# *Discovery of Inference Rules for Question Answering*

DEKANG LIN AND PATRICK PANTEL

*Department of Computing Science*
*University of Alberta*
*Edmonton, Alberta T6G 2H8 Canada*
*{lindek, ppantel}@cs.ualberta.ca*

**Abstract**

One of the main challenges in question-answering is the potential mismatch between the expressions in questions and the expressions in texts. While humans appear to use inference rules such as "X writes Y" implies "X is the author of Y" in answering questions, such rules are generally unavailable to question-answering systems due to the inherent difficulty in constructing them. In this paper, we present an unsupervised algorithm for discovering inference rules from text. Our algorithm is based on an extended version of Harris' Distributional Hypothesis, which states that words that occurred in the same contexts tend to be similar. Instead of using this hypothesis on words, we apply it to paths in the dependency trees of a parsed corpus. Essentially, if two paths tend to link the same set of words, we hypothesize that their meanings are similar. We use examples to show that our system discovers many inference rules easily missed by humans.

## 1 Introduction

One of the main challenges in question-answering and information retrieval is the potential mismatch between the expressions in questions and the expressions in texts. Suppose a question is phrased as "Who is the author of the 'Star Spangled Banner'?" Unless the system recognizes the relationship between "X wrote Y" and "X is the author of Y", it would not necessarily rank the sentence

> ... Francis Scott Key wrote the "Star Spangled Banner" in 1814.

higher than the sentence

> …comedian-actress Roseanne Barr sang her famous shrieking rendition of the "Star Spangled Banner" before a San Diego Padres-Cincinnati Reds game.

We call "*X wrote Y ≈ X is the author of Y*" an inference rule. In previous work, such relationships have been referred to as paraphrases or variants (Sparck Jones and Tait, 1984; Fabre and Jacquemin, 2000). In this paper, we use the term **inference rule** because we also want to include relationships that are not exactly paraphrases, but are nonetheless related and are potentially useful to question-

answering systems. For example, "*X caused Y ≈ Y is blamed on X*" is an inference rule even though the two phrases do not mean exactly the same thing.

Inference rules are extremely important in many fields such as natural language processing, information retrieval, and Artificial Intelligence in general. In the LASSO/FALCON systems (Harabagiu et al., 2000), the most successful QA systems in TREC-8 and TREC-9, a theorem prover is used to justify the answers. FALCON scored 19% higher with the answer justification process than without it.

Traditionally, knowledge bases containing such inference rules are created manually. This knowledge engineering task is extremely laborious. More importantly, building such a knowledge base is inherently difficult since humans are not good at generating a complete list of rules. For example, while it is quite trivial to come up with the rule "*X wrote Y ≈ X is the author of Y*", it seems hard to dream up a rule like "*X manufactures Y ≈ X's Y factory*", which can be used to infer that "*Chrétien visited Peugot's newly renovated car factory in the afternoon*" contains an answer to the query "*What does Peugot manufacture?*"

Most previous efforts on knowledge engineering have focused on creating tools for helping knowledge engineers transfer their knowledge to machines (Hahn and Schnattinger, 1998). Our goal is to automatically discover such rules.

In this paper, we present an unsupervised algorithm, **DIRT**, for **Discovering Inference Rules from Text**. Our algorithm is a generalization of previous algorithms for finding similar words (Hindle, 1990; Pereira, 1993; Lin, 1998). Algorithms for finding similar words assume the Distributional Hypothesis, which states that words that occurred in the same contexts tend to have similar meanings (Harris, 1985). Instead of applying the Distributional Hypothesis to words, we apply it to paths in dependency trees. Essentially, if two paths tend to link the same sets of words, we hypothesize that their meanings are similar. Since a path represents a binary relationship, we generate an inference rule for each pair of similar paths.

The remainder of this paper is organized as follows. In the next section, we review previous work. In Section 3, we define paths in dependency trees and describe their extraction from a parsed corpus. Section 4 presents the DIRT system. A comparison of our system's output with manually generated paraphrase expressions is shown in Section 5. Finally, we conclude with a discussion of future work.

## 2 Previous Work

Most previous work on variant recognition and paraphrase has been done in the fields of natural language generation, text summarization, and information retrieval.

The generation community focused mainly on rule-based text transformations in order to meet external constraints such as length and readability (Meteer and Shaked, 1988; Iordanskaja et al., 1991; Robin, 1994; Dras, 1997). Dras (1999)

described syntactic paraphrases using a meta-grammar with a synchronous Tree Adjoining Grammar (TAG) formalism.

In multi-document summarization, paraphrasing is important to avoid redundant statements in a summary. Given a collection of similar sentences (a theme) with different wordings, it is difficult to identify similar phrases that report the same fact. Barzilay et al. (1999) analyzed 200 two-sentence themes from a corpus and extracted seven lexico-syntactic paraphrasing rules. These rules covered 82% of syntactic and lexical paraphrases, which cover 70% of all variants. The rules are subsequently used to identify common statements in a theme by comparing the predicate-argument structure of the sentences within the theme.

In information retrieval, it is common to identify phrasal terms from queries and generate their variants for query expansion. It has been shown that such query expansion does promote effective retrieval (Arampatzis et al., 1998; Anick and Tipirneni, 1999). Morphological variant query expansion was treated by Sparck Jones and Tait (1984) using a semantic interpreter and by the Fastr system (Jacquemin et al., 1997). Also, Jacquemin (1999) proposed a rule-based system for the recognition of morpho-syntactic variants using morphological and light syntactic features (e.g. part-of speech and number agreement). Motivated by the fact that morpho-syntactic features inadequately separated correct and incorrect variants, Fabre and Jacquemin (2000) later extended this model using lexical semantics for obtaining noun-to-verb variants. The minor modifications to the model increased the recognition precision by 30% and reduced recognition recall by 10%.

In (Richardson, 1997), Richardson extracted semantic relationships (e.g., hypernym, location, material and purpose) from dictionary definitions using a parser and constructed a semantic network. He then described an algorithm that uses paths in the semantic network to compute the similarity between words. In a sense, our algorithm is a dual of Richardson's approach. While Richardson used paths as features to compute the similarity between words, we use words as features to compute the similarity of paths.

Many text mining algorithms aim at finding association rules between terms (Lin et al., 1998). In contrast, the output of our algorithm is a set of associations between relations. Term associations usually require human interpretation; however, some of them are considered to be uninterpretable even by humans (Feldman et. al., 1998).

## 3 Extraction of Paths from Dependency Trees

The inference rules discovered by DIRT are between paths in dependency trees. In this section, we first briefly describe the parser used to generate the dependency trees. Then, we describe an algorithm for extracting paths from the trees.

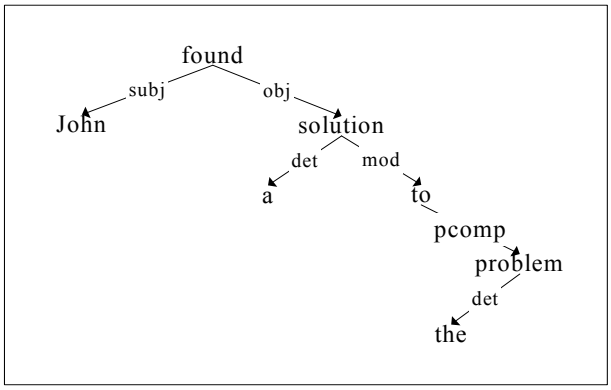



Figure 1. The dependency tree for the sentence *John found a solution to the problem* extracted by Minipar .

### *3.1 Minipar*

Minipar[1] is a principle-based English parser (Berwick, 1991). Like Principar (Lin, 1993), Minipar represents its grammar as a network where nodes represent grammatical categories and links represent types of syntactic (dependency) relationships. The grammar network consists of 35 nodes and 59 links. Additional nodes and links are created dynamically to represent subcategories of verbs. Minipar employs a message passing algorithm that essentially implements distributed chart parsing. Instead of maintaining a single chart, each node in the grammar network maintains a chart containing partially built structures belonging to the grammatical category represented by the node. The grammatical principles are implemented as constraints associated with the nodes and links.

The lexicon in Minipar is derived from syntactic features (parts of speech and subcategorization frames) in WordNet (Miller, 1990). With additional proper names, the lexicon contains about 130,000 entries (in base forms). The lexicon entry of a word lists all possible parts of speech of the word and its subcategorization frames (if any). The lexical ambiguities are handled by the parser instead of a tagger.

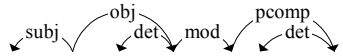
Minipar works with a constituency grammar internally, however the output of Minipar is a dependency tree. The conversion is straightforward because all the constituents in the constituency grammar have a head. Figure 1 shows an example dependency tree for the sentence "*John found a solution to the problem.*" The links in the diagram represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent types of dependency relations. Table 1 lists a subset of the dependency

[1] Available at http://www.cs.ualberta.ca/~lindek/minipar.htm

*Table 1*. A subset of the dependency relations in Minipar outputs.

| RELATION | DESCRIPTION | EXAMPLE |
|---|---|---|
| appo | appositive of a noun | the CEO, **John** |
| det | determiner of a noun | **the** dog |
| gen | genitive modifier of a noun | **John's** dog |
| mod | adjunct modifier of any type of head | **tiny** hole |
| nn | prenominal modifier of a noun | **station** manager |
| pcomp | complement of a preposition | in the **garden** |
| subj | subject of a verb | **John** loves Mary. |
| sc | small clause complement of a verb | She forced him to **resign** |

relations in Minipar outputs. For the sake of space, from now on we represent dependency trees in a more compact form as follows:



John found a solution to the problem.

Like chart parsers, Minipar constructs all possible parses of an input sentence. However, only the highest ranking parse tree is output. Although the grammar is manually constructed, the selection of the best parse tree is guided by the statistical information obtained by parsing a 1GB newspaper corpus with Minipar. The statistical ranking of parse trees is based on the following probabilistic model. The probability of a dependency tree is defined as the product of the probabilities of the dependency relationships in the tree. Formally, given a tree $T$ with root *root* consisting of $D$ dependency relationships ($head_i$, $relationship_i$, $modifier_i$), the probability of $T$ is given by:

$$P(T) = P(root)\prod_{i=1}^{D} P(relationship_i, modifier_i \mid head_i)$$

where $P(relationship_i, modifier_i \mid head_i)$ is obtained using Maximum Likelihood Estimation (MLE).

Minipar parses newspaper text at about 500 words per second on a Pentium-III 700Mhz with 500MB memory. Evaluation with the manually parsed SUSANNE corpus (Sampson, 1995) shows that about 89% of the dependency relationships in Minipar outputs are correct. The recall of Minipar output, defined as the percentage of dependency relationships in the SUSANNE corpus that are extracted by Minipar, varies a great deal depending on the genre of the input document from 80% (novels) to 87% (news reportage). This accuracy is comparable to other broad-coverage English parsers (Collins, 1996; Charniak, 2000).

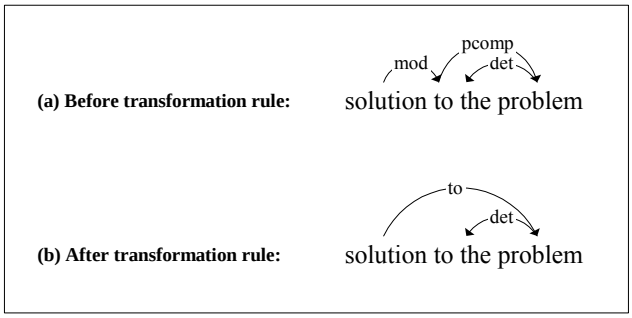



Figure 2. Effect of the transformation rule to connect the prepositional complement to the words modified by the preposition for the phrase *solution to the problem*.

### *3.2 Paths in Dependency Trees*

In the dependency trees generated by Minipar, prepositions are represented by nodes. We apply a simple transformation rule to connect the prepositional complement directly to the words modified by the preposition. We name this direct relationship with the preposition. Figure 2 gives an example for the phrase "*solution to the problem*" in which the two links in part (*a*) are replaced with a direct link shown in part (*b*).

After the transformation, each link between two words in a dependency tree represents a direct semantic relationship. A path allows us to represent indirect semantic relationships between two content words. We name a path by concatenating dependency relationships and words along the path, excluding the words at the two ends. For the sentence in Figure 1, the path between *John* and *problem* is named: N:subj:V←find→V:obj:N→solution→N:to:N (meaning "*X finds solution to Y*"). The reverse path of the above path can be written as: N:to:N←solution←N:obj:V←find→ V:subj:N. The **root** of both paths is *find*. A path begins and ends with two dependency relations. We call them the two **slots** of the path: *SlotX* on the left-hand side and *SlotY* on the right-hand side. The words connected by the path are the **fillers** of the slots. For example, *John* fills the *SlotX* of N:subj:V←find→V:obj:N→solution→N:to:N and *problem* fills the *SlotY*. The reverse is true for N:to:N←solution←N:obj:V←find→V:subj:N. In a path, dependency relations that are not slots are called **internal relations**. For example, find→V:obj:N→solution is an internal relation in the previous path.

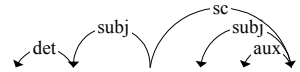
We impose a set of constraints on the paths to be extracted from text for the following reasons:

- most meaningful inference rules involve only paths that satisfy these conditions;

- the constraints significantly reduce the number of distinct paths and, consequently, the amount of computation required for computing similar paths; and
- the constraints alleviate the sparse data problem because long paths tend to have very few occurrences.

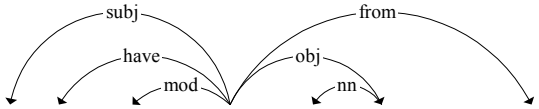
The constraints are:

- slot fillers must be nouns because slots correspond to variables in inference rules and we expect the variables to be instantiated by entities;
- any dependency relation that does not connect two content words (i.e. nouns, verbs, adjectives or adverbs) is excluded from a path. E.g. in Figure 1, the relation between *a* and *solution* is excluded;
- the frequency count of an internal relation must exceed a threshold; and
- an internal relation must be between a verb and an object-noun or a small clause. The relationship between *find* and *solution* in "*John found a solution to the problem*" is an example of a *verb-object* relation. The relationship between *force* and *resign* is an example of a *verb-small clause* relationship in the following sentence:



The board forced him to resign.

Consider the following sentence:



They had previously bought bighorn sheep from Comstock.

The paths extracted from this sentence and their meanings are:

| | | |
|---|---|---|
| (a) | N:subj:V←buy→V:from:N | (*X* buys something from *Y*) |
| (b) | N:subj:V←buy→V:obj:N | (*X* buys *Y*) |
| (c) | N:subj:V←buy→V:obj:N→sheep→N:nn:N | (*X* buys *Y* sheep) |
| (d) | N:nn:N←sheep←N:obj:V←buy→V:from:N | (*X* sheep is bought from *Y*) |
| (e) | N:obj:V←buy→V:from:N | (*X* is bought from *Y*) |

An inverse path is also added for each one above.

## 4 DIRT: Discovering Inference Rules from Text

A path is a binary relation between two entities. In this section, we present an algorithm to automatically discover the inference relations between such binary relations.

### *4.1 The Underlying Assumption*

Most algorithms for computing word similarity from text corpus are based on a principle known as the Distributional Hypothesis (Harris, 1985). The idea is that words that tend to occur in the same contexts tend to have similar meanings. Previous efforts differ in their representation of the context and in their formula for computing the similarity between two sets of contexts. Some algorithms use the words that occurred in a fixed window of a given word as its context while others use the dependency relationships of a given word as its context (Lin, 1998). Consider the words *duty* and *responsibility*. There are many contexts in which both of these words can fit. For example,

- *duty* can be modified by adjectives such as *additional*, *administrative*, *assigned*, *assumed*, *collective*, *congressional*, *constitutional*, …, so can *responsibility*;
- *duty* can be the object of verbs such as *accept*, *articulate*, *assert*, *assign*, *assume*, *attend to*, *avoid*, *become*, *breach*, …, so can *responsibility*.

Based on these common contexts, one can statistically determine that *duty* and *responsibility* have similar meanings.

In the algorithms for finding word similarity, dependency links are treated as contexts of words. In contrast, our algorithm for finding inference rules treats the words that fill the slots of a path as a context for the path. We make an assumption that this is an extension to the Distributional Hypothesis:

> **Extended Distributional Hypothesis:**
> If two paths tend to occur in similar contexts, the meanings of the paths tend to be similar.

For example, Table 2 lists a set of example pairs of words connected by the paths N:subj:V←find→V:obj:N→solution→N:to:N ("*X finds a solution to Y*") and N:subj:V← solve→V:obj:N ("*X solves Y*"). As it can be seen from the table, there are many overlaps between the corresponding slot fillers of the two paths. By the Extended Distributional Hypothesis, we can then claim that the two paths have similar meanings.

### *4.2 Triples*

To compute the path similarity using the Extended Distributional Hypothesis, we need to collect the frequency counts of all paths in a corpus and the slot fillers for the paths. For each instance of a path $p$ that connects two words $w_1$ and $w_2$, we increase the frequency counts of the two triples ($p$, *SlotX*, $w_1$) and ($p$, *SlotY*, $w_2$). We call (*SlotX*, $w_1$) and (*SlotY*, $w_2$) features of path $p$. Intuitively, the more features two paths share, the more similar they are.

Table 2. Sample slot fillers for two paths extracted from a newspaper corpus.

| "*X finds a solution to Y*" | | "*X solves Y*" | |
|---|---|---|---|
| *SLOTX* | *SLOTY* | *SLOTX* | *SLOTY* |
| commission | strike | committee | problem |
| committee | civil war | clout | crisis |
| committee | crisis | government | problem |
| government | crisis | he | mystery |
| government | problem | she | problem |
| he | problem | petition | woe |
| I | situation | researcher | mystery |
| legislator | budget deficit | resistance | crime |
| sheriff | dispute | sheriff | murder |

We use a **triple database** (a hash table) to accumulate the frequency counts of all features of all paths extracted from a parsed corpus. An example entry in the triple database for the path

N:subj:V←pull→V:obj:N→body→N:from:N ("*X pulls body from Y*")

is shown in Figure 3. The first column of numbers in Figure 3 represents the frequency counts of a word filling a slot of the path and the second column of numbers is the mutual information between a slot and a slot filler. Mutual information measures the strength of the association between a slot and a filler. We explain mutual information in detail in the next section. The triple database records the fillers of *SlotX* and *SlotY* separately. Looking at the database, one would be unable to tell which *SlotX* filler occurred with which *SlotY* filler in the corpus.

### *4.3 Mutual Information*

Mutual information is a commonly used measure for the association strength between two words (Church and Hanks, 1989). The mutual information between two events *x* and *y* is given by:

$$mi(x,y) = \log \frac{P(x,y)}{P(x)P(y)} \quad \textbf{(1)}$$

Mutual information is high when *x* and *y* occur together more often than by chance. Mutual information compares two models for predicting the co-occurrence of *x* and *y*: one is the MLE of the joint probability of *x* and *y* and the other is some baseline model. In Equation (1), the baseline model assumes that *x*

| **X pulls body from Y:** | | | |
|---|---|---|---|
| *SlotX*: | | | |
| | diver | 1 | 2.45 |
| | equipment | 1 | 1.65 |
| | police | 2 | 2.24 |
| | rescuer | 3 | 4.84 |
| | resident | 1 | 1.60 |
| | who | 2 | 1.32 |
| | worker | 1 | 1.37 |
| *SlotY*: | | | |
| | bus | 2 | 3.09 |
| | coach | 1 | 2.05 |
| | debris | 1 | 2.36 |
| | feet | 1 | 1.75 |
| | hut | 1 | 2.73 |
| | landslide | 1 | 2.39 |
| | metal | 1 | 2.09 |
| | wreckage | 3 | 4.81 |

Figure 3. An example entry in the triple database for the path "*X pulls body from Y*".

and *y* are independent. Note that in information theory, mutual information refers to the mutual information between two random variables rather than between two events as used in this paper. The mutual information between two random variables *X* and *Y* is given by:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \quad \textbf{(2)}$$

The mutual information between two random variables is the weighted average of all possible combinations of events involving the two variables.

A triple involves three events: the path, the slot, and the filler. Equation (1) defines the mutual information between two events. Alshawi and Carter (1994) generalized Equation (1) to handle three events:

$$mi(x,y,z) = \log \frac{P(x,y,z)}{P(x)P(y)P(z)} \quad \textbf{(3)}$$

Equation (3) compares the MLE of the joint probability of *x*, *y*, and *z* with the model that assumes independence between *x*, *y*, and *z*. Since paths, slots, and fillers are not independent, we use a slightly more accurate baseline model, which assumes that a path and a filler are conditionally independent given a slot. We get:

$$mi(p,Slot,w) = log \frac{P(p,Slot,w)}{P(Slot)P(p \mid Slot)P(w \mid Slot)} \quad \textbf{(4)}$$

where $p$ is a path, *Slot* is either *SlotX* or *SlotY*, and $w$ is a filler.

We use the notation $|p, SlotX, w|$ to denote the frequency count of the triple ($p$, *SlotX*, $w$), $|p, SlotX, *|$ to denote $\sum_{w \in W} |p, SlotX, w|$, and $|*, *, *|$ to denote $\sum_{p,s,w} |p, s, w|$. The mutual information of a triple ($p$, *Slot*, $w$) can be computed by the formula:

$$mi(p, Slot, w) = \log \frac{\frac{|p, Slot, w|}{|*,*,*|}}{\frac{|*, Slot, *|}{|*,*,*|} \frac{|p, Slot, *|}{|*, Slot, *|} \frac{|*, Slot, w|}{|*, Slot, *|}} = \log \frac{|p, Slot, w| \times |*, Slot, *|}{|p, Slot, *| \times |*, Slot, w|} \quad \textbf{(5)}$$

### *4.4 Similarity between Two Paths*

Once the triple database is created, the similarity between two paths can be computed in the same way that the similarity between two words is computed in (Lin, 1998). Essentially, two paths have high similarity if there are a large number of common features. However, not every feature is equally important. For example, the word *he* is much more frequent than the word *sheriff*. Two paths sharing the feature (*SlotX*, *he*) is less indicative of their similarity than if they shared the feature (*SlotX*, *sheriff*). The similarity measure proposed in (Lin, 1998) takes this into account by computing the mutual information between a feature and a path.

The similarity between a pair of slots: $slot_1 = (p_1, s)$ and $slot_2 = (p_2, s)$, is defined as:

$$sim(slot_1, slot_2) = \frac{\sum_{w \in T(p_1,s) \cap T(p_2,s)} mi(p_1, s, w) + mi(p_2, s, w)}{\sum_{w \in T(p_1,s)} mi(p_1, s, w) + \sum_{w \in T(p_2,s)} mi(p_2, s, w)} \quad \textbf{(6)}$$

where $p_1$ and $p_2$ are paths, $s$ is a slot, $T(p_i, s)$ is the set of words that fill in the $s$ slot of path $p_i$.

The similarity between a pair of paths $p_1$ and $p_2$ is defined as the geometric average of the similarities of their *SlotX* and *SlotY* slots:

$$S(p_1, p_2) = \sqrt{sim(SlotX_1, SlotX_2) \times sim(SlotY_1, SlotY_2)} \quad \textbf{(7)}$$

where $SlotX_i$ and $SlotY_i$ are path $i$'s *SlotX* and *SlotY* slots.

### *4.5 Finding the Most Similar Paths*

Given a path, the discovery of inference rules is made by finding its most similar paths. The challenge here is that there are a large number of paths in the triple database. The database used in our experiments contains over 200,000 distinct

Table 3. The top-50 most similar paths to "*X solves Y*".

| | | | |
|---|---|---|---|
| 1. | Y is solved by X | 26. | X clears up Y |
| 2. | X resolves Y | 27. | *X creates Y |
| 3. | X finds a solution to Y | 28. | *Y leads to X |
| 4. | X tries to solve Y | 29. | Y is eased between X |
| 5. | X deals with Y | 30. | X gets down to Y |
| 6. | Y is resolved by X | 31. | X worsens Y |
| 7. | X addresses Y | 32. | X ends Y |
| 8. | X seeks a solution to Y | 33. | *X blames something for Y |
| 9. | X do something about Y | 34. | X bridges Y |
| 10. | X solution to Y | 35. | X averts Y |
| 11. | Y is resolved in X | 36. | *X talks about Y |
| 12. | Y is solved through X | 37. | X grapples with Y |
| 13. | X rectifies Y | 38. | *X leads to Y |
| 14. | X copes with Y | 39. | X avoids Y |
| 15. | X overcomes Y | 40. | X solves Y problem |
| 16. | X eases Y | 41. | X combats Y |
| 17. | X tackles Y | 42. | X handles Y |
| 18. | X alleviates Y | 43. | X faces Y |
| 19. | X corrects Y | 44. | X eliminates Y |
| 20. | X is a solution to Y | 45. | Y is settled by X |
| 21. | X makes Y worse | 46. | *X thinks about Y |
| 22. | X irons out Y | 47. | X comes up with a solution to Y |
| 23. | *Y is blamed for X | 48. | X offers a solution to Y |
| 24. | X wrestles with Y | 49. | X helps somebody solve Y |
| 25. | X comes to grip with Y | 50. | *Y is put behind X |

paths. Computing the similarity between every pair of paths is obviously impractical.

Given a path *p,* our algorithm for finding the most similar paths of *p* takes three steps:

(a) Retrieve all the paths that share at least one feature with *p* and call them candidate paths. This can be done efficiently by storing for each word the set of slots it fills in.

(b) For each candidate path *c*, count the number of features shared by *c* and *p*. Filter out *c* if the number of its common features with *p* is less than a fixed percent (we used 1%) of the total number of features for *p* and *c*. This step effectively uses a simpler similarity formula to filter out some of the paths since computing mutual information is more costly than counting the number of features. This idea has previously been used in Canopy (McCallum et al., 2000).

Table 4. First 15 questions from TREC-8.

| QUESTION # | QUESTION |
|---|---|
| $Q_1$ | Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"? |
| $Q_2$ | What was the monetary value of the Nobel Peace Prize in 1989? |
| $Q_3$ | What does the Peugeot company manufacture? |
| $Q_4$ | How much did Mercury spend on advertising in 1993? |
| $Q_5$ | What is the name of the managing director of Apricot Computer? |
| $Q_6$ | Why did David Koresh ask the FBI for a word processor? |
| $Q_7$ | What debts did Qintex group leave? |
| $Q_8$ | What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)? |
| $Q_9$ | How far is Yaroslavl from Moscow? |
| $Q_{10}$ | Name the designer of the shoe that spawned millions of plastic imitations, known as "jellies". |
| $Q_{11}$ | Who was President Cleveland's wife? |
| $Q_{12}$ | How much did Manchester United spend on players in 1993? |
| $Q_{13}$ | How much could you rent a Volkswagen bug for in 1966? |
| $Q_{14}$ | What country is the biggest producer of tungsten? |
| $Q_{15}$ | When was London's Docklands Light Railway constructed? |

(c) Compute the similarity between $p$ and the candidates that passed the filter using equation (6) and output the paths in descending order of their similarity to $p$.

Table 3 lists the Top-50 most similar paths to "*X solves Y*" generated by DIRT. The ones tagged with an asterisk (*) are incorrect, as judged by the authors. Most of the paths can be considered as paraphrases of the original expression.

The Extended Distributional Hypothesis, as with the original Distributional Hypothesis, is a statement about general trend instead of individual instances. There are plenty of exceptions in the text, as evidenced by the asterisk-tagged paths in Table 3.

## 5 Experimental Results

Ideally, we would evaluate our system by injecting the inference rules into a full-fledged question-answering system. However, at this point, we have not built such a system. Therefore, we performed an evaluation of our algorithm by comparing the inference rules it generates with a set of human-generated paraphrases of the first 15 questions in the TREC-8 Question-Answering Track, listed in Table 4. TREC (Text REtrievial Conference) is a U.S. government sponsored competition on information retrieval held annually since 1992. In the Question-Answering Track, the task for participating systems is to find answers to natural-language questions like those in Table 4.

### *5.1 Experimental Setup*

We used Minipar to parse about 1GB of newspaper text (San Jose Mercury, Wall Street Journal and AP Newswire from the TREC collection). Using the methods discussed in Section 3, we extracted 7 million paths from the parse trees (231,000 unique) and stored them in a triple database.

### *5.2 Results*

The second column of Table 5 shows the paths that Minipar identified from the TREC-8 questions. For some questions, more than one path was identified. For others, no path was found (represented by ∅ in Table 5).

We compare the output of our algorithm with a set of manually generated paraphrases of the TREC-8 questions made available at ISI[2]. Table 6 gives the paraphrases for $Q_1$ and $Q_3$.

We also extracted paths from the manually generated paraphrases. For some paraphrases, an identical path is extracted. For example, "*What things are manufactured by Peugeot?*" and "*What products are manufactured by Peugeot?*" both map to the path "*X is manufactured by Y*". Additionally, some paraphrases do not map to any paths. For example, one of the paraphrases of $Q_2$ is the multi-sentence query "*When the Norwegian Nobel Committee issues a Peace Prize, it also gives a financial award. How much was that award in 1989?*" This paraphrase does not seem to contain any variation of the path in the original question. The number of paths for the manually generated paraphrases of TREC-8 questions is shown in the third column of Table 5.

For each of the paths $p$ in the second column of Table 5, we ran the DIRT algorithm to compute its Top-40 most similar paths using the triple database. We then manually inspected the outputs and classified each extracted path as *correct* or *incorrect*. A path $p'$ is judged correct if a sentence containing $p'$ might contain an answer to the question from which $p$ was extracted. Consider question $Q_3$ in

[2] Available at http://www.isi.edu/~gerber/Variations2.txt

*Table 5*. Evaluation of Top-40 most similar paths.

| QUESTION | PATHS | MANUAL | DIRT (CORRECT) | INTERSECTION | ACCURACY |
|---|---|---|---|---|---|
| $Q_1$ | X is author of Y | 7 | 21 | 2 | 52.5% |
| $Q_2$ | X is monetary value of Y | 6 | 0 | 0 | 0% |
| $Q_3$ | X manufactures Y | 13 | 37 | 4 | 92.5% |
| $Q_4$ | X spend Y | 7 | 16 | 2 | 40.0% |
| | spend X on Y | 8 | 15 | 3 | 37.5% |
| $Q_5$ | X is managing director of Y | 5 | 14 | 1 | 35.0% |
| $Q_6$ | X asks Y | 2 | 23 | 0 | 57.5% |
| | asks X for Y | 2 | 14 | 0 | 35.0% |
| | X asks for Y | 3 | 21 | 3 | 52.5% |
| $Q_7$ | X leave Y | 4 | 0 | 0 | 0% |
| $Q_8$ | X is disease with Y | 5 | 0 | 0 | 0% |
| $Q_9$ | ∅ | N/A | N/A | N/A | N/A |
| $Q_{10}$ | X is designer of Y | 5 | 7 | 2 | 17.5% |
| $Q_{11}$ | ∅ | N/A | N/A | N/A | N/A |
| $Q_{12}$ | ∅ | N/A | N/A | N/A | N/A |
| $Q_{13}$ | rent X for Y | 14 | 16 | 1 | 40.0% |
| $Q_{14}$ | X is producer of Y | 10 | 31 | 3 | 77.5% |
| $Q_{15}$ | ∅ | N/A | N/A | N/A | N/A |

Table 4 where we have $p$ = "*X manufactures Y*" and we find $p'$ = "*X's Y factory*" as one of $p$'s Top-40 most similar paths. Since "*Peugeot's car factory*" might be found in some corpus, $p'$ is judged correct. Note that not all sentences containing $p'$ necessarily contain an answer to $Q_3$ (e.g. "*Peugeot's Sochaux factory*" gives the location of a Peugeot factory in France).

The fourth column in Table 5 shows the number of Top-40 most similar paths classified as *correct* and the fifth column gives the intersection between columns three and four. Finally, the last column in Table 5 gives the percentage of top-40 paths classified as correct.

Table 6. Manually-generated paraphrases of questions 1 and 3 in TREC-8.

| Q # | PARAPHRASES |
|---|---|
| $Q_1$ | Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"? |
| | "The Iron Lady: A Biography of Margaret Thatcher" was the work of what writer? |
| | Name the author of Margaret Thatcher's biography which is called "The Iron Lady". |
| | Name the writer of a Margaret Thatcher biography titled, "The Iron Lady". |
| | What author penned, "The Iron Lady: A Biography of Margaret Thatcher"? |
| | What writer produced "The Iron Lady: A Biography of Margaret Thatcher"? |
| | Who authored "The Iron Lady: A Biography of Margaret Thatcher"? |
| | Who chronicled Margaret Thatcher's political career in "The Iron Lady"? |
| | Who wrote, "The Iron Lady: A Biography of Margaret Thatcher"? |
| $Q_3$ | What does the Peugeot company manufacture? |
| | What does the Peugeot company make? |
| | What does the Peugeot company produce? |
| | What line of business is Peugeot in? |
| | The Peugeot company is in what business? |
| | What things are manufactured by Peugeot? |
| | What products are manufactured by Peugeot? |
| | What goods or services are provided by Peugeot? |
| | What is manufactured by the Peugeot Company? |
| | What are the Peugeot company's products? |
| | Name a product from Peugeot. |
| | Name any Peugeot product. |
| | Name some products made by Peugeot. |
| | What are some examples of Peugeot products? |
| | What does Peugeot make? |
| | What is Peugeot a manufacturer of? |
| | What is Peugeot's main product? |
| | What kinds of things will we find in Peugeot's product line? |
| | What product does Peugeot make? |
| | What products will we find in a Peugeot catalog? |
| | What things does Peugeot manufacture? |
| | What will we find in a Peugeot catalog? |

Table 7. Paths found for five of the 15 questions in TREC-8 and the variations discovered manually and by DIRT.

| Q | PATHS | MANUAL VARIATIONS | DIRT VARIATIONS |
|---|---|---|---|
| $Q_1$ | *X* is author of *Y* | *Y* is the work of *X*; *X* is the writer of *Y*; *X* penned *Y*; *X* produced *Y*; *X* authored *Y*; *X* chronicled *Y*; *X* wrote *Y* | *X* co-authors *Y*; *X* is co-author of *Y*; *X* writes Y; *X* edits *Y*; *Y* is co-authored by *X*; *Y* is authored by *X*; *X* tells story in *Y*; *X* translates *Y*; *X* writes in *Y*; *X* notes in *Y*; … |
| $Q_3$ | *X* manufactures *Y* | *X* makes *Y*; *X* produce *Y*; *X* is in *Y* business; *Y* is manufactured by *X*; *Y* is provided by *X*; *Y* is *X*'s product; *Y* is product from *X*; *Y* is *X* product; *Y* is product made by *X*; *Y* is example of *X* product; *X* is manufacturer of *Y*; find *Y* in *X*'s product line; find *Y* in *X* atalog | *X* produces *Y*; *X* markets *Y*; *X* develops *Y*; *X* is supplier of *Y*; *X* ships *Y*; *X* supplies *Y*; *Y* is manufactured by *X*; *X* is maker of *Y*; *X* introduces *Y*; *X* exports *Y*; *X* makes *Y*; *X* builds *Y*; *X*'s production of *Y*; *X* unveils *Y*; *Y* is bought from *X*; *X*'s line of *Y*; *X* assembles *Y*; *X* is *Y* maker; *X*'s *Y* factory; *X*'s *Y* production; *X* is manufacturer of *Y*; *X*'s *Y* division; *X* meets demand for *Y*; … |
| $Q_4$ | *X* spend *Y* | *X* put *Y* into marketing; at *X*, *Y* was spent; *X* invest *Y*; *X* pay *Y*; *Y* is *X*'s budget; *Y* is *X*'s outlay; *Y* is *X*'s spending | *X* invests *Y*; *X* pays *Y*; *X* pays somebody *Y*; *X* contributes *Y*; *Y* is spent by *X*; *X* allocates *Y*; *X* wastes *Y*; *X* pours *Y*; *X* puts up *Y*; … |
| | spend *X* on *Y* | put *X* into *Y*; *X* was spent on *Y*; invest *X* in *Y*; pay *X* for *Y*; *X* is *Y* budget; *X* is *Y* outlay; *X* is spending for *Y*; *X* is *Y* spending | *X* pays for *Y*; *X* spends something for *Y*; *X*'s *Y* budget; *X* finances *Y*; *X* purchases *Y*; *X* goes ahead with *Y*; … |
| $Q_6$ | *X* asks *Y* | *X* request something from *Y*; *X*'s request to *Y* | *X* tells *Y*; *X* meets with *Y*; *X* informs *Y*; *X* contacts *Y*; *X* writes to *Y*; … |
| | asks *X* for *Y* | requests *Y* from *X*; request to *X* for *Y* | *X* grants somebody *Y*; *X* gives somebody *Y*; *Y* is granted by *X*; *X* approves *Y*; *X* grants *Y*; *Y* is sought from *X*; *Y* is received from *X*; … |
| | *X* asks for *Y* | *X* wants *Y*; *X* requests *Y*; *X*'s request for *Y* | *X* requests *Y*; *X* seeks *Y*; *X*'s request for *Y*; *X* obtains *Y*; *Y* is requested by *X*; *X* solicits *Y*; *X* requests for *Y*; *X* demands *Y*; *X* pleads for *Y*; *X* wants *Y*; *X* presses for *Y*; *X* appeals for *Y*; ... |
| $Q_{14}$ | *X* is producer of *Y* | *X* is *Y* producing nation; *X* is *Y* producer; *X* leads in *Y* production; *Y* comes from *X*; *X* produces *Y*; *X* is leader in *Y* production; produced *Y* in *X*; *X* bring *Y* [to world markets]; *X* mines *Y*; *X* tops list of *Y* production | *X* is *Y* producer; imports *Y* from *X*; *X* ships *Y*; *X*'s *Y* export; *X*'s *Y* output; *X*'s *Y* production; *Y* production in *X*; *X*'s *Y* industry; *X*'s *Y* mine; *X*'s production of *Y*; *Y* is produced in *X*; *X* is maker of *Y*; *X* produces *Y*; *X*'s *Y* business; *Y* shipment from *X*; *X* supplies *Y*; *X* exports *Y*; *X*'s *Y* reserve; *Y* is bought from *X*; … |

### *5.3 Observations*

There is very little overlap between the automatically generated paths and the paraphrases, even though the percentage of correct paths in DIRT outputs can be quite high. This suggests that finding potentially useful inference rules is very difficult for humans as well as machines. Table 7 shows some of the *correct* paths among the Top-40 extracted by our system for five of the TREC-8 questions. Many of the variations generated by DIRT that are correct paraphrases are missing from the manually generated variations, and vice versa. It is difficult for humans to generate a diverse list of paraphrases, given a starting formulation and no context. However, given the output of our system, humans can easily identify the correct inference rules. Hence, at the least, our system would greatly ease the manual construction of inference rules for a QA system.

The performance of DIRT varies a great deal for different paths. Usually, the performance for paths with verb roots is much better than for paths with noun roots. A verb phrase typically has more than one modifier[3], whereas nouns usually take a smaller number of modifiers. When a word takes less than two modifiers, it will not be the root of any path. As a result, paths with noun roots occur less often than paths with verb roots, which explains the lower performance with respect to paths with noun roots.

In Table 5, DIRT found no correct inference rules for three of the questions. The paths for $Q_2$ and $Q_8$ do not have any entries in the triple database. The performance for $Q_7$ is poor for a different reason. Although the triple database contains plenty of features for "*X leaves Y*", all of the similar paths found by DIRT refer to the travel sense of *leave*, such as "*X flees Y*" and "*X visits Y*". In "$Q_7$: *What debts did Qintex group leave?*" the intended meaning of *leave* is "to cause something to remain."

Another source of error in our algorithm is exemplified by the following: among the most similar paths of "*X asks Y*", we have "*X informs Y*" (which is correct), but also "*Y asks X*" and "*Y informs X*". The reason is that both askers and askees tend to be persons or organizations. Since the similarity of paths depends totally on the similarity of their slots, slots with the same kind of fillers are not distinguished in our algorithm. Predicting whether this type of error will happen in the outputs for a given path is easy. We can simply compute the similarity between its *SlotX* and its *SlotY*. However, teasing out the incorrect inference rules caused by this is still a problem.

## 6 Conclusion and Future Work

Better tools are necessary to tap into the vast amount of textual data that is growing at an astronomical pace. Knowledge about inference relationships between natural language expressions is extremely important for question-answering and

[3] In this paper, we use dependency grammar terminology where the term *modifier* refers to both modifiers and arguments in *X*-bar Theory.

many other applications of natural language processing. To the best of our knowledge, this is the first attempt to discover such knowledge automatically from a large corpus of text. We introduced the Extended Distributional Hypothesis, which states that paths in dependency trees have similar meanings if they tend to connect similar sets of words. Treating paths as binary relations, our algorithm is able to generate inference rules by searching for similar paths. Our experimental results show that the Extended Distributional Hypothesis can indeed be used to discover very useful inference rules, many of which, though easily recognizable, are difficult for humans to recall.

Many questions remain to be addressed. One is to recognize the polarity in inference relationships. High similarity values are often assigned to relations with opposite polarity. For example, "*X worsens Y*" has one of the highest similarity to "*X solves Y*" according to equation (6). For some questions, this may be helpful while for others it may cause confusion.

In another work (Lin and Pantel, 2001), we constructed semantic classes from text corpus with an unsupervised algorithm. For example, the following are two classes generated by our program (*Nq1446* and *Nq1471* are automatically generated class names):

Nq1446: coating, Resin, adhesive, sealant, plastic, material, chemical, polymer, product, "specialty chemical", paint, varnish, packaging, lubricant, ceramic, laminate, dye, film, glue, reagent, compound, pigment, wax, epoxy, sealer, lacquer, ink, gasket, covering, insulator

Nq1471: "booster rocket", booster, rocket, "rocket engine", engine, vehicle, motor, "propulsion system", tank, "fuel cell", injector

These classes may be used to extend paths with constraints on the inference rule's variables. For example, instead of generating a rule "*X manufactures Y ≈ X's Y factory*", we may want to generate a rule with an additional clause: "*X manufactures Y ≈ X's Y factory*, *where Y is in Nq1446 or Nq1471 or …*". The "*where*" clause can be potentially discovered by generalizing the intersection of the *SlotY* fillers of the two relations.

## References


Alshawi, H. and Carter, D. 1994. Training and Scaling Preference Functions for Disambiguation. Computational Linguistics, 20(4):635-648.

Anick, P.G. and Tipirneni, S. 1999. The Paraphrase Search Assistant: Terminological Feedback for Iterative Information Seeking. In *Proceedings of SIGIR-99*. pp. 153-159. Berkeley, CA.

Arampatzis, A. T., Tsoris, T., Koster, C. H. A., and van der Weide, T. P. 1998. Phrase-based infase-bas retrieval. *Information Processing & Management*, 34(6):693-707.

Barzilay, R., McKeown, K., and Elhadad, M. 1999. Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of ACL-99*. College Park, Maryland.

Berwick, R. C. 1991. Principles of Principle-Based Parsing. In Berwick, B. C., Abney, S. P., and Tenny, C. (eds.), *Principle-Based Parsing Computation and Psycholinguistics*. pp. 1-38. Kluwer Academic Publishers.

Charniak, E. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*. pp. 132-139. Seattle, WA.

Church, K. and Hanks, P. 1989. Word Association Norms, Mutual Information, and Lexicography. In *Proceedings of ACL-89*. pp. 76-83. Vancouver, Canada.

Collins, M. J. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of ACL-96*. pp. 184-191. Santa Cruz, CA.

Dras, M. 1997. Reluctant Paraphrase: Textual Restructuring Under an Optimisation Model. In *Proceedings of PACLING-97*. pp. 98-104. Ohme, Japan.

Dras, M. 1999. A meta-level Grammar: Redefining Synchronous TAGs for Translation and Paraphrase. In *Proceedings of ACL-99*. pp. 80-97. College Park, Maryland.

Fabre, C. and Jacquemin, C. 2000. Boosting Variant Recognition with Light Semantics. In *Proceedings of COLING-2000*. Sarrebrücken, Germany.

Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., Schler, Y., and Zamir, O. 1998. Text Mining at the Term Level. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*. pp. 65-73. Nantes, France.

Harabagiu, S., Pasca, M., and Maiorano, S. 2000. Experiments with Open-Domain Textual Question Answering. In *Proceedings of COLING-2000*. Sarrebrücken, Germany.

Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.), *The Philosophy of Linguistics*. New York: Oxford University Press. pp. 26-47.

Hahn, U. and Schnattinger, K. 1998. Towards text knowledge engineering. In *Proceedinfs of AAAI-98*. pp. 524-531. Menlo Park, California.

Hays, D. 1964. Dependency Theory: a Formalism and Some Observations. *Language*, 40:511-525.

Hearst, M. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of ACL-92*. Nantes, France.

Hindle, D. 1990. Noun Classification from Predicate-Argument Structures. In *Proceedings of ACL-90*. pp. 268-275. Pittsburgh, Pennsylvania.

Hudson, R. 1984. *Word Grammar*. Basil Blackwell Publishers Limited. Oxford, England.

Iordanskaja, L, Kittredge, R., and Polguere, A. 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer. Boston, MA.

Jacquemin, C., Klavans, J. L., and Tzoukermann, E. 1997. Expansion of Multi-Word Terms for Indexing and Retireval using Morphology and Syntax. In *Proceedings of ACL-97*. pp. 24-31. Madrid, Spain.

Jacquemin, C., Klavans, J. L., and Tzoukermann, E. 1999. NLP for Term Variant Extraction: A Synergy of Morphology, Lexicon, and Syntax. *Natural Language Information Retrieval*, T. Strzalkowski, editor. pp. 25-74. Kluwer. Boston, MA.

Larsen, B. and Aone, C. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of KDD-99*. pp. 16-22. San Diego, CA.

Lin, D. 1993. Parsing Without OverGeneration. In *Proceedings ACL-93*. pp. 112-120. Columbus, OH.

Lin, D. 1998. Extracting Collocations from Text Corpora. *Proceedings of the Workshop on Computational Terminology*. pp. 57-63. Montreal, Canada.

Lin, D. and Pantel, P. 2001. Induction of Semantic Classes from Natural Language Text. To appear in *Proceedings of KDD-2001*. San Francisco, CA.

Lin, S. H., Shih, C. S., Chen, M. C., et al. 1998. Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach. In *Proceedings of SIGIR-98*. Melbourne, Australia.

McCallum, A., Nigam, K., and Ungar, L. H. 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proceedings of KDD-2000*. Boston, MA.

Mel'čuk, I. A. 1987. *Dependency Syntax: theory and practice*. State University of New York Press. Albany, NY.

Meteer, M. M. and Shaked, V. 1988. Strategies for Effective Paraphrasing. In *Proceedings of COLING-88*. pp. 431-436 Budapest.

Miller, G. 1990. WordNet: An Online Lexical Database. *International Journal of Lexicography*, 1990.

Pereira, F., Tishby, N., and Lee, L. 1993. Distributional Clustering of English Words. In *Proceedings of ACL-93*. pp. 183-190. Columbus, Ohio.

Rajman, M. and Besançon, R. 1997. Text Mining: Natural Language Techniques and Text Mining Applications. In *Proceedings of the seventh IFIP 2.6 Working Conference on Database Semantics (DS-7)*.

Richardson, S. D. 1997. *Determining Similarity and the Infering Relations in a Lexical Knowledge-Base*. Ph.D. Thesis. The City University of New York.

Robin, J. 1994. *Revision-based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. Dissertation. Columbia University.

Sampson, G. 1995. *English for the Computer - The SUSANNE Corpus and Analytic Scheme*. Clarendon Press. Oxford, England.

Sparck Jones, K. and Tait, J. I. 1984. Automatic Search Term Variant Generation. *Journal of Documentation*, 40(1):50-66.