# LaTaT: Language and Text Analysis Tools

Dekang Lin
University of Alberta
Department of Computing Science
Edmonton, Alberta T6H 2E1 Canada
lindek@cs.ualberta.ca

## ABSTRACT

LaTaT is a Language and Text Analysis Toolset. This paper gives a brief description of the components comprising LaTaT, including a Minimalist parser and language and concept learning programs.

## 1. INTRODUCTION

In natural language processing, syntactic and semantic knowledge are deeply intertwined with each other, both in their acquisition and usage. The goal of our research is to build a syntactic and semantic knowledge base through an iterative process that involves both language processing and language acquisition. We start the process by parsing a large corpus with a manually constructed parser that has only syntactic knowledge. We then extract lexical semantic and statistical knowledge from the parsed corpus, such as similar words and phrases, collocations and idiomatic expressions, and selectional preferences. In the second cycle, the text corpus is parsed again with the assistance of the newly acquired semantic and statistical knowledge, which allows the parser to better resolve systematic syntactic ambiguities, removing unlikely parts of speech. Our hypothesis is that this will result in higher quality parse trees, which in turn allows extraction of higher quality semantic and statistical knowledge in the second and later cycles.

LaTaT is a Language and Text Analysis Toolset that demonstrates this iterative learning process. The main components in the toolset consist of the following:

- A broad coverage English parser, called Minipar. The grammar is constructed manually, based on the Minimalist Program (Chomsky 1995). Instead of using a large number of CFG rules, Minipar achieves its broad coverage by using a small set of principles to constrain the overgerating X-bar schema;
- A collocation extractor that extracts frequency counts of grammatical dependency relationships from a corpus parsed with Minipar. The frequency counts are then injected into Minipar to help it rank candidate parse trees;
- A thesaurus constructor (Lin, 1998) that automatically computes the word similarities based on the distributional characteristics of words in the parsed corpus. The resulting word similarity database can then be used to smooth the probability distribution in statistical language models (Dagan *et al*, 1997);
- A clustering algorithm that constructs Roget-like semantic categories in an unsupervised fashion (Lin and Pantel, 2001a); and
- An unsupervised learner to identify similar expressions from a parsed corpus (Lin and Pantel, 2001b).

## 2. Minipar

Minipar is a principle-based English parser (Berwick *et al*, 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 WordNet (Miller, 1990). With additional proper names, the lexicon contains about 130,000 entries (in base form). 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. A dependency relationship is an asymmetric binary relationship between a word called **head**, and another word called **modifier** (Mel'čuk, 1987). The structure of a sentence can be represented by a set of dependency relationships that form a tree. A word in the sentence may have several modifiers, but each word may modify at most one word. The root of the dependency tree does not modify any word. It is also called the head of the sentence.

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

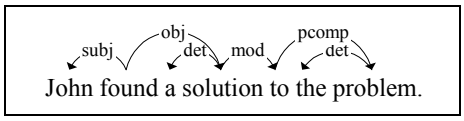


**Figure 1. Example dependency tree.**

**Table 1. A subset of 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 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. |

associated with the links represent types of dependency relations. Table 1 lists a subset of the dependency relations in Minipar outputs.

Minipar constructs all possible parses of an input sentence. However, only the highest ranking parse tree is outputted. Although the grammar is manually constructed, the selection of the best parse tree is guided by the statistical information obtained by parsing a 1GB 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.

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.

## 3. Collocation and Word Similarity

We define a collocation to be a dependency relationship that occurs more frequently than predicted by assuming the two words in the relationship are independent of each other. Lin (1998) presented a method to create a collocation database by parsing a large corpus. Given a word *w*, the database can be used to retrieve all the dependency relationships involving *w* and the frequency counts of the dependency relationships. Table 2 shows excerpts of the entries in the collocation database for the words *duty* and *responsibility*. For example, in the corpus from which the collocation database is constructed, *fiduciary duty* occurs 319 times and *assume [the] responsibility* occurs 390 times.

The collocation database entry of a given word can be viewed as a feature vector for that word. Similarity between words can be computed using the feature vectors. Intuitively, the more features that are shared between two words, the higher the similarity between the two words will be. This intuition is captured by the Distributional Hypothesis (Harris, 1985).

Features of words are of varying degree of importance. For example, while almost any noun can be used as object of *include*, very few nouns can be modified by *fiduciary*. Two words sharing the feature *object-of-include* is less indicative of their similarity

**Table 2. Excerpts of entries in the collocation database for *duty* and *responsibility*.**

| | DUTY | | RESPONSIBILITY |
|---|---|---|---|
| modified-by adjectives | fiduciary 319, active 251, other 82, official 76, additional 47, administrative 44, military 44, constitutional 41, reserve 24, high 23, moral 21, double 16, day-to-day 15, normal 15, specific 15, assigned 14, extra 13, operating 13, temporary 13, corporate 12, peacekeeping 12, possible 12, regular 12, retaliatory 12, heavy 11, routine 11, sacred 11, stiff 11, congressional 10, fundamental 10, hazardous 10, main 10, patriotic 10, punitive 10, special 10, … | modified-by adjectives | more 107, full 92, fiduciary 89, primary 88, personal 79, great 69, financial 64, fiscal 59, social 59, moral 48, additional 46, ultimate 39, day-to-day 37, special 37, individual 36, legal 35, other 35, corporate 30, direct 30, constitutional 29, given 29, overall 29, added 28, sole 25, operating 23, broad 22, political 22, heavy 20, main 18, shared 18, professional 17, current 15, federal 14, joint 14, enormous 13, executive 13, operational 13, similar 13, administrative 10, fundamental 10, specific 10, … |
| object-of verbs | have 253, assume 190, perform 153, do 131, impose 118, breach 112, carry out 79, violate 54, return to 50, fulfill 44, handle 42, resume 41, take over 35, pay 26, see 26, avoid 19, neglect 18, shirk 18, include 17, share 17, discharge 16, double 16, relinquish 16, slap 16, divide 14, split 13, take up 13, continue 11, levy 11, owe 10, … | object-of verbs | have 747, claim 741, take 643, assume 390, accept 220, bear 187, share 103, deny 86, fulfill 53, meet 48, feel 47, retain 47, shift 47, carry out 45, take over 41, shoulder 29, escape 28, transfer 28, delegate 26, give 25, admit 23, do 21, acknowledge 20, exercise 20, shirk 20, divide 19, get 19, include 19, assign 18, avoid 17, put 17, recognize 17, hold 16, understand 16, evade 15, disclaim 12, handle 12, turn over 12, become 11, expand 11, relinquish 11, show 11, violate 11, discharge 10, duck 10, increase 10, … |

than if they shared the feature *modified-by-fiduciary*. The similarity measure proposed in (Lin, 1998) takes this into account by computing the mutual information between two words involved in a dependency relationship.

Using the collocation database, (Lin, 1998) presented an unsupervised method to construct a similarity matrix. Given a word *w*, the matrix returns a set of similar words of *w* along with their similarity to *w*. For example, the 35 most similar words of *duty, Beethoven*, and *eat* are shown in Table 3. The similarity matrix consists of about 20,000 nouns, 4,000 verbs and 6,000 adjectives and adverbs.

**Table 3. The top 35 most similar words of *duty, Beethoven* and *eat* as given by (Lin, 1998).**

| WORD | SIMILAR WORDS (WITH SIMILARITY SCORE) |
|---|---|
| DUTY | responsibility 0.182, obligation 0.138, job 0.127, function 0.121, post 0.121, task 0.119, role 0.116, assignment 0.114, mission 0.109, requirement 0.109, tariff 0.109, position 0.108, restriction 0.103, procedure 0.101, tax 0.101, salary 0.1, fee 0.099, training 0.097, commitment 0.096, penalty 0.095, burden 0.094, quota 0.094, work 0.093, staff 0.093, regulation 0.093, sanction 0.093, liability 0.092, personnel 0.092, service 0.091, action 0.09, activity 0.09, rule 0.089, practice 0.089, authority 0.088 |
| BEETHOVEN | Mozart 0.193, Brahms 0.178, Schubert 0.148, Mahler 0.143, Bach 0.142, Tchaikovsky 0.128, Prokofiev 0.118, Wagner 0.089, chamber music 0.087, Handel 0.073, cello 0.069, classical music 0.067, Strauss 0.066, Shakespeare 0.063, concerto 0.062, Cole Porter 0.062, Verdi 0.06, Sonata 0.057, violin 0.056, Elvis 0.053, Berg 0.053, composer 0.053, Lenin 0.052, flute 0.049, Bernstein 0.047, jazz 0.047, Beatles 0.046, Frank Sinatra 0.045, Warhol 0.043, Bob Dylan 0.043, Napoleon 0.043, symphony 0.042, solo 0.042, tough guy 0.042, Bruce Springsteen 0.041, grandparent 0.041 |
| EAT | drink 0.204, cook 0.193, smoke 0.164, sleep 0.162, consume 0.156, love 0.153, enjoy 0.152, pick up 0.142, look at 0.141, feed 0.141, wear 0.14, talk about 0.139, watch 0.138, forget 0.136, like 0.136, taste 0.134, go out 0.133, sit 0.133, pack 0.133, wash 0.132, stay 0.131, burn 0.13, serve 0.129, ride 0.128, pick 0.128, grab 0.128, freeze 0.126, go through 0.126, throw 0.126, remember 0.124, get in 0.123, feel 0.123, learn 0.123, live 0.123 |

## 4. Unsupervised Induction of Semantic Classes

Consider the similar words of *Beethoven*. The quality of similar words obviously decreases as the similarity value decreases. Some of the words have non-zero similarity simply because they share common features with *Beethoven* by accident. For example, *tough guy* is similar to *Beethoven* because both *Beethoven* and *tough guy* can be used as the object of the verb *play*.

The similar words of *duty* exemplify another problem: The top similar words of a given word may be similar to different senses of the word. However, this is not made explicit by the similarity matrix.

LaTaT includes an algorithm called UNICON (UNsupervised Induction of CONcepts) that clusters similar words to create semantic classes (Lin and Pantel, 2001a). UNICON uses a heuristic maximal-clique algorithm, called CLIMAX, to find clusters in the similar words of a given word. The purpose of CLIMAX is to find small, tight clusters. For example, two of the clusters returned by CLIMAX are:

```
(Nq34
  "Harvard University"            0.610996
  Harvard                         0.482834
  "Stanford University"           0.469302
  "University of Chicago"         0.454686
  "Columbia University"           0.44262
  "New York University"           0.436737
  "University of Michigan"        0.43055
  "Yale university"               0.416731
  MIT                             0.414907
  "University of Pennsylvania"    0.384016
  "Cornell University"            0.333958
)
(Nq184
  "University of Rochester"       0.525389
  "University of Miami"           0.466607
  "University of Colorado"        0.46347
  "Ohio State University"         0.430326
  "University of Florida"         0.398765
  "Harvard Medical School"        0.39485
  "University of North Carolina"  0.394256
  "University of Houston"         0.371618
)
```

*Nq34* and *Nq184* are automatically generated names for the clusters. The number after each word in the clusters is the similarity between the word and the centroid of that cluster.

The UNICON algorithm computes the centroids of a cluster by averaging the collocational features of the words in the cluster. The CLIMAX algorithm is then recursively used to construct clusters of centroids and the clusters whose centroids are clustered together are merged. This process continues until no more clusters are merged. The details of the UNICON and CLIMAX algorithms are presented in (Lin and Pantel, 2001a). Table 4 shows 10 sample semantic classes identified by the UNICON algorithm, using a 1GB newspaper text corpus.

## 5. Automatic Discovery of Inference Rules

In many natural language processing and information retrieval applications, it is very useful to know the paraphrase relationships between natural language expressions. LaTaT includes an unsupervised method for discovering paraphrase inference rules from text, such as "*X is author of Y ≈ X wrote Y*", "*X solved Y ≈ X found a solution to Y*", and "*X caused Y ≈ Y is triggered by X*" (Lin and Pantel, 2001b). Our algorithm is based on an extended version of Harris' Distributional Hypothesis. Instead of using this hypothesis on words, we apply it to paths in the dependency trees of a parsed corpus.

Table 4. Ten concepts discovered by UNICON.

| CONCEPT | SIZE | MEMBERS |
|---|---|---|
| *Nq1* | 210 | "Max von Sydow", "Paul Newman", "Jeremy Irons", "Lynn Redgrave", "Lloyd Bridges", "Jack Lemmon", "Jaclyn Smith", "Judd Nelson", "Beau Bridges", "Raymond Burr", "Gerald McRaney", "Robert de Niro", "Tim Matheson", "Kevin Costner", "Kurt Russell", "Arnold Schwarzenegger", "Michael J. Fox", "Dustin Hoffman", "Tom Hanks", "Robert Duvall", "Michael Keaton", "Edward James Olmos", "John Turturro", "Robin Williams", "Sylvester Stallone", "John Candy", "Whoopi Goldberg", "Eddie Murphy", "Rene Auberjonois", "Vanessa Redgrave", "Jeff Bridges", "Robert Mitchum", "Clint Eastwood", "James Woods", "Al Pacino", "William Hurt", "Richard Dreyfuss", "Tom Selleck", "Barry Bostwick", "Harrison Ford", "Tom Cruise", "Jon Cryer", "Pierce Brosnan", "Donald Sutherland", "Anthony Quinn", "Farrah Fawcett", "Louis Gossett Jr.", "Mark Harmon", "Steven Bauer", "William Shatner", "Diane Keaton", "Billy Crystal", "Omar Sharif", "Paul Hogan", "Woody Allen", "Fred Savage", "Jodie Foster", "Chuck Norris", "Kirk Douglas", "Glenn Close", "Ed Asner", "Dan Aykroyd", "Steve Guttenberg", "Sissy Spacek", "Jonathan Pryce", "Sean Penn", "Bill Cosby", "Robert Urich", "Steve Martin", "Karl Malden", "John Lithgow", "Charles Bronson", "Danny DeVito", "Michael Douglas", "John Ritter", "Gerard Depardieu", "Val Kilmer", "Jamie Lee Curtis", "Randy Quaid", "John Cleese", "James Garner", "Albert Finney", "Richard Gere", "Jim Belushi", "Christopher Reeve", "Telly Savalas", "Chevy Chase".... |
| *Nq178* | 39 | Toyota, Honda, Volkswagen, Mazda, Oldsmobile, BMW, Audi, Mercedes-Benz, Cadillac, Volvo, Subaru, Chevrolet, Mercedes, Buick, Porsche, Nissan, VW, Mitsubishi, Renault, Hyundai, Isuzu, Jaguar, Suzuki, Dodge, Rolls-Royce, Pontiac, Fiat, Chevy, Saturn, Yugo, Ferrari, "Mercedes Benz", Plymouth, mustang, Beretta, Panasonic, Corvette, Nintendo, Camaro |
| *Nq214* | 41 | mathematics, physic, math, "political science", chemistry, "computer science", biology, sociology, "physical education", "electrical engineering", anthropology, astronomy, "social science", geology, psychology, "mechanical engineering", physiology, geography, economics, psychiatry, calculus, biochemistry, algebra, science, civics, journalism, literature, theology, "molecular biology", humanity, genetics, archaeology, nursing, anatomy, pathology, arithmetic, pharmacology, literacy, architecture, undergraduate, microbiology |
| *Nq223* | 59 | shirt, jacket, dress, pant, skirt, coat, sweater, T-shirt, hat, blouse, jean, trouser, sock, gown, scarf, slack, vest, boot, uniform, shoe, robe, cloth, sunglasses, clothing, outfit, glove, underwear, sneaker, blazer, jersey, costume, wig, mask, helmet, button, hair, collar, ribbon, short, belt, necktie, bra, stocking, sleeve, silk, red, pin, banner, badge, sheet, sticker, makeup, stripe, bow, logo, linen, curtain, shade, quilt |
| *Nq292* | 31 | barley, oat, sorghum, "feed grain", alfalfa, "soybean meal", "soybean oil", "sugar beet", maize, sunflower, "pork belly", soybean, millet, Rye, oilseed, wheat, "grain sorghum", rapeseed, canola, hay, "palm oil", durum, safflower, psyllium, "sunflower seed", flaxseed, bran, broiler, buckwheat, cantaloupe, cottonseed |
| *Nq293* | 22 | "Joseph Cicippio", "Terry Anderson", "Terry Waite", Cicippio, Waite, "Terry A. Anderson", "William Higgins", "John McCarthy", "Joseph James Cicippio", "Thomas Sutherland", "Brian Keenan", "Alann Steen", "Jesse Turner", "Alec Collett", "Edward Austin Tracy", "Edward Tracy", "Frank Reed", "American Terry Anderson", "Jack Mann", Buckley, westerner, "Giandomenico Picco", "Robert Polhill", "Benjamin Weir" |
| *Nq352* | 8 | heroin, cocaine, marijuana, narcotic, alcohol, steroid, crack, opium |
| *Nq356* | 15 | Saskatchewan, Alberta, Manitoba, "British Columbia", Ontario, "New Brunswick", Newfoundland, Quebec, Guangdong, "Prince Edward Island", "Nova Scotia", "Papua New Guinea", "Northwest Territories", Luzon, Mindanao |
| *Nq396* | 29 | sorrow, sadness, grief, anguish, remorse, indignation, insecurity, loneliness, discomfort, agony, despair, regret, heartache, dismay, shame, revulsion, angst, jubilation, humiliation, bitterness, pity, outrage, anxiety, empathy, happiness, mourning, letdown, distaste, indignity |
| *Nq776* | 30 | baldness, hemophilia, acne, infertility, sepsis, "cold sore", "sleeping sickness", "morning sickness", "kidney stone", "common cold", heartburn, "eye disease", "heroin addiction", osteoporosis, "pneumocystis carinii pneumonia", dwarfism, incontinence, "manic depression", atherosclerosis, "Dutch elm disease", hyperthyroidism, discoloration, "cancer death", spoilage, gonorrhea, hemorrhoid, wart, mildew, sterility, "athlete's foot" |

In the dependency trees generated by Minipar, 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 **root** of the path 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* and *problem* fills the *SlotY* in the above example.

We extract the fillers and frequency counts of all the slots of all the paths in a parsed corpus. Table 5 shows an excerpt of the fillers of two paths. The underlying assumption of algorithm is that when the meanings of paths are similar, their corresponding sets of fillers share a large number of common words.

Richardson (1997) 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.

We use the notation $|p, SlotX, w|$ to denote the frequency count of word $w$ filling in the *SlotX* of a path $p$, and $|p, SlotX, *|$ to denote $\sum_{w} |p, SlotX, w|$, and $|*, *, *|$ to denote $\sum_{p,s,w} |p, s, w|$.

Following (Lin, 1998), the mutual information between a path slot and its filler can be computed by the formula:

$$mi(p, Slot, w) = \log\left(\frac{|p, Slot, w| \times |*, Slot, *|}{|p, Slot, *| \times |*, Slot, w|}\right) \quad \textbf{(1)}$$

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

$$sim(slot_1, slot_2) = \frac{\sum_{w \in T(p_1,s) \cap T(p_2,s)} mi(p_1, s, w) + mi(p_2, s, w)}{\sum_{w \in T(p_1,s)} mi(p_1, s, w) + \sum_{w \in T(p_2,s)} mi(p_2, s, w)} \quad \textbf{(2)}$$

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:

$$S(p_1, p_2) = \sqrt{sim(SlotX_1, SlotX_2) \times sim(SlotY_1, SlotY_2)} \quad \textbf{(3)}$$

Table 6 and 7 list the top-50 most similar paths to "*X solves Y*". and "*X causes Y*" generated by our algorithm. The ones tagged with an asterisk (*) are incorrect. Most of the paths can be considered as paraphrases of the original expression.

**Table 5. 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 |
| legislator | budget deficit | researcher | mystery |
| sheriff | dispute | sheriff | murder |

**Table 6. The top-50 most similar paths to *"X solves Y".***

| | |
|---|---|
| *Y* is solved by *X* | *X* clears up *Y* |
| *X* resolves *Y* | **X* creates *Y* |
| *X* finds a solution to *Y* | **Y* leads to *X* |
| *X* tries to solve *Y* | **Y* is eased between *X* |
| *X* deals with *Y* | *X* gets down to *Y* |
| *Y* is resolved by *X* | *X* worsens *Y* |
| *X* addresses *Y* | *X* ends *Y* |
| *X* seeks a solution to *Y* | **X* blames something for *Y* |
| *X* do something about *Y* | *X* bridges *Y* |
| *X* solution to *Y* | *X* averts *Y* |
| *Y* is resolved in *X* | **X* talks about *Y* |
| *Y* is solved through *X* | *X* grapples with *Y* |
| *X* rectifies *Y* | **X* leads to *Y* |
| *X* copes with *Y* | *X* avoids *Y* |
| *X* overcomes *Y* | *X* solves *Y* problem |
| *X* eases *Y* | *X* combats *Y* |
| *X* tackles *Y* | *X* handles *Y* |
| *X* alleviates *Y* | *X* faces *Y* |
| *X* corrects *Y* | *X* eliminates *Y* |
| *X* is a solution to *Y* | *Y* is settled by *X* |
| *X* makes *Y* worse | **X* thinks about *Y* |
| *X* irons out *Y* | *X* comes up with a solution to *Y* |
| **Y* is blamed for *X* | *X* offers a solution to *Y* |
| *X* wrestles with *Y* | *X* helps somebody solve *Y* |
| *X* comes to grip with *Y* | **Y* is put behind *X* |

## 6. References


Berwick R., Abney S., and Tenny, C, editors. *Principle-Based Parsing: Computation and Psycholinguistics*. Kluwer Academic Publishers, 1991.

Chomsky N. 1995. *Minimalist Program.* MIT Press.

Dagan I, Lee L, and Pereira F., Similarity-based Methods for Word Sense Disambiguation. In *Proceedings of ACL/EACL-97,* pp.56-63. Madrid, Spain.

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

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

Lin, D. and Pantel, P. 2001b. DIRT: Discovery of Inference Rules from Text. To appear in Proceedings of KDD-2001. San Francisco, CA.

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

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

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

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

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

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

**Table 7. The top-50 most similar paths to "*X causes Y*".**

| | |
|---|---|
| *Y* is caused by *X* | **Y* contributes to *X* |
| *X* cause something *Y* | **X* results from *Y* |
| *X* leads to *Y* | **X* adds to *Y* |
| *X* triggers *Y* | *X* means *Y* |
| **X* is caused by *Y* | **X* reflects *Y* |
| **Y* causes *X* | *X* creates *Y* |
| *Y* is blamed on *X* | **Y* prompts *X* |
| *X* contributes to *Y* | *X* provoke *Y* |
| *X* is blamed for *Y* | *Y* reflects *X* |
| *X* results in *Y* | *X* touches off *Y* |
| *X* is the cause of *Y* | *X* poses *Y* |
| **Y* leads to *X* | *Y* is sparked by *X* |
| *Y* results from *X* | **X* is attributed to *Y* |
| *Y* is result of *X* | **Y* is cause of *X* |
| *X* prompts *Y* | **X* stems from *Y* |
| *X* sparks *Y* | **Y* is blamed for *X* |
| **Y* triggers *X* | **X* is triggered by *Y* |
| *X* prevents *Y* | *Y* is linked to *X* |
| **X* is blamed on *Y* | *X* sets off *Y* |
| *Y* is triggered by *X* | *X* is a factor in *Y* |
| *Y* is attributed to *X* | *X* exacerbates *Y* |
| *X* stems from *Y* | *X* eases *Y* |
| **Y* results in *X* | *Y* is related to *X* |
| **X* is result of *Y* | *X* is linked to *Y* |
| *X* fuels *Y* | *X* is responsible for *Y* |