Lexical Discovery with an Enriched Semantic Network

Doug Beeferman
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
`dougb@cs.cmu.edu`

Abstract

The study of lexical semantics has produced a systematic analysis of binary relationships between content words that has greatly benefited lexical search tools and natural language processing algorithms. We first introduce a database system called FreeNet that facilitates the description and exploration of finite binary relations. We then describe the design and implementation of Lexical FreeNet, a semantic network that mixes WordNet-derived semantic relations with data-derived and phonetically-derived relations. We discuss how Lexical FreeNet has aided in lexical discovery, the pursuit of linguistic and factual knowledge by the computer-aided exploration of lexical relations.

1 Motivation

This paper discusses Lexical FreeNet, a database system designed to enhance *lexical discovery*. By this we mean the pursuit of linguistic and factual knowledge with the computer-aided exploration of lexical relations. Lexical FreeNet is a semantic network that leverages WordNet and other knowledge and data sources to facilitate the discovery of nontrivial lexical connections between words and concepts.

A semantic network allied with the proper user interface can be a useful tool in its own right. By organizing words semantically rather than alphabetically, WordNet provides a means by which users can navigate its vocabulary logically, establishing connections between concepts and not simply character sequences. Exploring the WordNet *hyponym* tree starting at the word `mammal`, for instance, reveals to us that aardvarks are mammals; exploring WordNet's *meronym* relation at the word `mammal` reveals to us that mammals have `hair`. From these two explorations we can accurately conclude that aardvarks have hair.

Lexical exploration need not be limited to one step at a time, however. Viewing a semantic network as a computational structure awaiting graph-theoretic queries gives us the freedom to demand services beyond mere lookup. "Does the aardvark have hair?", or "What is the closest connection between aardvarks and hair?" or "How interchangably can the words `aardvark` and `anteater` be used?" are all reasonable questions with answers staring us in the face. Of course, the idea of finding shortest paths in semantic networks (through so-called *activation-spreading* or *intersection search*) is not new. But these questions have typically been asked of very limited graphs, networks for domains far narrower than the lexical space of English, say. We feel that formalizing how WordNet can be employed for this broader sort of lexical discovery is a good start. We also feel that it is necessary first to enrich the network with information that, as we shall see, cannot be easily gleaned from WordNet's current battery of relations. The very large electronic corpora and wide variety of linguistic resources that today's computing technology has enabled will in turn enable this.

The remainder of this paper is organized as follows. We shall first describe in Section 2 the FreeNet database system for the expression and analysis of relational data. In Section 3 we'll describe the design and construction of an instance of this database called Lexical FreeNet. We'll conclude by providing examples of applications of Lexical FreeNet to lexical discovery.

2 FreeNet

FreeNet, an acronym for *finite relation expression network*, is a system for describing and exploring finite binary relations. Here we mean relation in the mathematical sense, *i.e.* a set of ordered pairs. We concern ourselves with finite sets of pairs of tokens drawn from a finite set of tokens, or *vocabulary*.

2.1 Tokens and relations

A *token* in FreeNet is simply a normalized string of characters drawn from a finite vocabulary. The vocabulary might be a dictionary of English words, a set of movie titles, or a set of names of researchers. The system is assumed to implement normalization as a function from input strings to strings.

A *relation* in FreeNet is a finite set of ordered pairs of tokens, or *links*. Each relation has a name that, like a token, is simply a normalized string of characters drawn from a finite vocabulary (which we shall do better to call an *alphabet*, for reasons made clear below.)

Use of the FreeNet system can be seen to consist of three distinct processing phases: the *relation computation* stage, in which a set of relations is derived from some knowledge or data source and transduced to an explicit set of labeled ordered pairs; the *graph construction* stage, in which this set of labeled pairs is transduced to an efficient multigraph representation; and the *query* stage, in which a user can interact with the system to find paths in the multigraph that match a certain specification.

FreeNet consists of software to do the second and third phases. Implementation of a specific instance of FreeNet requires the user to write software to do the first phase, but support software exists for an optional *filtering* substage that constrains the input pair set in certain ways—eliminating pairs that contain stopwords, enforcing limits on the fanout of tokens, and enforcing strength thresholds, for instance.

The second phase, graph building, simply entails providing a set of triples (two tokens and a relation) to the system. The order in which the triples appear in the input does not matter, as it is the program's responsibility to reorder the links as necessary and to store the graph efficiently.

The third phase, querying, is the chief novel contribution, and is described below.

2.2 Regular expressions

The power behind FreeNet lies in the user's ability to compose primitive relations to build more complex relations that it may use in its queries.

The primary mechanism for building complex relations is the *regular expression* over the alphabet of relation names. Just as a regular expression over ASCII characters specifies a regular set of strings recursively in terms of other sets, so too can a regular expression over relation names specify a set of ordered pairs recursively in terms of other sets and various operators.

The following grammar specifies allowable regular expressions in FreeNet.

```
regexp =

    <rel>                   (relation name)
  | (regexp)                (parenthesization)

  | regexp regexp           (concatenation)
  | regexp | regexp         (union)
  | regexp & regexp         (conjunction)
  | regexp*                 (transitive closure)
  | regexp'                 (inverse)
  | regexp-                 (complement)
  | regexp%                 (sibling)
```

These regexp-building operators are described below.

Concatenation

The concatenation operator is used to compose two relations directly. The expression `r1 r2` denotes the set of pairs (a, b) such that for some token c, $(a, c) \in$ r1 and $(c, b) \in$ r2. For example, a network implementing a genealogy database might offer primitive `parent` and `brother` relations. In that case, the relation denoted by the regular expression (`parent brother`) is what we know of as the `uncle` relation.

Conjunction

Conjunction takes the intersection of two relations: plainly, the intersection of their respective pair sets. The expression `r1 & r2` denotes the set of pairs (a, b) such that $(a, b) \in$ r1 and $(a, b) \in$ r2.

Supposing that in a lexical semantic net we have the relations `required_by` and `requires`, then a symmetric `symbiotic_with` relation might be implemented as their conjunction.

Union

The union operator is used to join two relations. The expression `r1 | r2` denotes the set of pairs (a, b) such that $(a, b) \in$ r1 or $(a, b) \in$ r2. In an Erdös-number like application, for example, two authors may be "related" if they have coauthored a paper *or* if one has cited the other.

Transitive closure

We commonly reason about the transitive closure of relations. The transitive closure operator implements homogeneous reachability—is there a path between the tokens using links only of a certain type? Namely, let `r^1` denote the relation `r` and `r^i` for $i > 1$ denote the relation `(r r^(i-1))`. Then `r*` denotes the union of all `r^i` as i ranges from 0 to infinity. (Note that since we assume finite relations, this set is always finite.) In the genealogy example, `parent*` would be what we consider the "ancestor" relation.

Inverse, Complement, and Sibling

A few more unary operators are minor conveniences in building relations. The inverse operator swaps every pair: `r-` denotes the set of pairs (a, b) such that $(b, a) \in$ r. Taking the union of a relation with its inverse produces a new relation that is guaranteed to be symmetric.

The complement operator produces a set containing all pairs *but* those in a certain relation. `r'` denotes the set of pairs (a, b) such that $(b, a) \not\in$r. (The vocabulary is assumed to be fixed after the graph is built, and so the universe is well-defined.)

The sibling operator produces pairs that have in common their relation with a certain other token. `r%` denotes the set of pairs (a, b) such that $a \neq b$ and there exists a c such that $(a, c) \in$ r and $(b, c) \in$ r. Thus `(parent-)%` relation is the genealogical sibling relation formed by applying the inverse operator and then the sibling operator to the "parent" relation.

Note

A simple structural induction can be used to prove that any relation built from these operators is also a relation. Additional operators to support set addition and subtraction of constant pair sets are also available.

2.3 Queries

Queries in FreeNet are *path specifications* expressed as a sequence of tokens or token variables with interleaved relation regexps. More precisely, every query is of the form `(W <regexp>)* W`, where `W` is either a *constant token* or a *variable* w_i, and `<regexp>` is a regular expression over relations, as defined above.

FreeNet returns a shortest path (or all paths) in the multigraph that match the query, binding the variables in the query to concrete tokens. The output includes the names of all of the primitive relation links traversed.

Queries in the Internet version of FreeNet can take one of four forms, each parameterized by one or two tokens; but these demonstrate what are expected to be common queries. Below, the "ANY" regexp is the union of all available (or selected) primitive relations. The comma (",") represents the universal relation, linking all pairs of tokens; the comma relation can thus be used in FreeNet queries to implement conjunction of clauses.

- *Shortest path*: This query takes two arguments s and t, and outputs the result of the query "s `ANY*` t". This finds a shortest path, using any of the selected relations, between the source and the target.
- *Fanout*: This query takes a single argument s and outputs the result of "s `ANY` w_1". This simply shows all words related in some way to the source.
- *Intersection search*: This query takes two arguments s and t and outputs the result of "s `ANY` w_1 `,` t `ANY` w_1". This is useful for finding what two tokens "have in common" in terms of primitive relationships with other tokens. The two relations involved in such a path need not be identical.
- *Coercion*: This query takes two arguments s and t, two relations `re1` and `re2`, and outputs the result of "s `re1` w_1 `re2` w_2 `re1` t". This is useful for a wide variety of constraint-solving, such as, in the lexical semantic net case, pun and rhyme generation.

2.4 Implementation issues

A FreeNet multigraph is stored sparsely for efficient offline (disk) access as a list of variable-length adjacency lists. Each element in an adjacency list is a single 32-bit word that describes an arc by combining its destination token ID and relation ID; the source token ID for an arc is implicit in its row. An index of offsets into the list is precomputed and stored together with hash tables for the token and relation namespaces. At no point in query processing is more than a single line of the list (equivalently, a set of links emanating from the same source node) in memory at once.

Graph construction

A number of optimizations in the layout of the multigraph on disk are essential if arbitrary searches over large multigraphs are to be efficient. Of particular concern is disk seek time, because traversing the graph entails accessing different rows of the adjacency list representation in rapid succession. One simple preprocessing step is to sort each row of the representation by the word identifier's row location, so that all of the nodes emanating from a fixed source can be accessed wixth a unidirectional sweep.

A trickier concern is the ordering of the rows themselves. We desire to order the rows so that related words tend to appear near each other so that seek time between them is minimized. We can formalize this problem by asking for an ordering that minimizes the average offset difference between a randomly chosen edge in the multigraph. This problem is at least as computationally hard as the well-studied, NP-complete *bandwidth* problem in graph theory (Papadimitriou, 1976), which is to find a linear ordering of the vertices of a given graph such that the maximum difference in the ordering between any two adjacent vertices is minimal. We are studying approximation algorithms (Blum et al., to appear) that allow this preprocessing step to be carried out efficiently during database construction.

Querying

Supporting arbitrary FreeNet queries that allow the full range of regular expression operators, is a nontrivial data structures problem, because it is prohibitively expensive to add new links with the occurrence of a new regexp. Instead, the graph is static. Each relation in the "alphabet" of relations is converted to an ASCII character, and stock regexp processing software is used to convert each regexp in a query to a state machine. A query is converted to a single state machine by concatenating its constituent regexp state machines, interleaving "constraint points" that enforce the identity of multiple bindings of the same variable. A dynamic set of state IDs and backtrace IDs is associated with each token to support breadth-first search.

The query templates above are implemented without all this machinery, by simply performing breadth-first-search on the graph, maintaining a single backtrace ID for each node, and allowing or prohibiting certain relations as specified by the user. Coercion is implemented as a hard-coded path constraint.

3 Lexical FreeNet

Lexical FreeNet is an instance of FreeNet supporting a range of lexical semantic applications. It achieves this by mixing statistically-derived and knowledge-derived relations.

Tokens

The tokens in Lexical FreeNet are the words that appear in at least one of the program's various data sources. This includes over 130,000 words from the CMU Pronouncing Dictionary v1.6d (CMU, 1997), 160,000 words and multiple-word phrases from WordNet 1.6, and 60,000 words from the broadcast news transcripts used to train the trigger relation. The intersection between these three sources is significant, of course, and in total there are slightly under 200,000 distinct tokens, including phrases.

(a)

Database breakdown by relation		
Relation	**Symbol**	**Number of links**
Triggers	**TRG**	354800
Synonymous	**SYN**	249156
Generalizes	**GEN**	261275
Specializes	**SPC**	261281
Comprises	**COM**	23650
Part of	**PAR**	23650
Antonym of	**ANT**	18982
Rhymes	**RHY**	4533536
Sounds like	**SIM**	1483360
Anagram	**ANA**	91072
197598 distinct tokens		

(b)

	TRG	**SYN**	**GEN**	**SPC**	**COM**	**PAR**	**ANT**	**RHY**	**SIM**	**ANA**
TRG	354800									
SYN	1873	249156								
GEN	972	3390	261275							
SPC	1164	3390	1628	261281						
COM	330	1805	150	263	23650					
PAR	391	1805	263	150	310	23650				
ANT	469	38	28	28	5	5	18982			
RHY	1279	1576	1042	1042	30	30	1576	4533536		
SIM	10129	1518	150	150	15	15	92	587015	1483360	
ANA	363	1338	18	18	25	25	0	7871	9946	91072

Figure 1: Statistics on the relations in Lexical FreeNet. (a) The number of links in each relation. (b) Relation crossover counts. Each cell reports the number of word pairs that exist in both relations. One of the 5 pairs counted in the cell at (`ANT`, `COM`), for example, is (`DAY`, `NIGHT`).

Relations

Lexical FreeNet includes seven *semantic* relations, two *phonetic* relations, and one *orthographic* relation. These relations connect the token set with about seven million links, costing 30 MB of disk space. A summary of the relations is shown in Figure 1. Below we use a bidirectional arrow ($\Longleftrightarrow$) to indicate a symmetric relation, and a unidirectional arrow ($\Longrightarrow$) to indicate an assymetric relation.

"Synonym of" ($\stackrel{\mathrm{SYN}}{\Longleftrightarrow}$)

This relation is computed by taking, for each synonym set (or *synset*) in all WordNet 1.6 word categories, the cross-product of the synonym set with itself, excluding reflexive links (self-loops). That is to say, we include all pairs of lexemes in each synset except the links from a lexeme to itself. Thus we mix different lexeme senses into the same soup, conflating, for example, the noun and verb senses of `BIKE` in `bike` $\stackrel{\mathrm{SYN}}{\Longleftrightarrow}$ `bicycle` and `bike` $\stackrel{\mathrm{SYN}}{\Longleftrightarrow}$ `pedal`.

"Triggers" ($\stackrel{\mathrm{TRG}}{\Longrightarrow}$)

Trigger pairs are ordered word pairs that co-occur significantly in data; that is, they are pairs that appear near each other in text more frequently than would be expected if the words were unrelated. Given a large corpus of text data, we built the assymetric trigger relation by finding the pairs in the cross-product of the vocabulary that have the highest average *mutual information*, as in (Rosenfeld, 1994; Beeferman et al., 1997). Mutual information is one measure of whether an observed co-occurrence of two vocabulary words is not due to chance. Word pairs with high mutual information are likely to be semantically related in some way.

We chose 160 million words of Broadcast News data (LDC, 1997) for this computation, and defined co-occurrence as "occurring within 500 words", approximately the average document length. We selected the top 350,000 trigger pairs from the ranking to use in the relation, putting the size of the relation on par with the synonym relation.[1] Some of the top trigger pairs discovered by this procedure are shown in Table 2. In our implementation we limit the number of trigger links emanating from a token to the top 50, and prune away links that include any member of a handcoded stopword set that includes function words.

s	t
`Los`	`Angeles`
`United`	`States`
`White`	`House`
`President`	`Clinton`
`New`	`York`
`health`	`care`
...	...
`campaign`	`Bush`
`Haitian`	`Aristide`
`films`	`film`
`fed`	`rates`
`court`	`evidence`
`care`	`insurance`

Figure 2: The top six trigger pairs (s, t), ranked by mutual information, in the Lexical FreeNet trigger relation, and the 500th through 505th-ranked pairs. The highest-ranked pairs tend to be distance-one bigram phrases, while the remainder co-occur at greater distances.

"Specializes" ($\stackrel{\mathrm{SPC}}{\Longrightarrow}$) and "Generalizes" ($\stackrel{\mathrm{GEN}}{\Longrightarrow}$)

The specialization relation captures the lexical inheritance system underlying WordNet nouns (Miller, 1990) and verbs (Fellbaum, 1990). It is computed by taking, for each pair of WordNet synsets that appear as parent and child in the WordNet *hyponym* trees, the cross-product of the pair. For example, `shoe` $\stackrel{\mathrm{SPC}}{\Longrightarrow}$ `footwear`.

The generalization relation is simply the inverse of specialization relation, or `SPC-`. For example: `tree` $\stackrel{\mathrm{GEN}}{\Longrightarrow}$ `cypress`.

[1] We used the Trigger Toolkit, available at `http://www.cs.cmu.edu/ aberger/software.html, for this computation`

"Part of" ($\overset{\text{PAR}}{\Longrightarrow}$) and "Comprises" ($\overset{\text{COM}}{\Longrightarrow}$)

The $\overset{\text{PAR}}{\Longrightarrow}$ relation captures meronomy, another inheritance system which can informally be thought of as a "part of" tree over nouns. It is computed by taking, for each pair of WordNet synsets that are related in WordNet by the meronym relation, the cross-product of the pair. For example, `shoe` $\overset{\text{SPC}}{\Longrightarrow}$ `footwear`. The "comprises" relation is simply its inverse, `PAR-`, as in `tree` $\overset{\text{COM}}{\Longrightarrow}$ `cypress`.

"Antonym of" ($\overset{\text{ANT}}{\Longleftrightarrow}$)

The antonym relation uses the antonym relation defined in WordNet for nouns, verbs, adjectives, and adverbs. It is computed by taking, for each pair of WordNet synsets that are related in WordNet by the antonym relation, the cross-product of the pair. For example, `clear` $\overset{\text{SPC}}{\Longleftrightarrow}$ `opaque`.

"Phonetically similar to" ($\overset{\text{SIM}}{\Longleftrightarrow}$) and "Rhymes with" ($\overset{\text{RHY}}{\Longleftrightarrow}$)

To allow users to cross the dimensions of sound and meaning in their queries, two phonetic relations are added to the mix in Lexical FreeNet. These relations, while amusing for shortest path queries, are not expected to contribute to the text processing applications discussed later in this paper. Both relations leverage the phonetic and lexical stress transcriptions in the CMU Pronouncing Dictionary.

The $\overset{\text{SIM}}{\Longleftrightarrow}$ relation is computed by adding every pair of words in the vocabulary that have pronunciations which differ in *edit distance* by at most some number of edits. Edit distance is computed using a dynamic programming algorithm as the minimum number of substitutions, insertions, and deletions (unweighted, and blind to nearness in substitution) to the first word's phonetic sequence required to reach the second word's phonetic sequence. In our current implementation we limit the relation to pairs with edit distance at most 1, *e.g.* `cancel` $\overset{\text{SIM}}{\Longleftrightarrow}$ `candle`.

The $\overset{\text{RHY}}{\Longleftrightarrow}$ relation is computed by adding each pair of words that have pronunciations such that their phonetic suffixes including and following the primary stressed syllables match, *e.g.* `Reno` $\overset{\text{RHY}}{\Longleftrightarrow}$ `Casino`.

"Anagram of" ($\overset{\text{ANA}}{\Longleftrightarrow}$)

The final relation, $\overset{\text{ANA}}{\Longleftrightarrow}$, is almost, but not quite, completely useless, symmetrically linking lexemes that use the same distribution of letters, as in `Geraldine` $\overset{\text{ANA}}{\Longleftrightarrow}$ `realigned`. This is perhaps best described as a "wormhole" in lexical space.

Extensions

A portion of the wealth of WordNet was discarded in Lexical FreeNet—the verb entailment relation, for instance. Adjectives are somewhat slighted by the system, as their WordNet description in terms of bipolar attributes (Gross and Miller, 1990) is largely ignored.

Other possible semantic relations include the more specialized knowledge-engineered links that appear in typically narrow-coverage semantic nets, such as "acts on", "uses", "stronger than", and the like.

Data-driven approaches to relation induction that dig deeper than the collocation extraction of the trigger computation may prove useful and interesting. One approach (Richardson, 1997; Richardson et al., 1993) bootstraps a parser to induce many unconventional semantic relations from dictionary data. A link grammar (Sleator and Temperley, 1991) applied to data can conceivably be used to extract some interesting relations that live at the syntax/semantics interface.

4 Lexical discovery

A World Wide Web interface to Lexical FreeNet, depicted in Figure 3, is available and has become a popular online resource since its release in late January, 1998.[2]. The program allows the user to issue one of the four template queries to the database described in Section 2.3. One of these query templates ("Fanout") requires only a single source token as input, and this has become a popular lookup tool, providing some of the functionality of a thesaurus and rhyming dictionary. The other query functions require source *and* target tokens to be specified. Each token can itself contain spaces in the case of phrasal inputs, which are normalized to the underscore character in processing. The four basic queries allow the user to specify a subset of the ten primitives relations to permit in the output paths by clicking a series of checkboxes. Upon submission, the state of the checkboxes sets the `ANY` relation to be the union of checked relations.

An additional "Spell check" query mode allows the user to find database tokens that have similar (or exact) spelling to a given input token, where similarity is measured by an orthographic edit distance.

Upon submission, the system finds and displays the path or paths resulting from the query with arrow glyphs representing the various relations. Queries typically finish within an acceptable time window of three to ten seconds. The results screen summarizes the query and allows the user to resubmit it with modifications, improving the ease of database "navigation" over having to return to the title screen.

Feedback from the Web site indicates that the system has been used as an aid in writing poetry and lyrics; devising product names; generating puzzles for elementary school language arts classes; writing greeting cards; devising insults and compliments; and, above all, just exploring. Following are selected examples of the system's output in various configurations.

Shortest path queries

The *shortest path* query is the primary vehicle for establishing connections between words and concepts:

- Shortest path queries that allow all lexical relations can be used to aid in generating puns

[2]See `http://www.link.cs.cmu.edu/lexfn/`

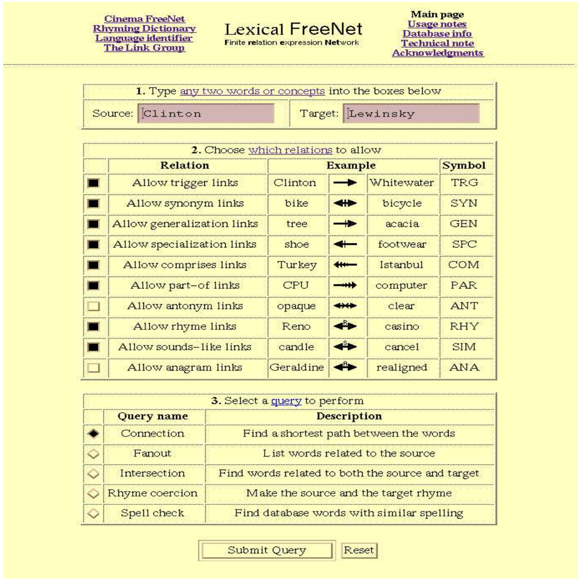

Figure 3: The front page of the Web interface to Lexical FreeNet

and quips involving the two endpoint concepts. For example, below is the shortest path between `Clinton` and `Lewinsky` using all relations:

$$\texttt{CLINTON} \overset{\text{TRG}}{\Longrightarrow} \texttt{HOUSE} \overset{\text{GEN}}{\Longrightarrow} \texttt{CABIN} \overset{\text{TRG}}{\Longrightarrow} \texttt{KACZYNSKI} \overset{\text{RHY}}{\Longleftrightarrow} \texttt{LEWINSKY}$$

- Shortest path queries allowing only the hyponomy relations can connect any two nouns in the WordNet hyponymy tree through their least common ancestor. For example, animals can be connected taxonomically, as in the shortest path between `potto` and `langur` using only the specialization ($\overset{\text{SPC}}{\Longrightarrow}$) and generalization ($\overset{\text{GEN}}{\Longrightarrow}$) and relations:

$$\texttt{POTTO} \overset{\text{SPC}}{\Longrightarrow} \texttt{LEMUR} \overset{\text{SPC}}{\Longrightarrow} \texttt{PRIMATE} \overset{\text{GEN}}{\Longrightarrow} \texttt{MONKEY} \overset{\text{GEN}}{\Longrightarrow} \texttt{OLD\_WORLD\_MONKEY} \overset{\text{GEN}}{\Longrightarrow} \texttt{LANGUR}$$

- Shortest path queries allowing only the meronomy relations can connect many noun pairs. For example, geographical connections can be made between place names to find the largest enclosing region, as in the shortest path between `Saskatoon` and `Winnipeg` using only the comprise ($\overset{\text{COM}}{\Longrightarrow}$) and part-of ($\overset{\text{PAR}}{\Longrightarrow}$) relations:

$$\texttt{SASKATOON} \overset{\text{PAR}}{\Longrightarrow} \texttt{SASKATCHEWAN} \overset{\text{PAR}}{\Longrightarrow} \texttt{CANADA} \overset{\text{COM}}{\Longrightarrow} \texttt{MANITOBA} \overset{\text{COM}}{\Longrightarrow} \texttt{WINNIPEG}$$

- It is counter-intuitive but true that most common words can be connected using only the synonym relation ($\overset{\text{SYN}}{\Longleftrightarrow}$). This demonstrates the high degree of polysemy exhibited by familiar words. Consider the shortest synonym path between `one` and `zero`, a computer scientist's favorite antonym pair. Every successive word pair exhibits a different sense:

$$\texttt{ZERO} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{CIPHER} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{CALCULATE} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{DIRECT} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{LEAD} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{STAR} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{ACE} \overset{\text{SYN}}{\Longleftrightarrow} \texttt{ONE}$$

- Using only the trigger ($\overset{\text{TRG}}{\Longrightarrow}$) relation, one can connect concepts that occur in the domain of the data used to train the trigger pairs, in this case broadcast news:

$$\texttt{SMOKING} \overset{\text{TRG}}{\Longrightarrow} \texttt{CIGARETTES} \overset{\text{TRG}}{\Longrightarrow} \texttt{MACHINES} \overset{\text{TRG}}{\Longrightarrow} \texttt{COMPUTERS}$$

- The trigger relation enriches the WordNet-derived vocabulary of common nouns with topical proper names, as in the shortest paths shown below. Trigger pairs are often expressible in terms of a sequence of one or more WordNet-derived relations. In many cases, however, news-based triggers defy any fixed set of hand-coded lexical relations.

$$\texttt{TITANIC} \overset{\text{TRG}}{\Longrightarrow} \texttt{SANK} \overset{\text{TRG}}{\Longrightarrow} \texttt{SHIP} \overset{\text{TRG}}{\Longrightarrow} \texttt{VALDEZ} \overset{\text{TRG}}{\Longrightarrow} \texttt{COFFEE}$$

$$\texttt{NADER} \overset{\text{TRG}}{\Longrightarrow} \texttt{REGULATIONS} \overset{\text{TRG}}{\Longrightarrow} \texttt{ENVIRONMENTAL} \overset{\text{TRG}}{\Longrightarrow} \texttt{GORE}$$

$$\texttt{FALWELL} \overset{\text{TRG}}{\Longrightarrow} \texttt{CHRISTIAN} \overset{\text{TRG}}{\Longrightarrow} \texttt{CONSERVATIVE} \overset{\text{TRG}}{\Longrightarrow} \texttt{GINGRICH}$$

- But when the WordNet-derived semantic relations are permitted in addition to the trigger relation, shortest paths become shorter, overcoming the inherent limitations of the data-derived triggers. In the case below, the pair `(relativity, physics)` did not occur sufficiently often in training data for the pair to make the grade as a trigger.

$$\texttt{EINSTEIN} \overset{\text{TRG}}{\Longrightarrow} \texttt{RELATIVITY} \overset{\text{PAR}}{\Longrightarrow} \texttt{PHYSICS} \overset{\text{TRG}}{\Longrightarrow} \texttt{VELOCITY} \overset{\text{GEN}}{\Longrightarrow} \texttt{SPEED\_OF\_LIGHT}$$

- For amusement, the phonetic relations, rhymes-with ($\overset{\text{RHY}}{\Longleftrightarrow}$) and sounds-like ($\overset{\text{SIM}}{\Longleftrightarrow}$), can be used alone to produce "word ladders" of sequentially similar words, as in the example below. In combination with the semantic relations, the phonetic relations can aid in creating rhymed poetry and puns.

$$\texttt{KNIFE} \overset{\text{SIM}}{\Longleftrightarrow} \texttt{NINE} \overset{\text{SIM}}{\Longleftrightarrow} \texttt{SPINE} \overset{\text{SIM}}{\Longleftrightarrow} \texttt{SPOON}$$

Intersection queries

Intersection queries can be used in Lexical FreeNet to find the set of concepts and words that two inputs both directly relate to in some way. We use the notation $(w_1 \overset{r_1}{\Longrightarrow}, w_2 \overset{r_2}{\Longrightarrow}) w_3$ to mean that "w_1 is related to w_3 by relation r_1, and w_2 is related to w_3 by relation r_2.

- For concrete nouns, the results are often expected but sometimes subtle:

$$(\texttt{FROG} \overset{\text{TRG}}{\Longrightarrow}, \texttt{TURTLE} \overset{\text{TRG}}{\Longrightarrow})\ \texttt{POND}$$

$$(\texttt{ORANGE} \xRightarrow{\text{TRG}}, \texttt{APPLE} \xRightarrow{\text{TRG}})\ \texttt{JUICE}$$

$$(\texttt{BANANA} \xRightarrow{\text{TRG}}, \texttt{ONION} \xRightarrow{\text{TRG}})\ \texttt{PEEL}$$

$$(\texttt{BOOK} \xRightarrow{\text{TRG}}, \texttt{TELEVISION} \xRightarrow{\text{TRG}})\ \texttt{STORY}$$

$$(\texttt{TREE} \xRightarrow{\text{COM}}, \texttt{TOOTH} \xRightarrow{\text{COM}})\ \texttt{CROWN}$$

- Triggers can be a useful tool for discovering what two names in the news have in common, or two names in history:

$$(\texttt{STARR} \xRightarrow{\text{TRG}}, \texttt{MCDOUGAL} \xRightarrow{\text{TRG}})\ \texttt{WHITEWATER}$$

$$(\texttt{CHURCHILL} \xRightarrow{\text{TRG}}, \texttt{STALIN} \xRightarrow{\text{TRG}})$$
$$\texttt{HITLER},\ \texttt{ROOSEVELT},\ \texttt{TRUMAN},\ \texttt{POTSDAM}$$

- In some cases, identification questions can be formulated as intersection queries. For example, "What's the name of that congresswoman from Colorado I'm always hearing about?" can be asked as an intersection query with arguments (`congresswoman`, `Colorado`). "What's the capital of the state of Nebraska?" can be asked as an intersection query with arguments (`Nebraska`, `state_capital`):

$$(\texttt{COLORADO} \xRightarrow{\text{TRG}}, \texttt{CONGRESSWOMAN} \xRightarrow{\text{TRG}})$$
$$\texttt{SCHROEDER}$$

$$(\texttt{NEBRASKA} \xRightarrow{\text{COM}}, \texttt{STATE\_CAPITAL} \xRightarrow{\text{GEN}})\ \texttt{LINCOLN}$$

Rhyme coercion queries

The phonetic relations in Lexical FreeNet are particularly useful for finding rhyming words with certain target meanings. The coercion function on the Web interface is hardcoded such that the relation `re1` (see Section 2.3) is simply the union of all semantic relations, and `re2` is the union of all phonetic relations. Thus, given two endpoint words (w_1, w_2), the system tries to find words (w_1', w_2'), with respectively related meanings, that rhyme or sound alike. For example, if you wanted to write a poem about petting a lion, you might do a coercion query with the words `touch` and `lion`. Amongst a few others, you'll get back the suggestions (`RUB`, `CUB`), since $\texttt{TOUCH} \xRightarrow{\text{GEN}} \texttt{RUB}$ and $\texttt{LION} \xRightarrow{\text{TRG}} \texttt{CUB}$; and (`PAT`, `CAT`), since $\texttt{TOUCH} \xRightarrow{\text{GEN}} \texttt{PAT}$ and $\texttt{LION} \xRightarrow{\text{TRG}} \texttt{CAT}$. Most rhyme coercion queries to the online system have produced at least one result in this manner.

5 Conclusion

We have introduced a database system called FreeNet that facilitates the description and exploration finite binary relations, and also an instance of the system called Lexical FreeNet that supports a range of lexical semantic applications. The program has proven itself to be a useful and entertaining resource for lexical discovery by Internet users. We hope to employ the system as a common algorithmic core for three text processing applications as well—segmentation, summarization, and information extraction.

Acknowledgments

The author thanks Michael Turniansky for early feedback on this work; Adam Berger for developing the Trigger Toolkit; Carl Burch for help with the phonetic and orthographic edit distance functions; Bob Harper and John Lafferty for useful discussions; and the many users of the World Wide Web interface who have provided entertaining feedback on the system.

References

D. Beeferman, A. Berger, and J. Lafferty. 1997. A model of lexical attraction and repulsion. In *Proceedings of the ACL*, Madrid, Spain.

A. Blum, G. Konjevod, R. Ravi, and S. Vempala. to appear. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. In *Proc. of the 30th ACM Symposium on the Theory of Computing*, pages 95–100.

CMU. 1997. Carnegie Mellon University Pronouncing Dictionary v0.6d. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

C. Fellbaum. 1990. English verbs as a semantic net. *International Journal of Lexicography*, 3,4:278–301.

D. Gross and K. Miller. 1990. Adjectives in WordNet. *International Journal of Lexicography*, 3,4:265–277.

LDC. 1997. DARPA Continuous Speech Recognition Corpus-IV: Radio Broadcast News (CSRIV Hub-4). http://morph.ldc.upenn.edu/.

G. Miller. 1990. Nouns in WordNet: a lexical inheritance system. *International Journal of Lexicography*, 3,4:245–264.

C. Papadimitriou. 1976. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270.

S. Richardson, L. Vanderwende, and W. Dolan. 1993. Combining dictionary-based and example-based methods for natural language analysis. In *Proc. Fifth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 69–79.

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

R. Rosenfeld. 1994. *Adaptive Statistical Language Modeling: a Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University, April.

D. Sleator and D. Temperley. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University.