

Logic Form Transformation of WordNet and its Applicability to Question Answering

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

WordNet is a rich source of world knowledge from which formal axioms can be derived. In this paper we present a method for transforming the WordNet glosses into logic forms and further into axioms. The transformation of WordNet glosses into logic forms is useful for theorem proving and other applications. The paper demonstrates the utility of the WordNet axioms in a question answering system to rank and extract answers.

1 Introduction

1.1 Motivation

It is well understood and agreed that world knowledge is necessary for many common sense reasoning problems. In this paper we argue that WordNet is an important source of world knowledge and show how this knowledge can be put to work for open-domain Question Answering systems.

Consider the TREC-QA 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 to know that drinking poisoned wine may be a cause of death. This extra knowledge is found in WordNet glosses (Miller, 1995). The gloss of concept *poison:v#2* (the second sense of verb poison) contains {*kill with poison*} and the first sense of verb *kill:v#1* is {*cause to die*}, which collectively justify the answer.

This paper presents a simple but consistent logic notation suitable for representing the English texts of the WordNet glosses and demonstrates its immediate applicability to Question Answering.

1.2 Research Goal

The goal of this research project is to transform all the WordNet glosses into logic representations

that enables reasoning mechanisms for many practical applications. In this paper we limit the discussion to the definitions and ignore the gloss examples.

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.

The main problems encountered are the selection of an appropriate *logic representation* and actual *implementation* of the rules that transform the English definitions into logic forms. Before the rules are applied, the glosses are passed through a preprocessing phase consisting of tokenization, part-of-speech tagging and syntactic parsing.

1.3 Approach

There are two criteria that guide our approach: (1) the notation be as close as possible to English, and (2) the notation 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 manageable and consistent logic representation that otherwise would be unfeasible. We have not noticed that these simplifications had any adverse effect on the TREC questions.

1.4 Related Work

This work is part of a larger project to extend WordNet outlined in citeHarabagiu99. Our work of processing WordNet glosses resembles previous efforts of extracting lexical information from machine readable dictionaries (MRD), as LDOCE (*Longman Dictionary of Contemporary English*) or Webster’s 2nd International Dictionary (W2). Different parsing methods of dictionary definitions were used: pattern-matching (Chodorow et al., 1985), specially constructed definition parsers (Wilks et al., 1996) or broad coverage parsers (Richardson et al., 1998) (ISI, 1998). All those efforts were limited to extracting genus terms, unlabeled or labeled relations, or build taxonomies (ISI, 1998).

2 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_i, x_1^i, x_2^i), where:

- e_i represents the *eventuality* of the action, state or event i stated by the verb to take place,
- x_1^i represents the *syntactic subject* of the action, event or state, and
- x_2^i 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_i, x_1^i, x_2^i, x_3^i). 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 omitted. 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 but 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

| Synset | Gloss | LFT |
|--------------------------|--|--|
| {masterstroke} | (an achievement demonstrating great skill <i>or</i> mastery) | achievement:n(x_1) & demonstrate(e_1, x_1, x_2) & <u>or</u> (x_2, x_3, x_4) & skill:n(x_3) & great:a(x_3) & mastery:n(x_4) |
| {tumble} | (roll <i>and</i> turn skillfully) | <u>and</u> (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 <i>but</i> embarrassing blunder) | blunder:n(x_1) & <u>but</u> (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 <i>of</i> value <i>for</i> payment) | deprive:v(e_1, x_1, x_2) & <u>of</u> (e_1, x_3) & value:n(x_3) & <u>for</u> (x_3, x_4) & payment:n(x_4) |
| {pitching} | (playing the position <i>of</i> pitcher <i>on</i> a baseball team) | playing:n(e_1, x_1, x_2) & position:n(x_2) & <u>of</u> (x_2, x_3) & pitcher:n(x_3) & <u>on</u> (e_1, x_4) & baseball_team:n(x_4) |

Table 2: Examples of preposition predicates

to the prepositional object. This predicative treatment of prepositional attachments was first reported in (Bear and 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.

3 Logic Form Transformation Rules

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 grammar rule obtained from the output of the parser it corresponds at least one 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 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.

Implementation

One of the most challenging problems in implementing LFT for natural language text is the large number of transformation rules that need to be written, as each grammar rule leads to one or more LFT rules. To deal with this coverage problem we devise a two-step procedure that works iteratively:

Step 1: implement first the most common grammar rules.

Step 2: increase the performance by adding more valuable rules.

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 10-90 rule of thumb

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: the top ten most frequently used rules cover more than 90% of the cases (see Table 5). We call this the *10-90 rule of thumb*. This might be explained by the relative uniform structure 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

| 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 |
| <i>ART ADJ₁ ADJ₂ NOUN</i> → <i>noun</i> (x_1) & <i>adj₁</i> (x_1) & <i>adj₂</i> (x_1) | return:n(x_1) & hard:a(x_1) & straight:a(x_1) | (a hard straight return (as in tennis or squash)) | {drive} |
| <i>ART ADJ₁ AND ADJ₂ NOUN</i> → <i>noun</i> (x_1) & <i>adj₁</i> (x_1) & <i>adj₂</i> (x_1) | light:n(x_1) & weak:a(x_1) & tremulous:a(x_1) | (a weak and tremulous light) | {shimmer, play} |
| <i>VERB ADV</i> → <i>verb</i> (e_1, x_1, x_2) & <i>adv</i> (e_1) | cut:v(e_1, x_1, x_2) & open:r(e_1) | (cut open) | {slash, gash} |
| <i>ART NOUN₁ 'S NOUN₂</i> → <i>noun₂</i> (x_1) & <i>noun₁</i> (x_2) & <i>pos</i> (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 |
| <i>VP₁ CONJ VP₂ PREP NP</i> → <i>conj</i> (e_1, e_2, e_3) & LFT(<i>VP₁</i> (e_2, x_1, x_2)) & LFT(<i>VP₂</i> (e_3, x_1, x_2)) & <i>prep</i> (e_1, x_3) & LFT(<i>NP</i>) | 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} |
| <i>NP₁ VP by NP₂ PREP NP₃</i> → LFT(<i>NP₁</i> (x_2)) & LFT(<i>VP</i> (e_1, x_1, x_2)) & LFT(<i>NP₂</i> (x_1)) & <i>prep</i> (x_1, x_3) & LFT(<i>NP₃</i> (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

| Phrase | Occurrences | Unique rules | Coverage |
|---------|-------------|--------------|----------|
| base NP | 33,643 | 857 | 69% |
| NP | 11,408 | 244 | 91% |
| VP | 19,415 | 450 | 70% |
| PP | 12,315 | 40 | 99% |
| S | 14,740 | 35 | 99% |

Table 5: Phrases and their coverage by the top 10 most frequent rules in 10,000 noun glosses

uncovered or at least postpone their implementation. The logic form transformation rules are implemented by hand, thus no errors are introduced. Thus the problem of precision in deriving the logic forms translates into a coverage problem of the selected grammar rules.

The *second step* of the procedure consists of adding to the set, new rules that prove to have the biggest impact on performance. These are determined through an iterative refinement process. The processing performed in the second step is outlined in the algorithm below.

```

procedure LFT( $S_0, G, \tau$ ) {
  input:  $S_0$  - initial set of rules
          $G$  - uncovered set of rules
          $\tau$  - improvement gain threshold
   $i = 0$ ;  $S = S_0$ ;  $\delta = 0$ ;  $S' = \emptyset$ ;
  old_perf = 0; apply( $S_0$ , new_perf);
   $\delta = \text{new\_perf} - \text{old\_perf}$ ;
  while (( $i \leq \text{upper\_limit}$ ) && ( $\delta \leq \tau$ )) {
     $S' = \text{most\_frequent\_rules}(G - S')$ ;
     $G = G - S'$ ;  $S = S \cup S'$ ;
    old_perf = new_perf;
    apply( $S$ , new_perf);
     $\delta = \text{new\_perf} - \text{old\_perf}$ ;
     $i = i + 1$ ; }
  return (new_perf);}

```

At each refinement phase, 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 step 1). 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 stops. The rules

in S' are determined based on their 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 iteration. At the end of step 2 the final set S contains the grammar rules most representative for the target corpus.

4 LFT Results

To validate our LFT implementation we have experimented it on a subset of 400 WordNet 1.6 noun glosses. The initial set of rules is formed from 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 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% for the grammar phrase the rule belongs to. At the end of the first step we have a set of the most frequent rules, S_0 . We run the set of rules S_0 (seventy rules) on our test data and compare the output with the LFTs obtained by hand. A coverage/precision of 72.5% was achieved. Then, we applied the procedure in step two to boost up the coverage. 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. We stop at iteration 8 when the coverage is 81% and the gain obtained with S_0 at the last iteration on the 400 glosses was less than the 1% threshold.

The results show that one can control the performance by adjusting the parameter τ and the cardinality of S' at the cost of a larger effort.

Since the transformation rules are done by hand, there LFT accuracy is the same as the coverage, namely 81%.

5 Applicability of LFT to Question Answering

The LFT of WordNet glosses are used in our Question Answering system (?) to extract some answers and provide answer explanations.

To be useful the glosses logic forms need to be transformed in axioms. One possibility is to generate several axioms for each gloss, from simple to more complex, as illustrated in Table 6.

5.1 From logic forms to axioms

There are some specific aspects in the derivation of axioms for each part of speech. Usually the **nouns** definition consists of a genus and differentia. The template for deriving noun axioms is: $concept(x) \rightarrow genus(x) \ \& \ differentia(x)$. Notice the propagation of arguments from the left hand side to the genus and differentia, without significant syntactic changes. The **verbs** also exhibit the same structural properties and the derivation is simple for the case of definitions containing only one verb. In the case of definitions consisting of a series of verbs, the derivation of axioms should take care of the syntactic function changes of the arguments on the right hand side from their counterparts on the left hand side. Consider $kill:v\#1 \rightarrow \{ "cause \ to \ die" \}$. The axiom is $kill:v\#1(e1, x1, x2, x3) \rightarrow cause(e2, x1, e3, x3) \ \& \ die(e3, x2)$. One notices the change of $x2$ from a direct object role for *kill* to a subject role for *die*. Also event $e1$ expands in two other events $e1, e2$. In the case of **adjectives** which modify nouns, the axioms borrow a virtual head noun as shown here: $American:a\#1(x1) \rightarrow of(x1, x2) \ \& \ United_States_of_America(x2)$. Similarly, since **adverbs** modify verbs - their arguments borrow the event of the verb. Adverb $fast:r\#1$ has an axiom: $fast:r\#1(e) \rightarrow quickly(e)$.

5.2 Answer extraction procedure

1. Transform questions and answers in logic forms. For each question, the information retrieval part of the QA system provides a set of candidate paragraphs that may contain the question answer. Thus, the input to the logic prover consists of the question and a candidate paragraph that needs to be evaluated. The first step here is to transform the question and paragraph into logic forms called QLF and ALF (from answer logic form). **2. Form lexical chains between pairs of concepts.** If all the keywords from a question are found in the answer paragraph we only need to check for the syntactic relation preservation. This is done via unification - explained later. Otherwise, we attempt to establish possible lexical chains between pair of concepts, one in QLF and one in ALF, to check whether or not are semantically linked. A chain between a pair of concepts is a sequence of other concepts that are linked via *hyponymy* relation and/or *axioms*. A chain is established when two paths each starting from different concepts intersect, that is have a WordNet concept in common.

Many chains may be found that link a pair of concepts. To evaluate all of them would be too costly and unnecessary. Thus a filtering mecha-

| |
|--|
| Colombian(x1) \leftrightarrow of(x1, x2) & Columbia(x2) |
| Colombian(x1) \leftrightarrow relate(e1, x1, x2) & Columbia(x2) |
| Colombian(x1) \leftrightarrow characteristic(x1) & of(x1, x2) & Columbia(x2) |
| Colombian(x1) \leftrightarrow of(x1, x2) & people(x2) & of(x2, x3) & Columbia(x3) |
| Colombian(x1) \leftrightarrow relate(e1, x1, x2) & people(x2) & of(x2, x3) & Columbia(x3) |
| Colombian(x1) \leftrightarrow characteristic(x1) & of(x1, x2) & people(x2) & of(x2, x3) & Columbia(x3) |

Table 6: Axioms extracted from the gloss of adjective *Colombian:a#1*: {of or relating to or characteristic of Colombia or its people}

| |
|--|
| kill:v#1(e, x, y, z) \leftrightarrow cause(e1, x, y, z) & die(e2, y) |
| kill:v#1(e, x, y, z) \leftrightarrow put(e, x, y, w) & to(e, w) & death(w) |

Table 7: Axioms extracted from the gloss of verb *kill:v#1*: {cause to die; put to death}

nism is required. We do this using two heuristics: (1) keep only the shortest lexical chains (2) if several short chains exist, pick those that contain more hypernymy relations. The rationale behind these is that the shorter the chain, the stronger the semantic relation between the pair of concepts, and hypernymy relation is prefer over axioms. **3. Apply unification on lexical chains.** Once chains are established between a pair of concepts from the QLF and ALF, the logic form representation provides us with a mechanism for performing agreement unification. This checks the syntactic constraints. Two concepts along the chain are unified if their predicates and arguments match. In a successful unification the arguments of question predicate will be bound to the arguments of answer predicate and the QLF and ALF updated to reflect the new status of arguments. Step 0 in Table 8 shows the QLF, respectively ALF. In step 1, the matching is performed between some predicates, e.g. *Lucelly_Garcia* from QLF, respectively ALF, and *x1* is bound to *x1'* which is reflected in the new QLF shown in step 1.

4. Extract inferences to provide explanation.

The concepts along those lexical chains that survive the unification test lead to inferences that explain the answer. It is only necessary to retrieve the concepts along the chain and the hypernymy and axioms explain the relation between them.

5.3 Examples

Example 1

Consider the TREC question:

Q045: When did Lucelly Garcia, former ambassador of Colombia to Honduras, die?

The answer is found in “*Several gunmen on a highway leading to the Colombian city of Ibague murdered Colombian Ambassador to Honduras*

Lucelly Garcia today”. As illustrated in Table 8 at Step 0 we are able to match a few predicates: *Lucelly_Garcia*, *ambassador*, *TIME-STAMP*. With the help of the axioms, chains are found: from *Colombian* in the answer to *Colombia* in the question, respectively from *murder* to *die* (see Figure 1). For *former* there was no link to a concept in the answer and we just ignore it (as being a modifier of an already matched predicate *ambassador*). The ALF in Step 1 shows *Colombian* expanded with axioms from WordNet (see Table 6). The new QLF to be proven contains only the predicate *die*. Step 2 in Table 8 shows the ALF after the expansion of *murder* with its corresponding axiom: *murder(e1, x1, x2) \leftrightarrow kill(e1,x1,x2) & intentionally(e1) & with(e1,x3) & premeditation(x3)*. Then Step 3 is derived using: *kill(e,x1,x2) \leftrightarrow cause(e1,x1,e2) & die(e2,x2)*. As explained earlier, the subject of *kill* is propagated as subject of *cause* and the object of *kill*, which is *Lucelly_Garcia*, as the subject to *die*. Also, we replicate the TIME-STAMP predicate to modify both *e2* and *e3*. The QLF is successfully proven as it becomes empty.

Example 2

Consider the TREC-9’s question:

Q481: Who shot Billy the Kid?

The Q/A system identifies a few paragraphs that contain all the keywords from the question and the answer type. 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.

| | |
|--------|--|
| Step 0 | QLF: Lucelly_Garcia(x1) & former(x1) & ambassador(x1) & of(x1, x2) & Colombia(x2) & to(x1, x3) & Honduras(x3) & die(e1,x1) & TIME-STAMP(e1) ALF: gunman(x2') & murder(e1',x2',x1') & Colombian(x1') & ambassador(x1') & to(x1',x3') & Honduras(x3') & Lucelly_Garcia(x1') & TIME-STAMP(e1') |
| Step 1 | QLF: of(x1', x2) & Colombia(x2) & die(e1,x1') ALF: gunman(x2') & murder(e1', x2', x1') & of(x1', x7') & Colombia(x7') & ambassador(x1') & to(x1',x3') & Honduras(x3') & Lucelly_Garcia(x1') & TIME-STAMP(e1') |
| Step 2 | QLF: die(e1,x1') ALF: gunman(x2') & kill(e1', x2', x1') & intentionally(e1') & with(e1', x8') & premeditation(x8') & of(x1', x7') & Colombia(x7') & ambassador(x1') & to(x2', x3') & Honduras(x3') & Lucelly_Garcia(x1') & TIME-STAMP(e1') |
| Step 3 | QLF: die(e1,x1') ALF: gunman(x2') & cause(e2', x2', e3') & die(e3', x1') & intentionally(e1') & with(e1', x') & premeditation(x8') & of(x1', x7') & Colombia(x7') & ambassador(x1') & to(x1', x3') & Honduras(x3') & Lucelly_Garcia(x1') & TIME-STAMP(e2') & TIME-STAMP(e3') |

Table 8: QLF and ALF as the prove for question Q045 proceeds

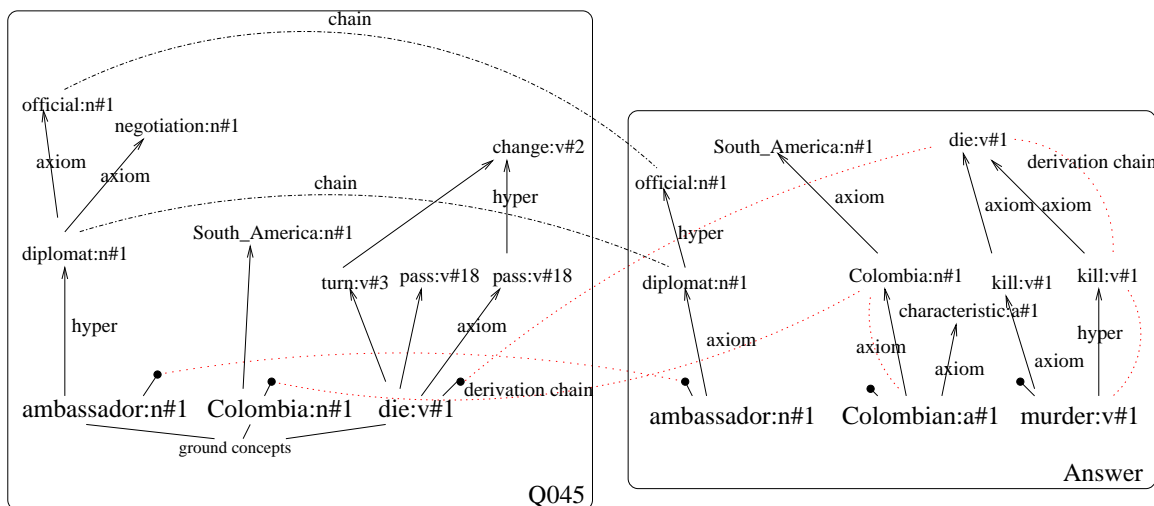


Figure 1: A partial illustration of chains and paths for question 045

, 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, the question has a representation of the form: $Q: PERSON(x1) \& shoot(e1, x1, x2) \& Billy_the_Kid(x2)$ where $x1$ is a variable that is to be unified with an entity of type PERSON from paragraphs. The paragraphs have the following LFT (we show only the relevant part):

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

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.

5.4 Q/A Results

Table 9 shows 5 questions for which all the keywords from the question are found in the answer. For these questions, the system found the correct answer in top 5 ranked answers (initial rank) just by matching keywords. The logic prover boosts the performance by eliminating the wrong answers

and bringing the correct answer to the first place.

Table 10 shows the results on 10 questions for

| Question | Initial rank | Final rank |
|----------|--------------|------------|
| Q074 | 2 | 1 |
| Q331 | 2 | 1 |
| Q381 | 5 | 1 |
| Q481 | 3 | 1 |
| Q640 | 2 | 1 |

Table 9: Examples of improvements to the TREC-9 results obtained by the system

| Q | Pairs | Paths | Chains | Concepts | Chains used |
|------|-------|-------|--------|----------|-------------|
| Q006 | 28 | 391 | 161 | 729 | 2 |
| Q034 | 30 | 257 | 20 | 90 | 2 |
| Q045 | 84 | 685 | 223 | 1025 | 3 |
| Q198 | 24 | 548 | 182 | 858 | 1 |
| Q302 | 54 | 617 | 251 | 1142 | 4 |
| Q424 | 27 | 472 | 180 | 840 | 2 |
| Q471 | 18 | 447 | 90 | 421 | 1 |
| Q498 | 21 | 157 | 12 | 52 | 2 |
| Q580 | 18 | 467 | 233 | 1087 | 2 |
| Q719 | 12 | 202 | 81 | 371 | 1 |

Table 10: Statistics for 10 questions

which we successfully retrieved chains between unmatched concepts in the question. To better evaluate the impact of the logic prover the word sense disambiguation task has been done manually for the questions, answers and for a set of targeted glosses. The *Pairs of Concepts* column illustrates the number of pairs of concepts from the question, respectively answer paragraph (ground concepts). The *paths* column shows the number of paths retrieved: these are paths that originate in the ground concepts, some of which intersect and form lexical chains. The *chains* column shows how many lexical chains were established. The next column, shows how many concepts were encountered along those paths. The *Chains used* column shows how many chains were selected to perform unification. In each case, one chain led to the correct answer.

6 Conclusions

We have presented here a procedure to transform WordNet glosses into logic forms. The notation used is first order logic and contains syntactic information as positional arguments. A two-step procedure to solve the coverage problem was introduced which obtained an overall coverage of 81% on 400 WordNet glosses. The coverage/precision is controllable. The paper demonstrates how

WordNet glosses provide world knowledge axioms essential for boosting the performance of a Question Answering system.

References

- John Bear and R. Jerry Hobbs. 1988. Localizing expression of ambiguity. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 235–242. Association for Computational Linguistic.
- M. Chodorow, R. Byrd, and G. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 299–304.
- David Davidson. 1967. The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press.
- Jerry R. Hobbs. 1986. Overview of the tacitus project. *Computational Linguistics*, 12(3).
- ISI. 1998. <http://www.isi.edu/natural-language/dpp/>.
- George Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- NIST. 2000. National institute for science and technology. <http://trec.nist.gov>.
- Stephen D. Richardson, William B. Dolan, and Lucy Vanderwende. 1998. Mindnet: acquiring and structuring semantic information from text. volume Proceedings of COLING '98.
- Y.A. Wilks, B.M. Slator, and L.M. Guthrie. 1996. *Electric Words- Dictionaries, Computers and Meanings*. The MIT Press.