital letter by itself is a WFF. Any WFF can be prefixed with â€œ~â€. Given past system responses in the SENSEVAL-3 English all-words task, we can evaluate past systems on the same corpus, but using the coarse-grained sense hierarchy provided by our sense-clustered taxonomy. We may then compare the scores of each system on the coarse-grained task against their scores given a random clustering at the same resolution.Â 2004). A guess by a system is given full credit if it was either the correct answer or if it was in the same cluster as the correct answer. Em- ploying an inductive logic programming algorithm, the approach generates expres- sive disambiguation rules which exploit several knowledge sources and can also model relations between them. The ap- proach is evaluated in two tasks: identifica- tion of the correct translation for a set of highly ambiguous verbs in English- Portuguese translation and disambiguation of verbs from the Senseval-3 lexical sam- ple task.Â We also experiment with a monolingual task by using the verbs from Senseval-3 lexical sample task. We explore knowledge from 12 syntactic, semantic and pragmatic sources.Â Table 1. Verbs and possible senses in our corpus. Both corpora were lemmatized and part-of-speech (POS) tagged using Minipar (Lin, 1993) and. 44. We evaluate our system on a set of ve benchmark datasets, Senseval-2, Senseval-3, SemEval-2007, SemEval-2013 and SenEval-2015 and show that the proposed model outperforms state-of-the-art knowledge-based WSD system. 2 Related Work. Lesk (Lesk 1986) is a classical knowledge-based WSD algorithm which disambiguates a word by selecting a sense whose denition overlaps the most with the words in its con-text.Â Most WSD systems use a sense repository to obtain a set of possible senses for each word. WordNet is a comprehensive lexical database for the English language (Miller 1995), and is commonly used as the sense repository in WSD systems. To negate a statement of the form "If A, then B" we should replace it with the statement "A and Not B". This might seem confusing at first, so let's take a look at a simple example to help understand why this is the right thing to do. Consider the statement "If I am rich, then I am happy." For this statement to be false, I would need to be rich and not happy. If A is the statement "I am rich" and B is the statement "I am happy,", then the negation of "A $\Rightarrow$ B" is "I am rich" = A, and "I am not happy"