l e x a n a \ n e t

February 2006

In Other Words: Expanding Queries with WordNet

By Drew Hamre

email

WordNet® is a digital dictionary that preserves and navigates the semantic relationships among words. Whereas a typical dictionary is organized alphabetically - and words with similar meanings are scattered throughout - WordNet turns this on its head.

WordNet[i] entries are organized by conceptual relatedness. Software can search these hierarchical networks to retrieve synonyms, calculate linguistic similarity, and so on. WordNet contains more than 207,000 word senses, and it has become a linchpin for natural language research referenced by a million-plus web pages.

WordNet is freely downloadable and may be used in commercial. However, WordNet displays its full power more dramatically in search engines and text analysis software. These latter systems disambiguate word usage, index and summarize documents, analyze discourse and assess essays. WordNet can be used to support information extraction systems, to help correct textual material, and to build custom ontologies and to drive linguistic research.

In this paper we'll provide an overview of WordNet – its acquisition, licensing, design, its varying implementations and its functionality. We'll present two software examples that retrieve WordNet 'coordinate terms' (helpful in expanding the semantic scope of a retrieval), and we'll review the role of query expansion in search applications.

Acquiring and installing WordNet

The latest version of WordNet (v2.1) was released in March 2005 for Linux/Unix and for Windows (the latter is our focus here). The 18-megabyte install package contains WordNet*'s* databases, API, and a desktop application that navigates this information.

WordNet's licensing is non-restrictive: "WordNet is unencumbered, and may be used in commercial applications in accordance with (its) license agreement". The agreement requires inclusion of a copyright statement wherever WordNet is used, and specifies that all rights to WordNet*'s* software, database and documentation remain with Princeton University, where WordNet was developed at the Cognitive Science lab.

For a tangible introduction to WordNet, users can experiment with the bundled desktop application[1]. Below, we've used this application to search for the term 'gladiola', and then requested that WordNet climb the semantic hierarchy where 'gladiola is a kind of…'.

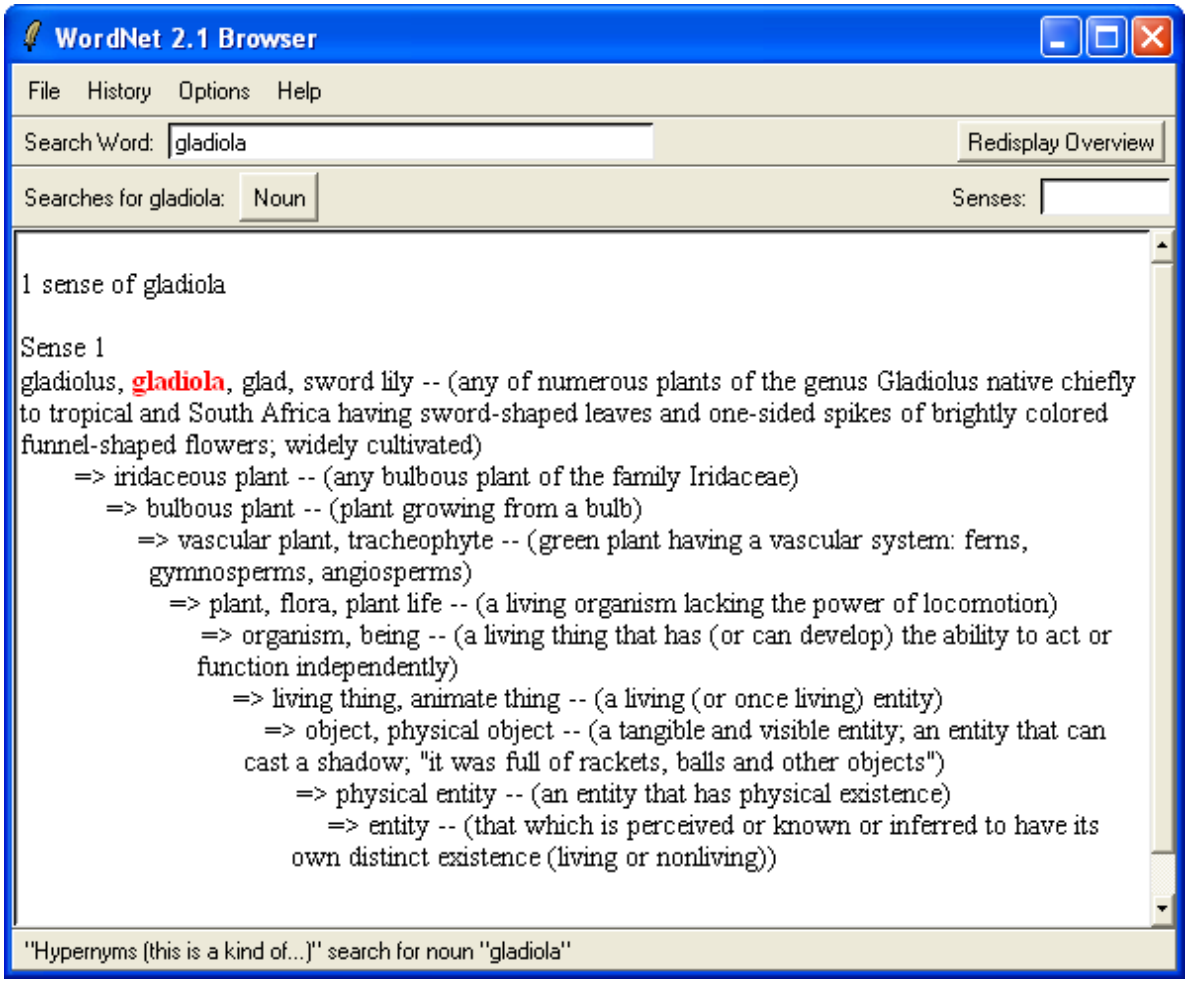

Figure 1: WordNet's desktop application

The words in the above hierarchy are linked through a single type of linguistic relation: *hypernyms*[2] ('is a kind of'). However, WordNet also captures more than 20 additional relationship types, a trait that separates it from a simple thesaurus. Embedded in WordNet*'s* design is the strong assumption that these relationships vary by parts of

---

[1] You may prefer to experiment with *WordNet's* web interfaces (both basic and advanced), as these require no local install. Another sophisticated web UI is the dynamic Java-powered display of *VisualThesaurus*.

[2] A certain amount of jargon is unavoidable when using *WordNet*, especially for those terms that describe linguistic relationships. For nouns, relationship types include *hypernyms* (is a kind of), *hyponyms* (is a part of), *coordinates* (terms sharing the same hypernym), and *meronyms* (parts of *term).* Different parts of speech are associated with different relationships. For example, adjectives include the relation 'is a value of …' (e.g., white is a value of the relative darkness or lightness of a color).

In *WordNet*, particular senses of words are grouped into *synsets* (words that are interchangeable in some context). The *synset* is central to *WordNet's* design, and in fact *WordNet* has been described simply as "a *synset* network connected by semantic relations" .

speech. That is, the relationships that associate nouns with other nouns are different from the relationships that link, say, adjectives with other adjectives.

WordNet*'s* presumption about the fundamental differences among parts of speech extends to the physical design of its database (below). By default, WordNet*'s* installer builds the following directory tree beneath \Program Files\WordNet\ to include databases, executables, API source (written in generic *C*) and documentation.

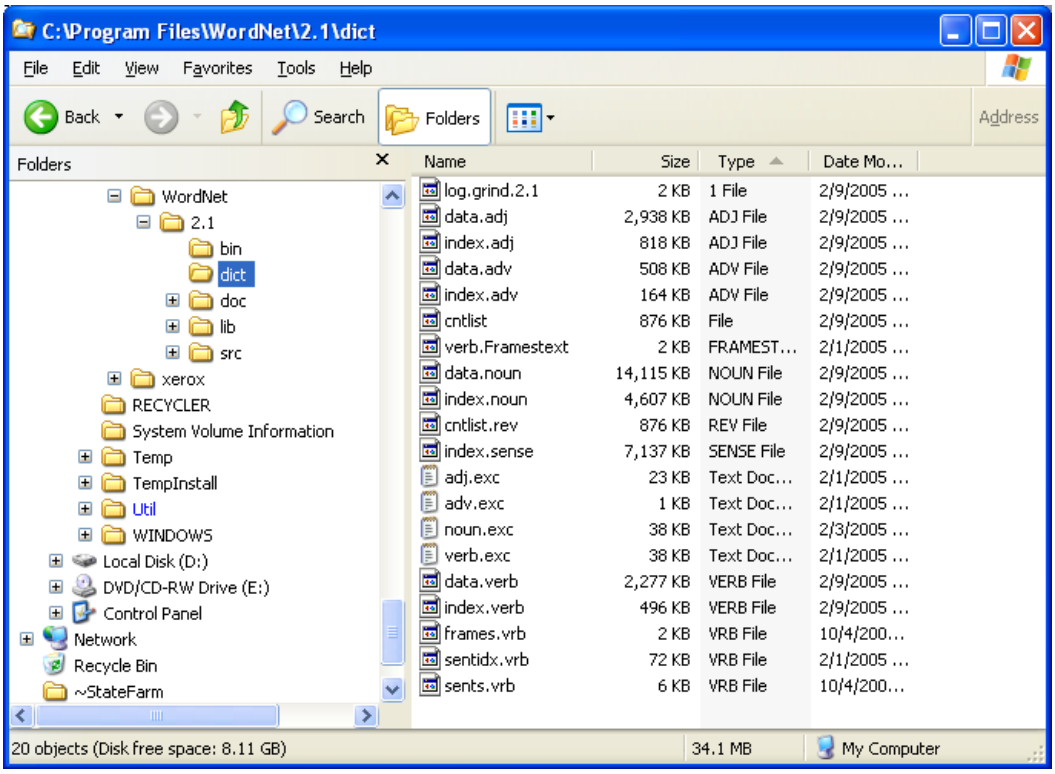

Figure 2: WordNet 2.1 installation, showing database sub-directory

WordNet*'s* data are stored in files that use a proprietary indexed format. Two design details are noteworthy: (1) different Parts of Speech (POS) are stored separately, and (2) each POS database is built from two component files -- *data* and *index* (discussed below). Other files provide navigation support.

WordNet entries may be composed of multiple words (e.g., 'turning point' or 'brand name', a form WordNet calls a *collocation*). Note that WordNet contains only nouns, verbs, adverbs and adjectives. WordNet does not include function words (*the*, *an*, and so on), pronouns, domain-specific language (e.g., *SQL*), or common product names.

Index files for WordNet are alphabetized lists of all words for that POS. Each index entry is followed by a list of byte offsets into the corresponding *data* file, one for each *synset* containing the word. Developers who reformat these files typically identify and list each item in the synset, insert each member of the synset list as a separate entry in the database, and for each member of the synset insert each of the other synset members in a related field.

Because of its complexity, WordNet's software (which includes low-level I/O routines) must be modified and re-released with each new iteration of the data files. WordNet's complexity -- and the desire to extend its coverage to include specialized terminology

and common brand names -- has spurred interest in re-architecting the system for other more easily modified environments, including relational.

Relational implementations of the WordNet data stores

Several independent projects have converted WordNet's data stores into relational formats. Their common strategy is to parse WordNet's native files and emit a script of corresponding SQL *INSERT* statements to load relational tables. Such efforts include a) the WORDNET2SQL project, b) a MySQL conversion, c) a SQL Server conversion, and d) WNSQLBUILDER.

Database designs differ for these relational projects (and some are quite complex), but the WORDNET2SQL schema is among the most intelligible.

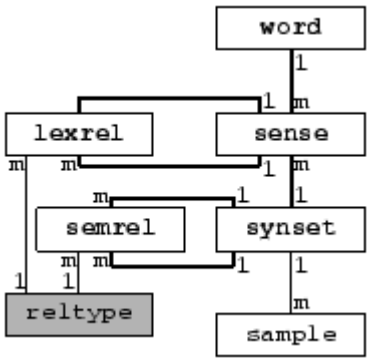

Figure 3: A relational implementation of WordNet (WORDNET2SQL project)

The core entities in the WORDNET2SQL implementation are:

- *Word* – ASCII-encoded written representation of any sequence of characters that we associate a meaning with.
- *Sense* - Association between a word and the semantic information it carries (expressed here in the *synset* table). Words may have multiple senses.
- *Synset* – WordNet's basic entity of semantic information, possibly containing several word senses that can be considered synonymous within a particular context.
- *Lexrel* – Expresses lexical relations that hold between usages of certain words. Relation types include *hyponym, hypernym, antonym*, and so on.
- *Semrel* – Expresses semantic relations that hold between *synsets*, including relation types like *hyponym, hypernym, antonym*, and so on.

We'll use a similar schema below (in our SQL Server-based alternative) to explore a common use of WordNet in information retrieval systems: *query expansion*.

An exemplary use of WordNet: Query expansion

Search engines and other document retrieval systems balance two conflicting design goals: *recall* (returning all relevant pages) and *precision* (returning only relevant pages). The problem is complicated by the queries themselves, which tend to be brief and variable (as less than 25% of users use identical terms to describe a concept).

Query expansion addresses this problem by programmatically embellishing the original query with related search terms. For example, if a user requested all documents that matched "gladiola", the query might be expanded to "gladiola, gladiolus, glad, sword lily". If coordinate[3] terms were used, the query would be extended further with "iris, flag, fleur-de-lis, blackberry-lily, leopard lily, crocus, freesia and corn lily".

Different expansion techniques (knowledge-based, statistical, linguistic) have been studied, and results are mixed at best. It's generally held that indiscriminate query expansion can lead to loss of precision or even hurt recall; the major causes of error are word sense ambiguity and lack of domain-specific precision in WordNet's general purpose vocabulary. Despite the recognized pitfalls, query expansion is widely used (even by Google, as we'll see below). WordNet-based query expansion plays at least a supporting role in many of these systems.

Techniques for *WordNet*-based query expansion: Two examples

Following are two *WordNet*-based query expansion examples, illustrating how to navigate both WordNet's native data store and a relational variation. In each, the strategy will be to look up search words in WordNet, and then retrieve related terms (synonyms and coordinates). A complete solution would then fold these additional terms into the query, with weights deprecated relative to the original search terms.

Both samples are implemented using Microsoft technologies (Visual Basic .NET and SQL Server) – a decision that often proves a hindrance rather than a help, as these tools aren't widely supported in natural language research[ii].

Searching native WordNet data stores from VB.NET

WordNet's API isn't directly callable from either COM or .NET, so third-party tools are used to bridge these particular interfaces. Therefore, our sample software will require that we load (in addition to WordNet 2.1) a just-released third-party library[4] of .NET objects (WordNet.NET) that access the native WordNet 2.1 data stores. The sample solution includes this third-party WordNet.NET project, plus our custom VB.NET project. Code snippets from the latter project are given below (and note that a different, unexpurgated example is here).

The following code highlights techniques including: a) instantiation of the WordNet access classes, b) the initial term lookup (.*OverviewFor* method), c) the determination of POS (and note a full implementation would need to allow for multiple senses in multiple POSs – e.g., 'train/verb-practice' vs. 'train/noun-conveyance), d) the retrieval of appropriate relation types for the relevant POS (.*Opt* collection), and e) the search for terms linked by a selected relationship type (.*Search* method).

```
Imports WordNetClasses.WN   ' Import interface library
' WordNet 2.1 datafile path
Private sDictPath As String = "C:\Program Files\WordNet\2.1\dict\"

Private Sub cmdLookup_Click()
```

---

[3] *Coordinates* have the same hypernym; that is, they share parent nodes in the lexical hierarchy.

[4] The .NET conversion was completed by Troy Simpson, based on initial work by Malcolm Crowe. The conversion targets .NET Framework 1.1 (with a 2.0 conversion underway). Libraries are written in C#, and the download package includes all source files. The libraries may be redistributed and/or modified under terms of the GNU Lesser General Public License. The current developer has stated his intent to redesign the class libraries as part of the 2.0 conversion.

```
    Private oWNC As WordNetClasses.WN = New WordNetClasses.WN(sDictPath)          '(a)
    ' - Locate definition of word (noun  presumed)
    oWNC.OverviewFor(sInput, "noun", bIsNoun, oWNSSNoun, arArrayList)             '(b)
    ' - arArrayList carries returned definition
    ShowResults(arArrayList)
    ' - Load all search relations for nouns
    Dim opt As Wnlib.Opt
    Dim posNoun As Wnlib.PartOfSpeech = Wnlib.PartOfSpeech.of("noun")
    For iOpt = 0 To Wnlib.Opt.Count - 1
        opt = opt.at(iOpt)
        If (opt.at(iOpt).pos Is posNoun) Then
            'Found new relationship type for nouns                              '(c)
        End If
    Next iOpt
End Sub

Private Sub Search(sSearchLabel As String)
    Dim opt As Wnlib.Opt
    For iOpt = 0 To Wnlib.Opt.Count - 1
        opt = opt.at(iOpt)
        If (opt.at(iOpt).label = sSearchLabel) Then                            '(d)
            Dim oSearch As Wnlib.Search = New Wnlib.Search(txtInput.Text, True, _  '(e)
              opt.pos, opt.sch, 0)
            ' oSearch.buf returns search results (must accumulate)
            sOutput = sOutput & oSearch.buf
        End If
    Next iOpt
End Sub
```

If we add a simple UI (wherein relationship types are selected via a drop-down), we can request coordinate terms (hyponyms of the search term's hypernyms), and thus retrieve the raw material with which to expand a user query. Shown below are the coordinates for 'gladiola'.

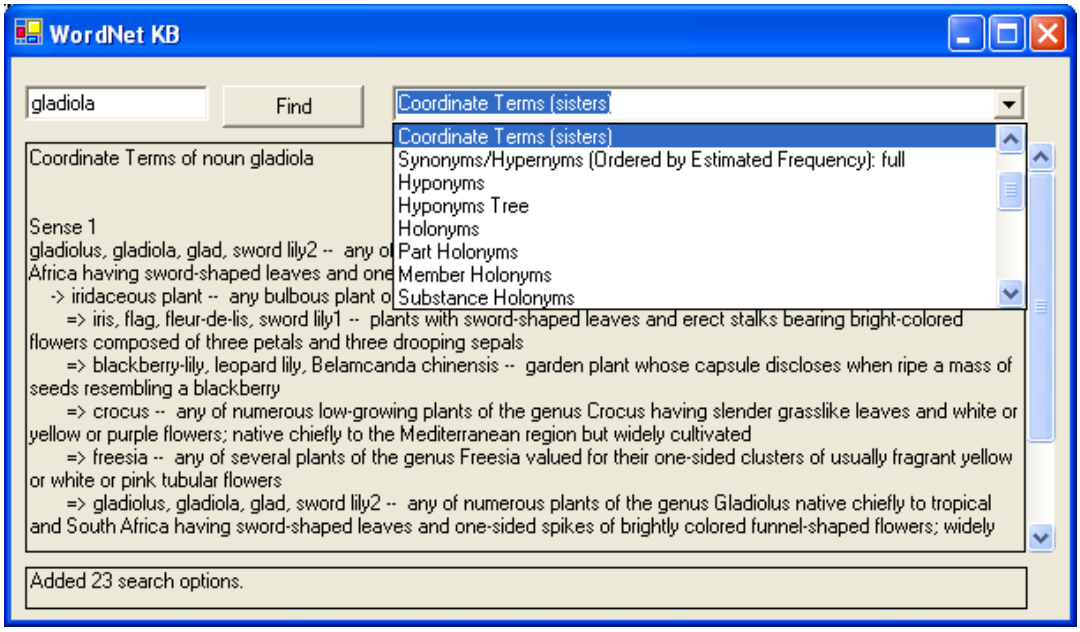

Figure 4: VB.NET retrieving coordinate terms from WordNet

We should flag two issues. First, the hierarchic formatting characters (line breaks, '=>', '->') are inserted *by the retrieval library itself*, not the UI layer. (Perhaps this behavior will be changed as part of the library's V2.0 re-design.) Second, we've chosen a sample search term ('gladiola') that has only one sense, thus hiding the complexity that would result for search terms with multiple senses.

Searching SQL Server-based relational WordNet

Princeton's WordNet website lists several ports made by unaffiliated developers to relational systems (chiefly to MySQL). Although a port to Microsoft's SQL Server is not among those listed, several have been released on the web. The schema for the implementation[5] used below is similar to the WordNet2SQL schema (see Figure 3, above).

The following T-SQL code retrieves coordinate terms from this WordNet/SQL Server database (unexpurgated sample).

```
DECLARE @synsetno int
DECLARE @synsetno_hyp int
-- Get the synset for gladiola (word->sense->synset)
SET @synsetno = (
       SELECT TOP 1 sy.synsetno FROM synset sy, sense se, word w
       WHERE w.lemma = 'gladiola' AND
       w.wordno = se.wordno AND
       se.synsetno = sy.synsetno )
PRINT @synsetno
-- Get the hypernym of this synset
SET @synsetno hyp = (
       SELECT TOP 1 sr.synset2 FROM semrel sr
       WHERE sr.reltype = 2 AND
       (synset1 = @synsetno))
PRINT @synsetno hyp
-- Get all words for all senses in hyponym synsets for the hypernym
SELECT w.* FROM word w
WHERE w.wordno IN
       (SELECT se.wordno from sense se
        WHERE se.synsetno IN
              (SELECT sr.synset2 FROM semrel sr
               WHERE sr.reltype = 3 AND
               sr.synset1 = @synsetno_hyp)
       )
```

In the above, our first query retrieves the synset ID associated with the search term (and note the 'TOP 1' restriction is merely a kludge to enforce our single-sense simplification). The second query retrieves the hypernym's synset. The third query (with nested SELECTs) descends along this chain to retrieve any words associated with the hypernym's hyponym synsets. These coordinate terms would then be used to embellish and expand a user query, as discussed earlier.

---

[5] The SQL Server implementation of WordNet is a work in progress. To load this information into SQL Server 2000: a) define an empty database called 'WordNet20' with an initial space allocation of 50-megabytes; b) set the recovery model for the database to 'Bulk-Logged'; c) edit the table creation script so that field 'description' in table 'reltype' is length 80, not 30; d) execute the table creation script against the empty WordNet20 database; and then e) run the remaining load scripts in the following order (after inserting 'SET NOCOUNT ON' as the first line in each script): *word, synset, sample, sense, lexrel, semrel, adjmod, frame, lexname, reltype, and verbframe*. The indexing script contains errors, but need not be executed to demonstrate the functionality shown here.

A commercial role for query expansion?

As noted above, simple forms of query expansion are generally held to aid recall (though not always) and to hurt precision; key research has been carried out by Ellen Voorhees. The results are laid to difficulties in choosing appropriate senses for the search term (as disambiguation remains an open issue in language research). In fact, Voorhees and others have concluded that a) for short queries when disambiguation is perfect, then WordNet expansion does improve performance. However, if disambiguation isn't perfect (and it rarely is), then expansion can diminish recall and precision.

Despite the hurdles, WordNet-based query expansion remains a vital topic among researchers. Successful techniques include systems that use WordNet expansion only for disambiguated terms, or for libraries where the topic domain is restricted. For example, retrieval scores for a geographic information system were improved by pre-indexing documents with WordNet expansion terms.

Explicit query expansion in Google and MySQL

Query expansion is commonly used within the search industry. For example, Google allows users to request query expansion using the tilde ('~') operator. Google's search engine then includes synonym suggestions for words preceded by this tilde (for example, '~inexpensive' will also match 'cheap', 'affordable', 'low cost', and so on). Google has applied for a patent covering certain types of automatic query expansion.

Among prominent DBMSs, MySQL now supports query expansion retrieval syntax:

```
mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('database' WITH QUERY EXPANSION);
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  3 | Optimizing MySQL  | In this tutorial we will show ...        |
+----+-------------------+------------------------------------------+
3 rows in set (0.00 sec)
```

Other WordNet opportunities

A recent overview of natural language processing lists many open research problems. Some problems are 'small' (information extraction (database filling), question answering, thesaurus/key phrase generation) and some are 'large' (document summarization, document categorization, clustering and topic detection).

WordNet is positioned to play a role in all of these efforts, in addition to supporting more prosaic functionality in everyday business applications. With sophisticated text processing finally capturing financial attention (e.g., the $1.85-billion acquisition of

AskJeeves), and with growing recognition of the sheer volume of corporate textual data and its value, WordNet is a welcome addition to a developer's toolbox.

Summary

The freely-available WordNet database provides an English lexicon (glossary) with more than 207,000 word senses, and also captures the semantic relationships among these words. Software can navigate these conceptual networks to retrieve synonyms, calculate linguistic similarity, categorize text, and so on.

This paper discusses WordNet's installation, design, functionality, and its potential application to common business problems. It includes WordNet samples to expand the semantic scope of a query by retrieving 'coordinate terms' for a search string.

Resources

The *WordNet* homepage at Princeton University
http://wordnet.princeton.edu/

Introduction to WordNet: An On-line Lexical Database by George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller
http://wordnet.princeton.edu/5papers.pdf

WordNet Bibliography (now maintained by Andras Csomai)
http://mira.csci.unt.edu/~wordnet/

Representing the Meaning of Documents by Jimmy Lin
http://www.umiacs.umd.edu/~jimmylin/LBSC796-INFM718R-2005-Spring/lecture5.pdf

Acknowledgement

Thanks to reviewers for their time and suggestions; any inaccuracies remain mine.

Drew Hamre is a principal of lexana\net and lives in Golden Valley, Minnesota.

---

[i]

Cognitive psychologist George A. Miller has directed WordNet's development since its inception. Dr. Miller, age 85, remains the McDonnell Distinguished University Professor of Psychology, Emeritus, at Princeton University. Fifty-years ago, Dr. Miller wrote a classic paper on human memory (The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information). This paper was one of the first applications of information theory to cognitive processes (and almost certainly the first appearance of terms like *bits* and *channel capacity* in *The Psychological Review*). Miller's paper – and convergent linguistic research by Noam Chomsky – undercut Skinnerian behaviorism.

[ii] The software samples shown here are Microsoft-centric (VB.NET and SQL Server). However, among WordNet developers (and more generally, among University research communities) Microsoft development is increasingly anomalous. The code emerging from research universities today is overwhelmingly written for Linux, Java, and various other open platforms. For example, Open Source text processing software includes (in addition to WordNet) tools like the Lucene search engine, spell checkers, text mining algorithms, and named-entity recognition