Transformation of WordNet Glosses into Logic Forms

Dan I. Moldovan and **Vasile Rus**
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122
{moldovan, vasile}@seas.smu.edu

Abstract

This paper presents a method and implementation results for the transformation of WordNet glosses into logic forms. The glosses, currently expressed in English are a rich source of world knowledge. Logic forms are useful for logic proofs, inference, and many other AI applications. We demonstrate their applicability to a Question Answering system.

Introduction

It is well known that logic form representations of natural language text help applications such as text inference(Moldovan *et al.* 1999), abductive reasoning (Hobbs, Stickel, & Martin 1993), Question/Answering (Moldovan *et al.* 1999) and others.

Consider the TREC-QA's question (NIST 2000):

$Q198$: *How did Socrates die*?

The answer to this question appears in the text "...Socrates' death came when he chose to drink poisoned wine...". To prove that this is a plausible answer one needs extra knowledge. From WordNet (Miller 1995) the gloss of *poison:v#2* (the second sense of verb poison) we have *kill with poison* and from the first sense of verb *kill:#1* we have *cause to die*, which justifies the answers.

This paper presents a logic notation suitable for representing the English texts in the WordNet glosses. Since the automatic Logic Form Transformation (LFT) for open text requires an exceedingly large number of rules, we were faced with a coverage problem: to select those transformation rules that have the broadest impact - i.e. transform more glosses than other rules. An iterative procedure is presented that picks up such rules.

Logic Form Transformation for WordNet glosses

After glosses are syntactically parsed the next step is to transform them into a more abstract logical representation. The logic form is an intermediary step between syntactic parse and the deep semantic form - which we do not address here. The LFT codification acknowledges syntax-based relationships such as: (1) syntactic subjects, (2) syntactic objects, (3) prepositional attachments, (4) complex nominals, and (5) adjectival/adverbial adjuncts.

Approach to implement Logical Form Transformations (LFTs)

There are two criteria that guide our approach: (1) the notation to be as close as possible to English, and (2) the notation should be syntactically simple. Our approach is to derive the *LFT directly from the output of the syntactic parser.* The parser resolves the structural and syntactic ambiguities. This way, we avoid the very hard problems of logic representation of natural language. We follow closely the successful representation used by Hobbs in TACITUS (Hobbs 1986). Hobbs explains that for many linguistic applications it is acceptable to relax ontological scruples, intricate syntactic explanations, and the desire for efficient deductions in favor of a simpler notation closer to English.

For the logic representation of WordNet glosses we ignore: plurals and sets, verb tenses, auxiliary verbs, quantifiers and modal operators, comparatives and negation. This decision is based on our desire to provide an acceptable and consistent logic representation that otherwise would be unfeasible.

LFT Definitions

Predicates

A predicate is generated for every noun, verb, adjective or adverb encountered in any gloss. The name of the predicate is a concatenation of the morpheme's base form, the part-of-speech and the WordNet semantic sense, thus capturing the full lexical and semantic disambiguation. For example, the LFT of the gloss of {student, pupil, educatee} is (a learner who is enrolled in an educational institution). It will contain the predicates learner:n, enroll:v and educational_institution:n.

Fix slot-allocation

In the spirit of the Davidsonian treatment of the action predicates (Davidson 1967), all verb predicates (as

well as the nominalizations representing actions, events or states) have three arguments: action/state/event-predicate(e,x_1,x_2), where:
• e represents the *eventuality* of the action, state or event i stated by the verb to take place,
• x_1 represents the *syntactic subject* of the action, event or state, and
• x_2 represents the *syntactic direct object* of the action, event or state.
For example, the LFT of (a person who backs a politician), the gloss of {supporter, protagonist, champion, admirer, booster, friend} is: [person:n(x_1) & back:v(e_1,x_1,x_2) & politician:n(x_2)]. Several clarifications are in order here.
(a) In case when the predicate is a ditransitive verb, its representation is verb(e,x_1,x_2,x_3). For example: professor gives students the grades is represented as: professor(x_1) & give(e_1,x_1,x_2,x_3) & grade(x_2) & student(x_3). This condition is detected by the presence of two noun phrases following a verb in active voice.
(b) The arguments of verb predicates are always in the order: subject, direct object, indirect object. In the case when one of these syntactic roles is missing, its respective argument appears under the verb predicate, but that argument will not be used by any other predicate. This is a so-called *"slot-allocation"* representation since the position of the arguments is fixed for the purpose of a simpler notation. Since in WordNet glosses not many verbs have indirect objects, the argument x_3 is used only when necessary, otherwise is omited. However, the arguments for the subjects and direct objects are always present, even when the verb does not have these syntactic roles. We found that this simple and consistent representation is easy to derive and use.

Modifiers

The role of complements within a phrase is replicated in the LFTs. Predicates generated from modifiers share the same arguments with the predicates corresponding to the phrase heads. Adjective predicates share the same argument as the predicate corresponding to the noun they modify. An exemplification is the LFT of the gloss of {artifact, artifact}, which maps (a man-made object) into [object:n(x_1) & man-made:a(x_1)]. Similarly, the argument of adverbial predicate is the argument marking the eventuality of the event/state/action they modify. For example, the gloss of the verb synset {hare} is (run quickly), producing the LFT = [run(e_1,x_1,x_2) & quickly(e_1)].

Conjunctions

Conjunctions are transformed in predicates, which enable the aggregation of several predicates under the same syntactic role (e.g. subject, object or prepositional object). By convention, conjunction-predicates have a variable number of arguments, since they cover a variable number of predicates. The first argument represents the "result" of the logical operation induced by the conjunction (e.g. a *logical and* in the case of the and conjunction, or a *logical or* in the case of the or conjunction). The rest of the arguments indicate the predicates covered by the conjunction, as they are arguments of those predicates as well. Table 1 provides examples of conjunction predicates.

Prepositions

We also generate predicates for every preposition encountered in the gloss. The preposition predicates always have two arguments: the first argument corresponding to the predicate of the head of the phrase to which prepositional phrase is attached, whereas the second argument corresponds to the prepositional object. This predicative treatment of prepositional attachments was first reported in (Bear & Hobbs 1988). Table 2 shows some examples of preposition predicates.

Complex nominals

Many complex nominals are encoded currently in WordNet as synset entries comprising several words, known as WordNet collocations (e.g. flea market, baseball team, joint venture). Still, many compound nouns are not encoded as WordNet entries, and need to be recognized as a single nominal. The way of doing this was first devised in TACITUS (Hobbs 1986), when the predicate nn was first introduced. Similar to conjunction predicates, the nn predicates can have a variable number of arguments, with the first one representing the result of the aggregation of the nouns corresponding to the rest of the arguments. Examples from Table 3 show the transformation of some complex nominals.

LFT Implementation

The implementation of LFTs relies on information provided by the syntactic parser. We have developed a set of *transformation rules* that create predicates and assign them arguments. For every rule of the parser, it ought to be a transformation rule which produces the corresponding logical formula. There are two classes of rules: (1) intra-phrase and (2) inter-phrase transformation rules. The *intra-phrase transformation rules* generate predicates for every noun, verb, adjective or adverb. They also assign the variables that describe dependencies local to the phrase.

The *inter-phrase transformation rules* provide with the arguments of the verb predicates, preposition predicates and inter-phrasal conjunctions. Verb predicate arguments are identified by recognizing the syntactic subject and object of the respective verb, based on a few grammar rules and relative pronoun interpretation. Dependencies between adjectival (adverbial) phrases and noun (verb) phrases are predicted based on vicinity. Both intra- and inter-phrase transformation rules are produced from the parser. Examples of transformation rules are shown in Table 4.

Coverage Issue

One of the most challenging problems in implementing LFT for natural language text is the fact that each grammar rule leads to one or more LFT rules. Table 5

Synset	Gloss	LFT
{enterprise}	(an organization created for business ventures)	organization:n(x_2) & create(e_1,x_1,x_2) & for(e_1,x_3) & nn(x_3,x_4,x_5) & business:n(x_4) & venture:n(x_5)
{tax income, taxation, tax revenue, revenue}	(government income credited to taxation)	nn(x_2,x_3,x_4) & government:n(x_3) & income:n(x_4) & credit:v(e_1,x_1,x_2) & to(e_1,x_5) & taxation:n(x_5)

Table 3: Examples of complex nominal predicates

Intra-phrase transformation rules			
Rule	**Transformation(LFT)**	**Gloss**	**Synset**
ART ADJ_1 ADJ_2 *NOUN*→ *noun*(x_1) & adj_1(x_1) & adj_2(x_1)	return:n(x_1) & hard:a(x_1) & straight:a(x_1)	(a hard straight return (as in tennis or squash))	{drive}
ART ADJ_1 AND ADJ_2 *NOUN*→ *noun*(x_1) & adj_1(x_1) & adj_2(x_1)	light:n(x_1) & weak:a(x_1) & tremulous:a(x_1)	(a weak and tremulous light)	{shimmer, play}
VERB ADV→ *verb*(e_1,x_1,x_2) & *adv*(e_1)	cut:v(e_1,x_1,x_2) & open:r(e_1)	(cut open)	{slash, gash}
ART $NOUN_1$ *'S* $NOUN_2$→ $noun_2$(x_1) & $noun_1$(x_2) & *pos*(x_1,x_2)	body:n(x_1) & person:n(x_2) & pos(x_1,x_2)	(a person's body)	{body}
Inter-phrase transformation rules			
Rule	**Transformation**	**Gloss**	**Synset**
VP_1 *CONJ* VP_2 *PREP NP*→ *conj*(e_1,e_2, e_3) & LFT(VP_1(e_2,x_1,x_2)) & LFT(VP_2(e_3,x_1,x_2)) & *prep*(e_1,x_3) & LFT(*NP*)	or(e_1,e_2, e_3) & keep:v(e_2,x_1,x_2)) & maintain:v(e_3,x_2,x_2)) & in(e_1,x_3) & condition:n(x_3) & unaltered:a(x_3)	(keep or maintain in unaltered condition)	{continue, uphold carry_on, bear_on preserve}
NP_1 *VP by* NP_2 *PREP* NP_3→ LFT(NP_1(x_2)) & LFT(VP(e_1,x_1,x_2)) & LFT(NP_2(x_1)) & *prep*(x_1,x_3) & LFT(NP_3(x_3))	nn(x_2,x_4,x_5) & garment:n(x_4 closure:n(x_6 conceal:v(e_1,x_1,x_2) & fold:n(x_1) & of(x_1,x_3) & cloth:n(x_3)	(a garment closure (zipper or buttons) concealed by a fold of cloth)	{fly, fly front}

Table 4: Examples of LFT rules

stops. The rules in S' are determined based on the frequency of occurrences. The cardinality of S' determines how fast δ falls below τ. A larger value would generate a smaller number of refinement steps but more effort is spent on rules that bring little benefit, especially in the last steps. At the end of step 2 the final set S contains the grammar rules most representative for the target corpus.

Experiments and Results

To validate our procedure we experimented it on a subset of WordNet 1.6 noun glosses.

The initial set of rules is formed by taking the most frequent rules for each grammar phrase detected in a corpus of 10,000 noun glosses randomly selected from the noun data file of WordNet 1.6. We tag and parse them. From the parse trees obtained we extract all grammar rules and their number of occurrences and sort them according to their frequency. Then, we select the most frequent rules up to the point where the gain in coverage is less than 1%. At the end of the first step we have a set of most frequent rules.

To test the overall performance of the rules found in step one we built a corpus of 400 noun glosses from the *artifact* hierarchy. We expanded the glosses' definitions, ran Brill's tagger, corrected the tags and parsed them. Finally, the LFT for each expanded gloss was done manually. We run the set of rules S_0 (seventy rules) on our test data and compared the output with the LFTs obtained by hand. A coverage of 72.5% was achieved. Then, we applied the procedure in step two to boost up the performance. The initial set of rules S_0 is extended at each refinement iteration with a fixed number of rules (cardinality of S'): i.e. three for each grammar phrase from the training corpus of 10,000 glosses. At each iteration the gain in coverage over the test data is determined. The chart in Figure 1 shows the evolution of the improvement in coverage δ achieved as the number of refinement iterations increases. We stop at iteration 8 when the coverage is 81% and the gain obtained with S at the last iteration on the 400 glosses was less than the established threshold. The results obtained are encouraging. One can get a better performance by adjusting the two parameters τ and the cardinality of S' at the cost of a larger effort.

Applications

The LFT can help a Question Answering system that we built (Moldovan *et al.* 1999) with providing better answers.

Synset	Gloss	LFT
{masterstroke}	(an achievement demonstrating great skill *or* mastery)	achievement:n(x_1) & demonstrate(e_1,x_1,x_2) & or(x_2,x_3,x_4) & skill:n(x_3) & great:a(x_3) & mastery:n(x_4)
{tumble}	(roll *and* turn skillfully)	and(e_1,e_2,e_3) & roll:v(e_2,x_1,x_2) & turn:v(e_3,x_1,x_2) & skillfully:r(e_1)
{trip, stumble, misstep}	(an unintentional *but* embarrassing blunder)	blunder:n(x_1) & but(x_1,x_2,x_3) & unintentional:a(x_2) & embarrassing:a(x_3)

Table 1: Examples of conjunction predicates

Synset	Gloss	LFT
{demonetize}	(deprive *of* value *for* payment)	deprive:v(e_1,x_1,x_2) & of(e_1,x_3) & value:n(x_3) & for(x_3,x_4) & payment:n(x_4)
{pitching}	(playing the position *of* pitcher *on* a baseball team)	playing:n(e_1,x_1,x_2) & position:n(x_2) & of(x_2,x_3) & pitcher:n(x_3) on(e_1,x_4) & baseball_team:n(x_4)

Table 2: Examples of preposition predicates

Part of speech	Rules
noun	5,392
verb	1,837
adjectives	1,958
adverbs	639
Total	9,826

Table 5: Size of grammar for WordNet glosses per part of speech

shows the number of distinct grammar rules retrieved from parse trees of all glosses in WordNet. Since the LFT rules are generated manually, it becomes important to identify those that have the most coverage. The total number of nearly 10,000 rules (see Table 5) that should be implemented is by far too large to possibly be implemented. To deal with this coverage problem we devised a two-steps procedure that works iteratively:

Step 1: implement first the most common grammar rules

Step 2: adjust the performance by applying some patches

The first step is basically an acquisition phase: from a representative corpus of glosses we derive the most common grammar cases and resolve them. The second step consists of deriving incrementally a set of patches to boost up the performance.

We have observed that although the total number of grammar rules is large, a small number of rules for a specific phrase cover a large percentage of all occurrences. This might be explained by the relative structural uniformity of glosses: genus and differentia. This relative uniformity is more pronounced for nouns and verb glosses.

Based on this observation we implement first the most common rules and leave the others uncovered or at least postpone their implementation. No LFT is generated for grammar rules that are not frequent enough. Thus, the problem of precision translates into a coverage problem and vice-versa.

Table 6 shows the distribution of most common phrases for 10,000 randomly selected noun glosses, the number of unique grammar rules and the percentage of phrases covered by the top ten rules. From the table one notices that the coverage of top ten most frequent used rules is above 90%.

To decide whether a rule is common or not we use an empirical criteria: if it brings a gain in coverage greater than a specified threshold then we consider that case to be common. The value of the threshold should be a compromise between the effort spent to implement the rule and the gain in coverage brought by that rule. The value of the threshold in the case of WordNet glosses was empirically established to be 1%. At the end of the first step we have validated a set of most frequent rules S_0. Denote with G the set of rules uncovered yet.

The second step of the procedure consists of applying a set of patches to boost up the performance obtained in the first step. The patches are determined through an iterative refinement process. The processing performed in the second step is outlined in the algorithm below.

procedure LFT(S_0, G, τ) {
i = 0; S = S_0; δ = 0; S' = $\emptyset$;
old_perf = 0; apply(S_0, new_perf);
δ = new_perf - old_perf;
while ((i≤upper_limit) && ($\delta \leq \tau$)) {
S' = most_frequent_rules(G - S');
G = G - S'; .25inS = S $\bigcup$ S';
old_perf = new_perf; apply(S, new_perf);
δ = new_perf - old_perf;
i = i + 1; }
return (new_perf);}

At each refinement step, a set of rules S' is added to the set of rules obtained in previous steps S (the initial set is S_0 obtained in phase one). The gain in coverage δ of this set is computed and compared against a threshold τ: if the value of δ is lower than threshold τ the process

Phrase	Occurrences	Unique rules	Coverage of top ten
base NP	33,643	857	.69
NP	11,408	244	.95
VP	19, 415	450	.70
PP	12,315	40	.99
S	14,740	35	.99

Table 6: Phrases and their coverage of top 10 most frequent rules in 10,000 noun glosses

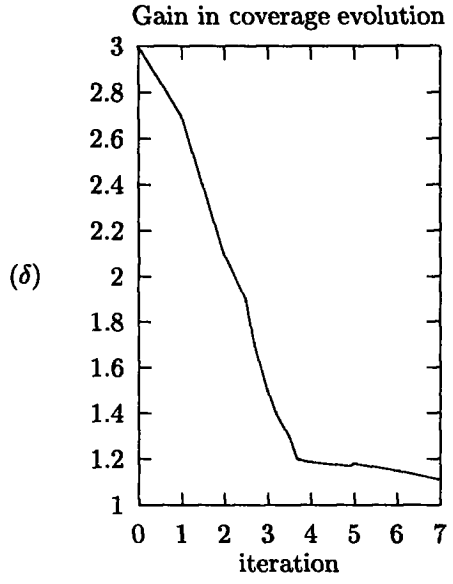

Figure 1: Gain in coverage evolution in phase two

Consider TREC-QA's question (NIST 2000):

*Q*481 : *Who shot Billy the Kid?*

The Q/A system identifies first that the answer type is PERSON. It uses keywords *Billy the Kid* and *shot* to retrieve paragraphs that may contain the answer. Two such paragraphs are:

- P1: The scene called for Phillips ' character to be saved from a lynching when Billy the Kid (Emilio Estevez) shot the rope in half just as he was about to be hanged .
- P2: In 1881 , outlaw William H. Bonney Jr. , alias Billy the Kid , was shot and killed by Sheriff Pat Garrett in Fort Sumner , N.M.

The answer is provided by paragraph P2 and is depicted by the system as follows. Using LFT transformation, the question has a representation of the form: *Q: PERSON(x1) & shot(e, x1, x2) & Billy_the_Kid(x2)* where *x1* is to be instantiated from paragraphs.

The paragraphs have the following LFT (we show only the relevant part):

- P1: Billy_the_Kid($x1'$) & shot($e1'$, $x1'$, $x2'$) & rope($x2'$)
- P2: Billy_the_Kid($x1'$) & shot($e1'$, $x2'$, $x1'$) & Sheriff_Pat_Garrett($x2'$)

The logic prover attempts to prove the question starting from the paragraph. The logic proof for P1 fails because Billy_the_Kid is the agent of shooting, not the object as requested by the question. The logic proof for P2 succeeds and the agent of shooting *Sheriff_Pat_Garrett* unifies with PERSON from the question. The prover yields e1 = $e1'$, $x1 = x2'$ and x2 = $x1'$. Note that our logic form representation based on slot-allocation played a crucial role in this proof.

Conclusions

We presented a logic notation suitable for representing WordNet glosses. A two-step procedure to solve the coverage problem was introduced which obtained an overall coverage of 81% on 400 WordNet glosses. Parser's accuracy and irregularities in some WordNet glosses influenced mainly the error rate in our approach. We showed how the logic form helps in practical applications as Question/Answering.

References

Bear, J., and Hobbs, R. J. 1988. Localizing expression of ambiguity. In *Proceedings of the Second Conference on Applied Natural Language Processing*, 235–242. Association for Computational Linguistic.

Davidson, D. 1967. The logical form of action sentences. In Rescher, N., ed., *The Logic of Decision and Action*. University of Pittsburgh Press. 81–95.

Hobbs, J.; Stickel, M.; and Martin, P. 1993. Interpretation as abduction. *Artificial Intelligence* 63:69–142.

Hobbs, J. R. 1986. Overview of the TACITUS project. *Computational Linquistics* 12(3).

Miller, G. 1995. WordNet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Moldovan, D.; Harabagiu, S.; Pasca, M.; Mihalcea, R.; Goodrum, R.; Girju, R.; and Rus, V. 1999. Lasso: A tool for surfing the answer net. In *Proceedings of the Text Retrieval Conference (TREC-8).*

NIST. 2000. National institute of standards and technology. *TREC - Text RETrieval Conference, http://trec.nist.gov.*