Building and Using a Lexical Knowledge Base of Near-Synonym Differences

Diana Inkpen[*]
University of Ottawa

Graeme Hirst[†]
University of Toronto

Choosing the wrong word in a machine translation or natural language generation system can convey unwanted connotations, implications, or attitudes. The choice between near-synonyms such as ***error****,* ***mistake****,* ***slip****, and* ***blunder****—words that share the same core meaning, but differ in their nuances—can be made only if knowledge about their differences is available.*

We present a method to automatically acquire a new type of lexical resource: a knowledge base of near-synonym differences. We develop an unsupervised decision-list algorithm that learns extraction patterns from a special dictionary of synonym differences. The patterns are then used to extract knowledge from the text of the dictionary.

The initial knowledge base is later enriched with information from other machine-readable dictionaries. Information about the collocational behavior of the near-synonyms is acquired from free text. The knowledge base is used by Xenon, a natural language generation system that shows how the new lexical resource can be used to choose the best near-synonym in specific situations.

1. Near-Synonyms

Near-synonyms are words that are almost synonyms, but not quite. They are not fully intersubstitutable, but vary in their shades of denotation or connotation, or in the components of meaning they emphasize; they may also vary in grammatical or collocational constraints. For example, the word *foe* emphasizes active warfare more than *enemy* does (Gove 1984); the distinction between *forest* and *woods* is a complex combination of size, proximity to civilization, and wildness (as determined by the type of animals and plants therein) (Room 1981); among the differences between *task* and *job* is their collocational behavior with the word *daunting*: *daunting task* is a better collocation than *daunting job*. More examples are given in Table 1 (Hirst 1995).

There are very few absolute synonyms, if they exist at all. So-called dictionaries of synonyms actually contain near-synonyms. This is made clear by dictionaries such as *Webster's New Dictionary of Synonyms* (Gove 1984) and *Choose the Right Word* (hereafter CTRW) (Hayakawa 1994), which list clusters of similar words and explicate the differences between the words in each cluster. An excerpt from CTRW is presented in Figure 1. These dictionaries are in effect dictionaries of near-synonym discrimination.

[*] School of Information Technology and Engineering, Ottawa, ON, Canada, K1N 6N5; diana@site.uottawa.ca.

[†] Department of Computer Science, Toronto, ON, Canada, M5S 3G4; gh@cs.toronto.edu.

Submission received: 5 October 2004; revised submission received: 15 June 2005; accepted for publication: 4 November 2005.

© 2006 Association for Computational Linguistics

Table 1
Examples of near-synonym variations.

Type of variation	Example
Stylistic, formality	*pissed* : *drunk* : *inebriated*
Stylistic, force	*ruin* : *annihilate*
Expressed attitude	*skinny* : *thin* : *slim*
Emotive	*daddy* : *dad* : *father*
Continuousness	*seep* : *drip*
Emphasis on different aspects of meaning	*enemy* : *foe*
Fuzzy boundary	*woods* : *forest*
Collocational	*task* : *job* (in the context of *daunting*)

Writers often turn to such resources when confronted with a choice between near-synonyms, because choosing the wrong word can be imprecise or awkward or convey unwanted implications. These dictionaries are made for human use, and they are available only on paper, not in electronic format.

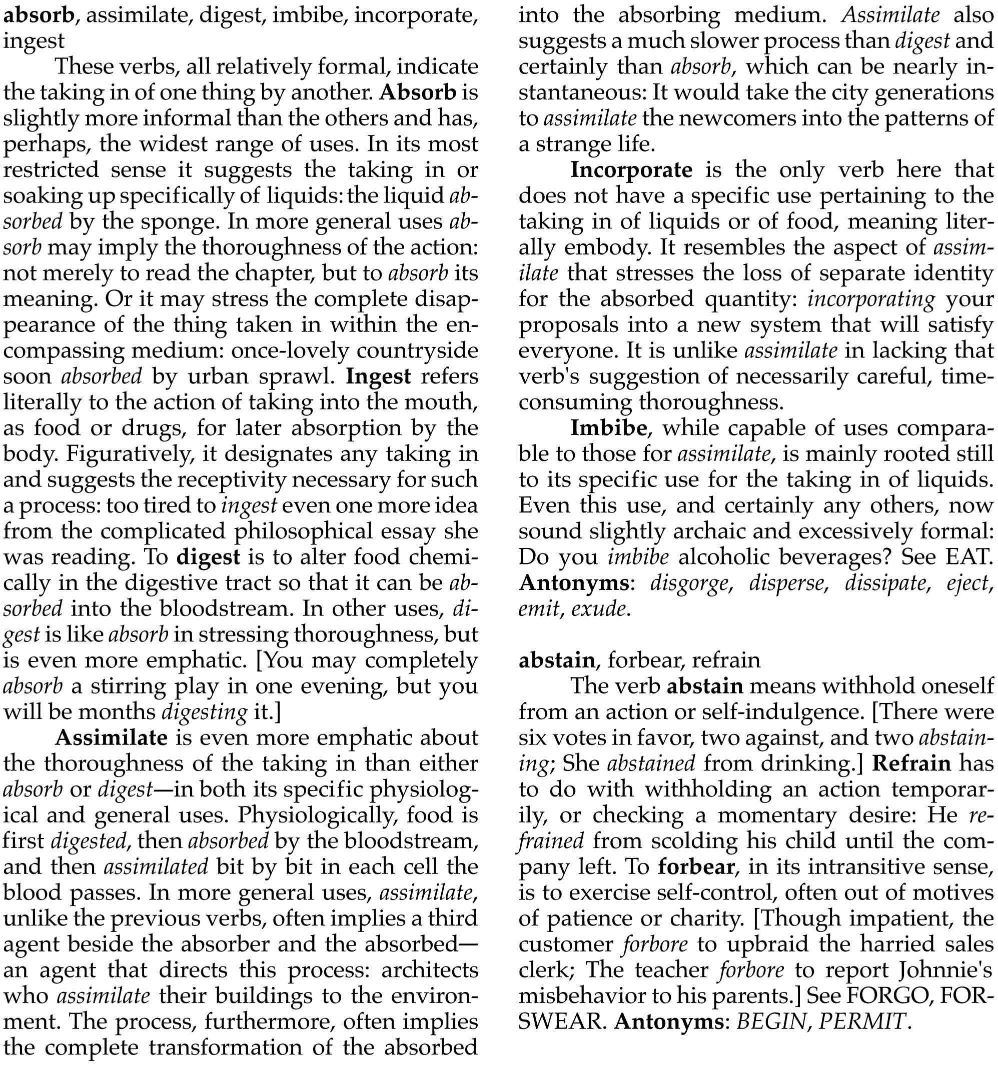
Understanding the differences between near-synonyms is important for fine-grained distinctions in machine translation. For example, when translating the French word *erreur* to English, one of the near-synonyms *mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism* could be chosen, depending on the context and on the nuances that need to be conveyed. More generally, knowledge of near-synonyms is vital in natural language generation systems that take a nonlinguistic input (semantic representation) and generate text. When more than one word can be used, the choice should be based on some explicit preferences. Another application is an intelligent thesaurus, which would assist writers not only with lists of possible synonyms but also with the nuances they carry (Edmonds 1999).

1.1 Distinctions among Near-Synonyms

Near-synonyms can vary in many ways. DiMarco, Hirst, and Stede (1993) analyzed the types of differences adduced in dictionaries of near-synonym discrimination. They found that there was no principled limitation on the types, but a small number of types occurred frequently. A detailed analysis of the types of variation is given by Edmonds (1999). Some of the most relevant types of distinctions, with examples from CTRW, are presented below.

Denotational distinctions Near-synonyms can differ in the **frequency** with which they express a component of their meaning (e.g., *Occasionally, **invasion** suggests a large-scale but unplanned **incursion***), in the **latency** (or **indirectness**) of the expression of the component (e.g., ***Test** strongly implies an actual application of these means*), and in fine-grained **variations** of the idea itself (e.g., ***Paternalistic** may suggest either benevolent rule or a style of government determined to keep the governed helpless and dependent*). The frequency is signaled in the explanations in CTRW by words such as *always*, *usually*, *sometimes*, *seldom*, *never*. The latency is signaled by many words, including the obvious words *suggests*, *denotes*, *implies*, and *connotes*. The strength of a distinction is signaled by words such as *strongly* and *weakly*.

Attitudinal distinctions Near-synonyms can convey different **attitudes** of the speaker toward an entity in the situation. Attitudes can be **pejorative**, **neutral**, or **favorable**. Examples of sentences in CTRW expressing attitudes, in addition to denotational distinctions, are these: ***Blurb** is also used pejoratively to denote the extravagant and insincere*



absorb, assimilate, digest, imbibe, incorporate, ingest

These verbs, all relatively formal, indicate the taking in of one thing by another. **Absorb** is slightly more informal than the others and has, perhaps, the widest range of uses. In its most restricted sense it suggests the taking in or soaking up specifically of liquids: the liquid *absorbed* by the sponge. In more general uses *absorb* may imply the thoroughness of the action: not merely to read the chapter, but to *absorb* its meaning. Or it may stress the complete disappearance of the thing taken in within the encompassing medium: once-lovely countryside soon *absorbed* by urban sprawl. **Ingest** refers literally to the action of taking into the mouth, as food or drugs, for later absorption by the body. Figuratively, it designates any taking in and suggests the receptivity necessary for such a process: too tired to *ingest* even one more idea from the complicated philosophical essay she was reading. To **digest** is to alter food chemically in the digestive tract so that it can be *absorbed* into the bloodstream. In other uses, *digest* is like *absorb* in stressing thoroughness, but is even more emphatic. [You may completely *absorb* a stirring play in one evening, but you will be months *digesting* it.]

Assimilate is even more emphatic about the thoroughness of the taking in than either *absorb* or *digest*—in both its specific physiological and general uses. Physiologically, food is first *digested*, then *absorbed* by the bloodstream, and then *assimilated* bit by bit in each cell the blood passes. In more general uses, *assimilate*, unlike the previous verbs, often implies a third agent beside the absorber and the absorbed—an agent that directs this process: architects who *assimilate* their buildings to the environment. The process, furthermore, often implies the complete transformation of the absorbed into the absorbing medium. *Assimilate* also suggests a much slower process than *digest* and certainly than *absorb*, which can be nearly instantaneous: It would take the city generations to *assimilate* the newcomers into the patterns of a strange life.

Incorporate is the only verb here that does not have a specific use pertaining to the taking in of liquids or of food, meaning literally embody. It resembles the aspect of *assimilate* that stresses the loss of separate identity for the absorbed quantity: *incorporating* your proposals into a new system that will satisfy everyone. It is unlike *assimilate* in lacking that verb's suggestion of necessarily careful, time-consuming thoroughness.

Imbibe, while capable of uses comparable to those for *assimilate*, is mainly rooted still to its specific use for the taking in of liquids. Even this use, and certainly any others, now sound slightly archaic and excessively formal: Do you *imbibe* alcoholic beverages? See EAT. **Antonyms**: *disgorge, disperse, dissipate, eject, emit, exude*.

abstain, forbear, refrain

The verb **abstain** means withhold oneself from an action or self-indulgence. [There were six votes in favor, two against, and two *abstaining*; She *abstained* from drinking.] **Refrain** has to do with withholding an action temporarily, or checking a momentary desire: He *refrained* from scolding his child until the company left. To **forbear**, in its intransitive sense, is to exercise self-control, often out of motives of patience or charity. [Though impatient, the customer *forbore* to upbraid the harried sales clerk; The teacher *forbore* to report Johnnie's misbehavior to his parents.] See FORGO, FORSWEAR. **Antonyms**: *BEGIN, PERMIT*.

Figure 1
An excerpt from *Choose the Right Word* (CTRW) by S. I. Hayakawa. Copyright ©1987. Reprinted by arrangement with HarperCollins Publishers, Inc.

praise common in such writing. ***Placid*** *may have an unfavorable connotation in suggesting an unimaginative, bovine dullness of personality.*

Stylistic distinctions Stylistic variations of near-synonyms concern their level of **formality**, **concreteness**, **force**, **floridity**, and **familiarity** (Hovy 1990). Only the first three of these occur in CTRW. A sentence in CTRW expressing stylistic distinctions is this: ***Assistant*** *and* ***helper*** *are nearly identical except for the latter's greater informality.* Words that signal the degree of formality include *formal*, *informal*, *formality*, and *slang*. The degree of **concreteness** is signaled by words such as *abstract*, *concrete*, and *concretely*. Force can be signaled by words such as *emphatic* and *intensification*.

1.1.1 The Class Hierarchy of Distinctions. Following the analysis of the distinctions among near-synonyms of Edmonds and Hirst (2002), we derived the class hierarchy of distinctions presented in Figure 2. The top-level class DISTINCTIONS consists of DENOTATIONAL DISTINCTIONS, ATTITUDE, and STYLE. The last two are grouped together in a class ATTITUDE-STYLE DISTINCTIONS because they are expressed by similar syntactic constructions in the text of CTRW. Therefore the algorithm to be described in Section 2.2 will treat them together.

The leaf classes of DENOTATIONAL DISTINCTIONS are SUGGESTION, IMPLICATION, and DENOTATION; those of ATTITUDE are FAVORABLE, NEUTRAL, and PEJORATIVE; those of STYLE are FORMALITY, CONCRETENESS, and FORCE. All these leaf nodes have the attribute STRENGTH, which takes the values `low`, `medium`, and `high`. All the leaf nodes except those in the class STYLE have the attribute FREQUENCY, which takes the values `always`, `usually`, `sometimes`, `seldom`, and `never`. The DENOTATIONAL DISTINCTIONS have an additional attribute: the peripheral concept that is suggested, implied, or denoted.

1.2 The Clustered Model of Lexical Knowledge

Hirst (1995) and Edmonds and Hirst (2002) show that current models of lexical knowledge used in computational systems cannot account well for the properties of near-synonyms.

The conventional view is that the denotation of a lexical item is represented as a concept or a structure of concepts (i.e., a word sense is linked to the concept it lexicalizes), which are themselves organized into an ontology. The ontology is often language independent, or at least language neutral, so that it can be used in multilingual applications. Words that are nearly synonymous have to be linked to their own slightly different concepts. Hirst (1995) showed that such a model entails an awkward taxonomic proliferation of language-specific concepts at the fringes, thereby defeating the purpose of a language-independent ontology. Because this model defines words

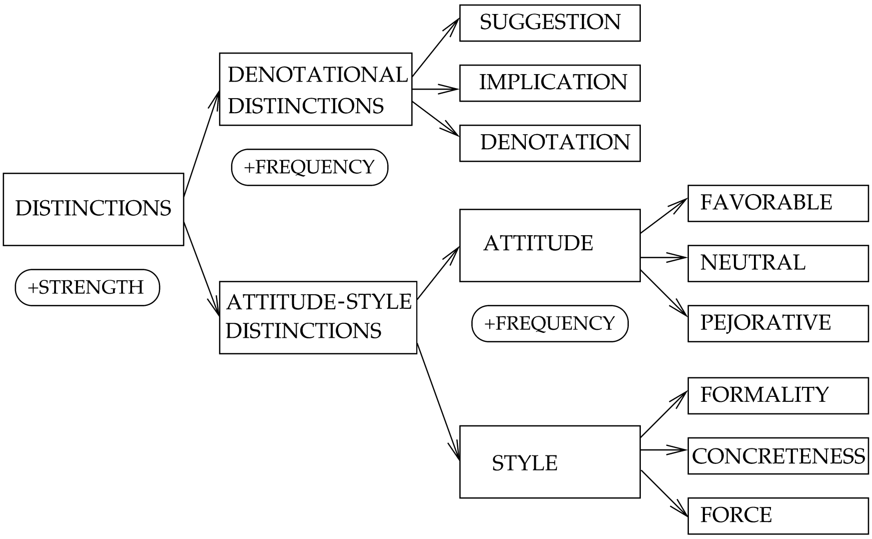


Figure 2
The class hierarchy of distinctions: rectangles represent classes, ovals represent attributes that a class and its descendants have.

in terms of necessary and sufficient truth conditions, it cannot account for indirect expressions of meaning or for fuzzy differences between near-synonyms.

Edmonds and Hirst (2002) modified this model to account for near-synonymy. The meaning of each word arises out of a context-dependent combination of a context-independent denotation and a set of explicit differences from its near-synonyms, much as in dictionaries of near-synonyms. Thus the meaning of a word consists both of necessary and sufficient conditions that allow the word to be selected by a lexical choice process and a set of nuances of indirect meaning that may be conveyed with different strengths. In this model, a conventional ontology is cut off at a coarse grain and the near-synonyms are clustered under a shared concept, rather than linking each word to a separate concept. The result is a *clustered model of lexical knowledge*. Thus, each cluster has a core denotation that represents the essential shared denotational meaning of its near-synonyms. The internal structure of a cluster is complex, representing semantic (or denotational), stylistic, and expressive (or attitudinal) differences between the near-synonyms. The differences or lexical nuances are expressed by means of peripheral concepts (for denotational nuances) or attributes (for nuances of style and attitude).

The clustered model has the advantage that it keeps the ontology language neutral by representing language-specific distinctions inside the cluster of near-synonyms. The near-synonyms of a core denotation in each language do not need to be in separate clusters; they can be part of one larger cross-linguistic cluster.

However, building such representations by hand is difficult and time-consuming, and Edmonds and Hirst (2002) completed only nine of them. Our goal in the present work is to build a knowledge base of these representations automatically by extracting the content of all the entries in a dictionary of near-synonym discrimination. Unlike lexical resources such as WordNet (Miller 1995), in which the words in synsets are considered “absolute” synonyms, ignoring any differences between them, and thesauri such as *Roget's* (Roget 1852) and *Macquarie* (Bernard 1987), which contain hierarchical groups of similar words, the knowledge base will include, in addition to the words that are near-synonyms, explicit explanations of differences between these words.

2. Building a Lexical Knowledge Base of Near-Synonym Differences

As we saw in Section 1, each entry in a dictionary of near-synonym discrimination lists a set of near-synonyms and describes the differences among them. We will use the term *cluster* in a broad sense to denote both the near-synonyms from an entry and their differences. Our aim is not only to automatically extract knowledge from one such dictionary in order to create a lexical knowledge base of near-synonyms (LKB of NS), but also to develop a general method that could be applied to any such dictionary with minimal adaptation. We rely on the hypothesis that the language of the entries contains enough regularity to allow automatic extraction of knowledge from them. Earlier versions of our method were described by Inkpen and Hirst (2001).

The task can be divided into two phases, treated by two consecutive modules, as shown in Figure 3. The first module, the *extraction module*, will be described in this section. The generic clusters produced by this module contain the concepts that near-synonyms may involve (the peripheral concepts) as simple strings. This generic LKB of NS can be adapted for use in any Natural Language Processing (NLP) application. The second module customizes the LKB of NS so that it satisfies the requirements of the particular system that is to employ it. This customization module transforms the

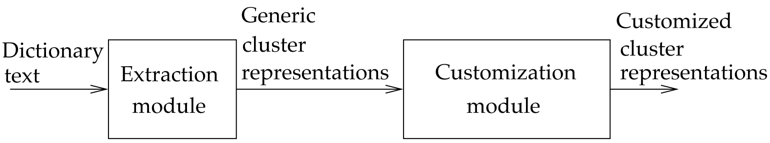

Dictionary text
Extraction module
Generic cluster representations
Customization module
Customized cluster representations

Figure 3
The two modules of the task.

strings from the generic clusters into concepts in the particular ontology. An example of a customization module will be described in Section 6.

The dictionary that we use is *Choose the Right Word* (Hayakawa 1994) (CTRW),[1] which was introduced in Section 1. CTRW contains 909 clusters, which group 5,452 near-synonyms (more precisely, near-synonym senses, because a word can be in more than one cluster) with a total of 14,138 sentences (excluding examples of usage), from which we derive the lexical knowledge base. An example of the results of this phase, corresponding to the second, third, and fourth sentence for the *absorb* cluster in Figure 1, is presented in Figure 4.

This section describes the extraction module, whose architecture is presented in Figure 5. It has two main parts. First, it learns extraction patterns; then it applies the patterns to extract differences between near-synonyms.

2.1 Preprocessing the Dictionary

After OCR scanning of CTRW and error correction, we used XML markup to segment the text of the dictionary into cluster name, cluster identifier, members (the near-synonyms in the cluster), entry (the textual description of the meaning of the near-synonyms and of the differences among them), cluster's part of speech, cross-references to other clusters, and antonyms list. Sentence boundaries were detected by using general heuristics, plus heuristics specific for this particular dictionary; for example, examples appear in square brackets and after a colon.

2.2 The Decision-List Learning Algorithm

Before the system can extract differences between near-synonyms, it needs to learn extraction patterns. For each leaf class in the hierarchy (Figure 2) the goal is to learn a set of words and expressions from CTRW—that is, extraction patterns—that characterizes descriptions of the class. Then, during the extraction phase, for each sentence (or fragment of a sentence) in CTRW the program will decide which leaf class is expressed, with what strength, and what frequency. We use a decision-list algorithm to learn sets of words and extraction patterns for the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS. These are split further for each leaf class, as explained in Section 2.3.

The algorithm we implemented is inspired by the work of Yarowsky (1995) on word sense disambiguation. He classified the senses of a word on the basis of other words that the given word co-occurs with. Collins and Singer (1999) classified proper names

[1] We are grateful to HarperCollins Publishers, Inc. for permission to use CTRW in this project.

Cluster: absorb, assimilate, digest, imbibe, incorporate, ingest

```
[absorb, usually, low, Formality]
[absorb, usually, medium, Suggestion, the taking in of liquids]
[absorb, sometimes, medium, Implication, the thoroughness of the action]
...
```

Figure 4
Example of distinctions extracted from CTRW.

as PERSON, ORGANIZATION, or LOCATION using contextual rules (that rely on other words appearing in the context of the proper names) and spelling rules (that rely on words in the proper names). Starting with a few spelling rules (using some proper-name features) in the decision list, their algorithm learns new contextual rules; using these rules, it then learns more spelling rules, and so on, in a process of mutual bootstrapping. Riloff and Jones (1999) learned domain-specific lexicons and extraction patterns (such as *shot in* $\langle x \rangle$ for the terrorism domain). They used a mutual bootstrapping technique to alternately select the best extraction pattern for a category and add its extractions to the semantic lexicon; the newly added entries in the lexicon help in the selection of the next best extraction pattern.

Our decision-list (DL) algorithm (Figure 6) is tailored for extraction from CTRW. Like the algorithm of Collins and Singer (1999), it learns two different types of rules: *Main rules* are for words that are significant for distinction classes; *auxiliary rules* are for frequency words, strength words, and comparison words. Mutual bootstrapping in the algorithm alternates between the two types. The idea behind the algorithm is that starting with a few main rules (seed words), the program selects examples containing them and learns a few auxiliary rules. Using these, it selects more examples and learns new main rules. It keeps iterating until no more new rules are learned.

The rules that the program learns are of the form $x \rightarrow h(x)$, meaning that word x is significant for the given class with confidence $h(x)$. All the rules for that class form a decision list that allows us to compute the confidence with which new patterns

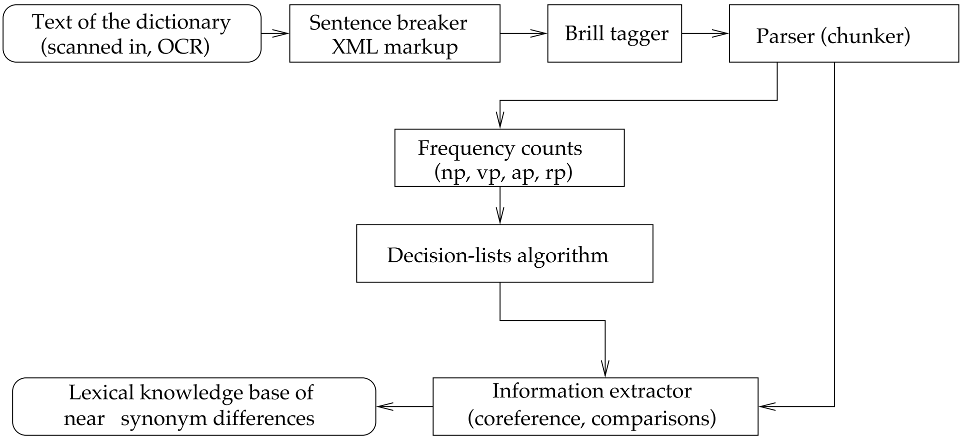

Text of the dictionary (scanned in, OCR)
Sentence breaker XML markup
Brill tagger
Parser (chunker)
Frequency counts (np, vp, ap, rp)
Decision-lists algorithm
Information extractor (coreference, comparisons)
Lexical knowledge base of near synonym differences

Figure 5
The architecture of the extraction module.

Input: Set E of training examples, class, main seed words for class, part of speech (pos) for words that are to be in mainDL, and pos for words that are to be in auxDL.

Output: Two decision lists for the given class: main decision list (mainDL) and auxiliary decision list (auxDL), plus list E' of patterns for the class. (Each decision list contains rules of the form $x \rightarrow h(x)$, meaning that the word x is significant for that class with confidence $h(x)$ computed by Equation 1.)

1. Set $N = 10$, the maximum number of rules to be induced at each step.
2. Initialization: Set the mainDL to the set of main seed words (with confidence 0.99). Set E' to empty set.
3. Add to mainDL those words in chunks from E that have the same stem as any words already in mainDL. (For example, if *suggest* is in mainDL, add *suggests, suggesting, suggested, suggestion*.)
4. Select examples (chunks) from $E - E'$ that contain words in mainDL, and add them to E'.
5. Use E' to compute more auxiliary rules. For each word x not in any DL, compute the confidence $h(x)$ using Equation 1. Take the N highest values and add them to auxDL.
6. Select more examples from $E - E'$ using auxDL, and add them to E'. Stop if E' is unchanged.
7. Using the new E', compute more main rules. For each word x not in any DL, compute the confidence $h(x)$. Take the N highest values and add them to mainDL.
8. Go to step 3 unless E' is unchanged.

Figure 6
The decision-list learning algorithm.

are significant for the class. The confidence $h(x)$ for a word x is computed with the formula:

$$h(x) = \frac{\text{count}(x, E') + \alpha}{\text{count}(x, E) + k\alpha} \tag{1}$$

where E' is the set of patterns selected for the class, and E is the set of all input data. So, we count how many times x is in the patterns selected for the class versus the total number of occurrences in the training data. Following Collins and Singer (1999), $k = 2$, because there are two partitions (relevant and irrelevant for the class). $\alpha = 0.1$ is a smoothing parameter.

In order to obtain input data, we replace all the near-synonyms in the text of the dictionary with the term `near_syn`; then we chunk the text with Abney's chunker (Abney 1996). The training set E is composed of all the verb phrases, noun phrases, adjectival phrases, and adverbial phrases (denoted `vx`, `nx`, `ax`, `rx`, respectively) that occur more than t times in the text of the dictionary (where $t = 3$ in our experiments). Phrases that occur very few times are not likely to be significant patterns and eliminating them makes the process faster (fewer iterations are needed).

We apply the DL algorithm for each of the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS. The input to the algorithm is as follows: the set E of all chunks, the main seed words, and the restrictions on the part of speech of the words in main and auxiliary rules. For the class DENOTATIONAL DISTINCTIONS the main seed words are *suggest*, *imply*, *denote*, *mean*, *designate*, *connote*; the words in main rules are verbs and nouns, and the words in auxiliary rules are adverbs and modals. For the class ATTITUDE-STYLE DISTINCTIONS the main seed words are *formal*, *informal*, *pejorative*, *disapproval*, *favorable*, *abstract*, *concrete*; the words in main rules are adjectives and nouns, and the words in auxiliary rules are adverbs. For example, for the class DENOTATIONAL DISTINCTIONS, starting with the rule *suggest* → 0.99, the program selects examples such as these (where the numbers give the frequency in the training data):

```
[vx [md can] [vb suggest]]--150
[vx [rb sometimes] [vb suggest]]--12
```

Auxiliary rules are learned for the words *sometimes* and *can*, and using these rules, the program selects more examples such as these:

```
[vx [md can] [vb refer]]--268
[vx [md may] [rb sometimes] [vb imply]]--3
```

From these, new main rules are learned for the words *refer* and *imply*. With these rules, more auxiliary rules are selected—for the word *may* and so on.

The ATTITUDE and STYLE classes had to be considered together because both of them use adjectival comparisons. Examples of ATTITUDE-STYLE DISTINCTIONS class are these:

```
[ax [rbs most] [jj formal]]--54
[ax [rb much] [more more] [jj formal]]--9
[ax [rbs most] [jj concrete]]--5
```

2.3 Classification and Extraction

After we run the DL algorithm for the class DENOTATIONAL DISTINCTIONS, we split the words in the list of main rules into its three sub-classes, as shown in Figure 2. This sub-classification is manual for lack of a better procedure. Furthermore, some words can be insignificant for any class (e.g., the word *also*) or for the given class; during the sub-classification we mark them as OTHER. We repeat the same procedure for frequencies and strengths with the words in the auxiliary rules. The words marked as OTHER and the patterns that do not contain any word from the main rules are ignored in the next processing steps. Similarly, after we run the algorithm for the class ATTITUDE-STYLE DISTINCTIONS, we split the words in the list of main rules into its sub-classes and sub-sub-classes (Figure 2). Frequencies are computed from the auxiliary rule list, and strengths are computed by a module that resolves comparisons.

Once we had obtained the words and patterns for all the classes, we implemented an automatic knowledge-extraction program that takes each sentence in CTRW and tries to extract one or more pieces of knowledge from it. Examples of results after this stage are presented in Figure 4. The information extracted for denotational distinctions is the near-synonym itself, the class, frequency, and strength of the distinction, and the peripheral concept. At this stage, the peripheral concept is a string extracted from the sentence. Strength takes the value `low`, `medium`, or `high`; frequency takes the value `always`, `usually`, `sometimes`, `seldom`, or `never`. Default values (`usually` and `medium`) are used when the strength or the frequency are not specified in the sentence. The information extracted for attitudinal and stylistic distinctions is analogous.

The extraction program considers what near-synonyms each sentence fragment is about (most often expressed as the subject of the sentence), what the expressed distinction is, and with what frequency and relative strength. If it is a denotational distinction, then the peripheral concept involved has to be extracted too (from the object position in the sentence). Therefore, our program looks at the subject of the sentence (the first noun phrase before the main verb) and the object of the sentence (the first noun phrase after the main verb). This heuristic works for sentences that present one piece of information. There are many sentences that present two or more pieces of information. In such cases, the program splits a sentence into coordinated clauses (or coordinated verb phrases) by using a parser (Collins 1996) to distinguish when a coordinating conjunction (*and*, *but*, *whereas*) is conjoining two main clauses or two parts of a complex verb phrase. From 60 randomly selected sentences, 52 were correctly dealt with (41 needed no split, 11 were correctly split). Therefore, the accuracy was 86.6%. The eight mistakes included three sentences that were split but should not have been, and five that needed splitting but were not. The mistakes were mainly due to wrong parse trees.

When no information is extracted in this way, a few general patterns are matched with the sentence in order to extract the near-synonyms; an example of such pattern is: `To NS1 is to NS2 ....` There are also heuristics to retrieve compound subjects of the form *near-syn and near-syn* and *near-syn, near-syn, and near-syn*. Once the class is determined to be either DENOTATIONAL DISTINCTIONS or ATTITUDE-STYLE DISTINCTIONS, the target class (one of the leaves in the class hierarchy in Figure 2) is determined by using the manual partitions of the rules in the main decision list of the two classes.

Sometimes the subject of a sentence refers to a group of near-synonyms. For example, if the subject is *the remaining words*, our program needs to assert information about all the near-synonyms from the same cluster that were not yet mentioned in the text. In order to implement coreference resolution, we applied the same DL algorithm to retrieve expressions used in CTRW to refer to near-synonyms or groups of near-synonyms.

Sometimes CTRW describes stylistic and attitudinal distinctions relative to other near-synonyms in the cluster. Such comparisons are resolved in a simple way by considering only three absolute values: `low`, `medium`, `high`. We explicitly tell the system which words represent what absolute values of the corresponding distinction (e.g., *abstract* is at the low end of `Concreteness`) and how the comparison terms increase or decrease the absolute value (e.g., *less abstract* could mean a `medium` value of `Concreteness`).

2.4 Evaluation

Our program was able to extract 12,365 distinctions from 7,450 of the 14,138 sentences of CTRW. (The rest of the sentences usually do not contain directly expressed distinctions; for example: *A* ***terror-stricken*** *person who is drowning may in panic resist the efforts of someone who is trying to save him.*)

In order to evaluate the final results, we randomly selected 25 clusters as a development set, and another 25 clusters as a test set. The development set was used to tune the program by adding new patterns if they helped improve the results. The test set was used exclusively for testing. We built by hand a standard solution for each set. The baseline algorithm is to choose the default values whenever possible. There are no defaults for the near-synonyms the sentence is about or for peripheral concepts; therefore, for these, the baseline algorithm assigns the sentence subject and object, respectively, using only tuples extracted by the chunker.

The measures that we used for evaluating each piece of information extracted from a sentence fragment were *precision* and *recall*. The results to be evaluated have four components for ATTITUDE-STYLE DISTINCTIONS and five components for DENOTATIONAL DISTINCTIONS. There could be missing components (except strength and frequency, which take default values). Precision is the total number of correct components found (summed over all the sentences in the test set) divided by the total number of components found. Recall is the total number of correct components found divided by the number of components in the standard solution.

For example, for the sentence *Sometimes, however,* ***profit*** *can refer to gains outside the context of moneymaking*, the program obtains ⟨`profit`, `usually`, `medium`, `Denotation`, `gains outside the context of moneymaking`⟩, whereas the solution is ⟨`profit`, `sometimes`, `medium`, `Denotation`, `gains outside the context of money-making`⟩. The precision is .80 (four correct out of five found), and the recall is also .80 (four correct out of five in the standard solution).

Table 2 presents the results of the evaluation.[2] The first row of the table presents the results as a whole (all the components of the extracted lexical knowledge base). Our system increases precision by 36 percentage points and recall by 46 percentage points over baseline on the development set.[3] Recall and precision are both slightly higher still on the test set; this shows that the patterns added during the development stage were general.

The second row of the table gives the evaluation results for extracting only the class of the distinction expressed, ignoring the strengths, frequencies, and peripheral concepts. This allows for a more direct evaluation of the acquired extraction patterns. The baseline algorithm attained higher precision than in the case when all the components are considered because the default class `Denotation` is the most frequent in CTRW. Our algorithm attained slightly higher precision and recall on the development set than it did in the complete evaluation, probably due to a few cases in which the frequency and strength were incorrectly extracted, and slightly lower on the test set, probably due to some cases in which the frequency and strength were easy to extract correctly.

2.5 Conclusion

The result of this stage is a generic lexical knowledge base of near-synonym differences. In subsequent sections, it will be enriched with knowledge from other sources; information about the collocational behavior of the near-synonyms is added in Section 3, and

[2] The improvement over the baseline is statistically significant (at $p = 0.005$ level or better) for all the results presented in this article, except in one place to be noted later. Statistical significance tests were done with the paired t test, as described by Manning and Schütze (1999, pages 208–209).

[3] These values are improved over those of earlier systems presented by Inkpen and Hirst (2001).

Table 2
Precision and recall of the baseline and of our algorithm (for all the components and for the distinction class only; boldface indicates best results).

	Baseline algorithm		Our system (dev. set)		Our system (test set)	
	Precision	Recall	Precision	Recall	Precision	Recall
All	.40	.23	.76	.69	**.83**	**.73**
Class only	.49	.28	.79	.70	.82	.71

more distinctions acquired from machine-readable dictionaries are added in Section 4. To be used in a particular NLP system, the generic LKB of NS needs to be customized (Section 6). Section 7 shows how the customized LKB of NS can actually be used in Natural Language Generation (NLG).

The method for acquiring extraction patterns can be applied to other dictionaries of synonym differences. The extraction patterns that we used to build our LKB of NS are general enough to work on other dictionaries of English synonyms. To verify this, we applied the extraction programs presented in Section 2.3, without modification, to the usage notes in the *Merriam-Webster Online Dictionary*.[4] The distinctions expressed in these usage notes are similar to the explanations from CTRW, but the text of these notes is shorter and simpler. In a sample of 100 usage notes, we achieved a precision of 90% and a recall of 82%.

3. Adding Collocational Knowledge from Free Text

In this section, the lexical knowledge base of near-synonym differences will be enriched with knowledge about the collocational behavior of the near-synonyms. We take collocations here to be pairs of words that appear together, consecutively or separated by only a few non-content words, much more often than by chance. Our definition is purely statistical, and we make no claim that the collocations that we find have any element of idiomaticity; rather, we are simply determining the preferences of our near-synonyms for combining, or avoiding combining, with other words. For example *daunting task* is a preferred combination (a collocation, in our terms), whereas *daunting job* is less preferred (it should not be used in lexical choice unless there is no better alternative), and *daunting duty* is an *anti-collocation* (Pearce 2001) that sounds quite odd and must not be used in lexical choice.

There are three main steps in acquiring this knowledge, which are shown in Figure 7. The first two look in free text—first the British National Corpus, then the World Wide Web—for collocates of all near-synonyms in CTRW, removing any closed-class words (function words). For example, the phrase *defeat the enemy* will be treated as *defeat enemy*; we will refer to such pairs as bigrams, even if there were intervening words. The third step uses the *t*-test (Church et al. 1991) to classify less frequent or unobserved bigrams as less preferred collocations or anti-collocations. We outline the three steps below; a more-detailed discussion is presented by Inkpen and Hirst (2002).

[4] http://www.m-w.com/cgi-bin/dictionary

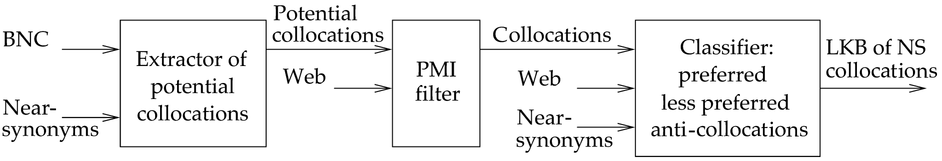


Figure 7
The three steps in acquiring collocational knowledge for near-synonyms.

3.1 Extracting Collocations from the British National Corpus

In step 1 of our procedure, our data was the 100-million-word part-of-speech-tagged British National Corpus (BNC).[5] Only 2.61% of the near-synonyms are absent from the BNC; and only 2.63% occur between one and five times. We first preprocessed the BNC by removing all words tagged as closed-class and all the proper names, and then used the Ngram Statistics Package[6] (Pedersen and Banerjee 2003), which counts bigram (or *n*-gram) frequencies in a corpus and computes various statistics to measure the degree of association between two words: pointwise mutual information (MI), Dice, chi-square (χ^2), log-likelihood (LL), and Fisher's exact test. (See Manning and Schütze [1999] for a review of statistical methods that can be used to identify collocations.)

Because these five measures rank potential collocations in different ways and have different advantages and drawbacks, we decided to combine them by choosing as a collocation any bigram that was ranked by at least two of the measures as one of that measure's T most-highly ranked bigrams; the threshold T may differ for each measure. Lower values for T increase the precision (reduce the chance of accepting noncollocations) but may not get many collocations for some of the near-synonyms; higher values increase the recall at the expense of lower precision. Because there is no principled way of choosing these values, we opted for higher recall, with step 2 of the process (Section 3.2) filtering out many noncollocations in order to increase the precision. We took the first 200,000 bigrams selected by each measure, except for Fisher's measure for which we took all 435,000 that were ranked equal top. From these lists, we retained only those bigrams in which one of the words is a near-synonym in CTRW.[7]

[5] http://www.hcu.ox.ac.uk/BNC/

[6] http://www.d.umn.edu/~tpederse/nsp.html. We used version 0.4, known at the time as the Bigram Statistics Package (BSP).

[7] Collocations of a near-synonym with the wrong part of speech were not considered (the collocations are tagged), but when a near-synonym has more than one major sense, collocations for senses other than the one required in the cluster could be retrieved. For example, for the cluster *job, task, duty*, and so on, the collocation *import duty* is likely to be for a different sense of *duty* (the tariff sense). Therefore disambiguation is required (assuming one sense per collocation). We experimented with a simple Lesk-style method (Lesk 1986). For each collocation, instances from the corpus were retrieved, and the content words surrounding the collocations were collected. This set of words was then intersected with the entry for the near-synonym in CTRW. A non-empty intersection suggests that the collocation and the entry use the near-synonym in the same sense. If the intersection was empty, the collocation was not retained. However, we ran the disambiguation algorithm only for a subset of CTRW, and then abandoned it, because hand-annotated data are needed to evaluate how well it works and because it was very time-consuming (due to the need to retrieve corpus instances for each collocation). Moreover, skipping disambiguation is relatively harmless because including these wrong senses in the final lexical knowledge base of near-synonym collocations will not hurt. For example, if the collocation *import duty* is associated with the cluster *job, duty*, etc., it simply will not be used in practice because the concepts of *import* and this sense of *duty* are unlikely to occur together in coherent interlingual input.

3.2 Filtering with Mutual Information from Web Data

In the previous step we emphasized recall at the expense of precision: Because of the relatively small size of the BNC, it is possible that the classification of a bigram as a collocation in the BNC was due to chance. However, the World Wide Web (the portion indexed by search engines) is big enough that a result is more reliable. So we can use frequency on the Web to filter out the more dubious collocations found in the previous step.[8] We did this for each putative collocation by counting its occurrence on the Web, the occurrence of each component word, and computing the pointwise mutual information (PMI) between the words. Only those whose pointwise mutual information exceeded a threshold T_{pmi} were retained.

More specifically, if w is a word that collocates with one of the near-synonyms x in a cluster, a proxy PMI_{prox} for the pointwise mutual information between the words can be given by the ratio

$$\frac{P(w,x)}{P(x)} = \frac{n_{wx}}{n_x}$$

where n_{wx} and n_x are the number of occurrences of wx and x, respectively. The formula does not include $P(w)$ because it is the same for various x. We used an interface to the AltaVista search engine to do the counts, using the number of hits (i.e., matching documents) as a proxy for the actual number of bigrams.[9] The threshold T_{pmi} for PMI_{prox} was determined empirically by finding the value that optimized results on a standard solution, constructed as follows. We selected three clusters from CTRW, with a total of 24 near-synonyms. For these, we obtained 916 candidate collocations from the BNC. Two human judges (computational linguistics students, native speakers of English) were asked to mark the true collocations (what they considered to be good usage in English). The candidate pairs were presented to the judges in random order, and each was presented twice.[10] A bigram was considered to be a true collocation only if both judges considered it so. We used this standard solution to choose the value of T_{pmi} that maximizes the accuracy of the filtering program. Accuracy on the test set was 68.3% (compared to approximately 50% for random choice).

3.3 Finding Less Preferred Collocations and Anti-Collocations

In seeking knowledge of less preferred collocations and anti-collocations, we are looking for bigrams that occur infrequently or not at all. The low frequency or absence of a

[8] Why not just use the Web and skip the BNC completely? Because we would then have to count Web occurrences of every near-synonym in CTRW combined with every content word in English. Using the BNC as a first-pass filter vastly reduces the search space.

[9] The collocations were initially acquired from the BNC with the right part of speech for the near-synonym because the BNC is part-of-speech-tagged, but on the Web there are no part-of-speech tags; therefore a few inappropriate instances may be included in the counts.

[10] One judge was consistent (judged a collocation in the same way both times it appeared) in 90.4% of the cases and the other in 88% of the cases. The agreement between the two judges was 78% (computed in a strict way; i.e., we considered agreement only when the two judges had the same opinion including the cases when they were not consistent), yielding $\kappa = .51$ with a 95% confidence interval of 0.47 to 0.55. (The notion of confidence intervals for κ is defined, e.g., by Sim and Wright [2005]. The computations were done with the PEPI statistical package [http://sagebrushpress.com/pepibook.html].) These figures show that the task is not easy for humans.

bigram in the BNC may be due to chance. However, the World Wide Web is big enough that a negative result is more reliable. So we can again use frequency on the Web—this time to determine whether a bigram that was infrequent or unseen in the BNC is truly a less preferred collocation or anti-collocation.

The bigrams of interest now are those in which collocates for a near-synonym that were found in step 1 and filtered in step 2 are combined with another member of the same near-synonym cluster. For example, if the collocation *daunting task* was found, we now look on the Web for the apparent noncollocations *daunting job, daunting duty*, and other combinations of *daunting* with near-synonyms of *task*. A low number of co-occurrences indicates a less preferred collocation or anti-collocation. We employ the *t*-test, following Manning and Schütze (1999, pages 166–168), to look for differences.

The collocations of each near-synonym with a given collocate are grouped in three classes, depending on the *t* values of pairwise collocations. A *t* value comparing each collocation and the collocation with maximum frequency is computed, and so is the *t* value between each collocation and the collocation with minimum frequency. Table 3 presents an example.

After the *t*-test scores were computed, a set of thresholds was determined to classify the collocations in the three groups: preferred collocations, less preferred collocations, and anti-collocations. Again, we used a standard solution in the procedure. Two judges manually classified a sample of 2,838 potential collocations obtained for the same three clusters of near-synonyms from 401 collocations that remained after filtering. They were instructed to mark as preferred collocations all the potential collocations that they considered good idiomatic use of language, as anti-collocations the ones that they would not normally use, and as less preferred collocations the ones that they were not comfortable classifying in either of the other two classes. When the judges agreed, the class was clear. When they did not agree, we used simple rules, such as these: When one judge chose the class-preferred collocation, and the other chose the class anti-collocation, the class in the solution was less preferred collocation (because such cases seemed to be difficult and controversial); when one chose preferred collocation, and the other chose less preferred collocation, the class in the solution was preferred collocation; when one chose anti-collocation, and the other chose less preferred collocation, the class in the solution was anti-collocation. The agreement between judges was 84%, $\kappa = 0.66$ (with a 95% confidence interval of 0.63 to 0.68).

Table 3
Example of counts, mutual information scores, and *t*-test scores for the collocate *daunting* with near-synonyms of *task*. The second column shows the number of hits for the collocation *daunting x*, where *x* is the near-synonym in the first column. The third column shows $\mathrm{PMI}_{\mathrm{prox}}$ (scaled by 10^5 for readability), the fourth column, the *t* values between the collocation with maximum frequency (*daunting task*) and *daunting x*, and the last column, the *t*-test between *daunting x* and the collocations with minimum frequency (*daunting stint* and *daunting hitch*).

x	**Hits**	$\mathrm{PMI}_{\mathrm{prox}}$	***t* max**	***t* min**
task	63,573	0.011662	–	252.07
job	485	0.000022	249.19	22.02
assignment	297	0.000120	250.30	17.23
chore	96	0.151899	251.50	9.80
duty	23	0.000022	251.93	4.80
stint	0	0	252.07	–
hitch	0	0	252.07	–

Table 4
Example of results of our program for collocations of near-synonyms in the *task* cluster. √ marks preferred collocations, ? marks less preferred collocations, and ∗ marks anti-collocations. The combined opinion of the judges about the same pairs of words is shown in parentheses.

	Collocates		
Near-synonyms	*daunting*	*particular*	*tough*
task	√ (√)	√ (√)	√ (√)
job	? (√)	√ (√)	√ (√)
assignment	∗ (√)	√ (√)	√ (√)
chore	∗ (∗)	? (√)	∗ (?)
duty	∗ (?)	√ (√)	∗ (∗)
stint	∗ (∗)	∗ (?)	∗ (?)
hitch	∗ (∗)	∗ (?)	∗ (∗)

We used this standard solution as training data to C4.5[11] to learn a decision tree for the three-way classifier. The features in the decision tree are the t-test between each collocation and the collocation from the same group that has maximum frequency on the Web, and the t-test between the current collocation and the collocation that has minimum frequency (as presented in Table 3). We did 10-fold cross-validation to estimate the accuracy on unseen data. The average accuracy was 84.1%, with a standard error of 0.5%; the baseline of always choosing the most frequent class, anti-collocations, yields 71.4%. We also experimented with including PMI_{prox} as a feature in the decision tree, and with manually choosing thresholds (without a decision tree) for the three-way classification, but the results were poorer. The three-way classifier can fix some of the mistakes of the PMI filter: If a wrong collocation remains after the PMI filter, the classifier can classify it in the anti-collocations class. We conclude that the acquired collocational knowledge has acceptable quality.

3.4 Results

We obtained 1,350,398 distinct bigrams that occurred at least four times. We selected collocations for all 909 clusters in CTRW (5,452 near-synonyms in total). Table 4 presents an example of results for collocational classification of bigrams, where √ marks preferred collocations, ? marks less preferred collocations, and ∗ marks anti-collocations. This gave us a lexical knowledge base of near-synonym collocational behavior. An example of collocations extracted for the near-synonym *task* is presented in Table 5, where the columns are, in order, the name of the measure, the rank given by the measure, and the value of the measure.

4. Adding Knowledge from Machine-Readable Dictionaries

Information about near-synonym differences can be found in other types of dictionaries besides those explicitly on near-synonyms. Although conventional dictionaries, unlike CTRW, treat each word in isolation, they may nonetheless contain useful information

[11] http://www.cse.unsw.edu.au/∼quinlan

Table 5
Example of collocations extracted for the near-synonym *task*. The first collocation was selected (ranked in the set of first T collocations) by four measures; the second collocation was selected by two measures.

Collocation	Measure	Rank	Score
daunting/A task/N	MI	24,887	10.85
	LL	5,998	907.96
	χ^2	16,341	122,196.82
	Dice	2,766	0.02
repetitive/A task/N	MI	64,110	6.77
	χ^2	330,563	430.40

about near-synonyms because some definitions express a distinction relative to another near-synonym. From the SGML-marked-up text of the *Macquarie Dictionary*[12] (Delbridge et al. 1987), we extracted the definitions of the near-synonyms in CTRW for the expected part of speech that contained another near-synonym from the same cluster. For example, for the CTRW cluster *burlesque, caricature, mimicry, parody, takeoff, travesty*, one definition extracted for the near-synonym *burlesque* was *any ludicrous take-off or debasing caricature* because it contains *caricature* from the same cluster. A series of patterns was used to extract the difference between the two near-synonyms wherever possible. For the *burlesque* example, the extracted information was

⟨`burlesque, usually, medium, Denotation, ludicrous`⟩,
⟨`burlesque, usually, medium, Denotation, debasing`⟩.

The number of new denotational distinctions acquired by this method was 5,731.

We also obtained additional information from the *General Inquirer*[13] (Stone et al. 1966), a computational lexicon that classifies each word in it according to an extendable number of categories, such as pleasure, pain, virtue, and vice; overstatement and understatement; and places and locations. The category of interest here is *Positiv/Negativ*. There are 1,915 words marked as *Positiv* (not including words for *yes*, which is a separate category of 20 entries), and 2,291 words marked as *Negativ* (not including the separate category of *no* in the sense of refusal). For each near-synonym in CTRW, we used this information to add a favorable or unfavorable attitudinal distinction accordingly. If there was more than one entry (several senses) for the same word, the attitude was asserted only if the majority of the senses had the same marker. The number of attitudinal distinctions acquired by this method was 5,358. (An attempt to use the occasional markers for formality in WordNet in a similar manner resulted in only 11 new distinctions.)

As the knowledge from each source is merged with the LKB, it must be checked for consistency in order to detect conflicts and resolve them. The algorithm for resolving conflicts is a voting scheme based on the intuition that neutral votes should have less weight than votes for the two extremes. The algorithm outputs a list of the conflicts and a proposed solution. This list can be easily inspected by a human, who can change

[12] http://www.macquariedictionary.com.au/
[13] http://www.wjh.harvard.edu/~inquirer/

Cluster: mistake=blooper=blunder=boner=contretemps=error=faux pas=goof=slip=solecism=
...
a near_syn(blunder) is a blatant near_syn(error), usually one involving behavior or judgment, and implying an ignorant or uninformed assessment of a situation

```
<blunder, usually, medium, Implications, an ignorant or uninformed
assessment of a situation>
```

near_syn(slip) emphasizes the accidental rather than ignorant character of a near_syn(mistake) and is often used to mean the careless divulging of secret or private information

```
<slip, usually, high, Denotations, the accidental rather than ignorant
character of a NS_mistake>
<slip, usually, high, Denotations, the accidental rather than ignorant
character of a NS_mistake>
```

near_syn(blooper), an informal term, is usually applied to particularly infelicitous mix-ups of speech, such as "rit of fellus jage" for "fit of jealous rage"

```
<blooper, usually, medium, Denotations, to particularly infelicitous mix-ups
of speech, such as rit of fellus jage for fit of jealous rage>
<blooper, usually, low, Formality>
```

...

Figure 8
Fragment of the representation of the *error* cluster (prior to customization).

the solution of the conflict in the final LKB of NS, if desired. The consistency-checking program found 302 conflicts for the merged LKB of 23,469 distinctions. After conflict resolution, 22,932 distinctions remained. Figure 8 shows a fragment of the knowledge extracted for the near-synonyms of *error* after merging and conflict resolution.

5. Related Work

5.1 Building Lexical Resources

Lexical resources for natural language processing have also been derived from other dictionaries and knowledge sources. The ACQUILEX[14] Project explored the utility of constructing a multilingual lexical knowledge base (LKB) from machine-readable versions of conventional dictionaries. Ide and Véronis (1994) argue that it is not possible to build a lexical knowledge base from a machine-readable dictionary (MRD) because the information it contains may be incomplete, or it may contain circularities. It is possible to combine information from multiple MRDs or to enhance an existing LKB, they say, although human supervision may be needed.

Automatically extracting world knowledge from MRDs was attempted by projects such as MindNet at Microsoft Research (Richardson, Dolan, and Vanderwende 1998), and Barrièrre and Popowich's (1996) project, which learns from children's dictionaries. IS-A hierarchies have been learned automatically from MRDs (Hearst 1992) and from corpora (Caraballo [1999] among others).

[14] http://www.cl.cam.ac.uk/Research/NL/acquilex/acqhome.html

Research on merging information from various lexical resources is related to the present work in the sense that the consistency issues to be resolved are similar. One example is the construction of Unified Medical Language System (UMLS)[15] (Lindberg, Humphreys, and McCray 1993), in the medical domain. UMLS takes a wide range of lexical and ontological resources and brings them together as a single resource. Most of this work is done manually at the moment.

5.2 Acquiring Collocational Knowledge

There has been much research on extracting collocations for different applications. Like Church et al. (1991), we use the *t*-test and mutual information (MI), but unlike them we use the Web as a corpus for this task (and a modified form of mutual information), and we distinguish three types of collocations. Pearce (2001) improved the quality of retrieved collocations by using synonyms from WordNet (Pearce 2001); a pair of words was considered a collocation if one of the words significantly prefers only one (or several) of the synonyms of the other word. For example, *emotional baggage* is a good collocation because *baggage* and *luggage* are in the same synset and ∗*emotional luggage* is not a collocation. Unlike Pearce, we use a combination of *t*-test and MI, not just frequency counts, to classify collocations.

There are two typical approaches to collocations in previous NLG systems: the use of phrasal templates in the form of canned phrases, and the use of automatically extracted collocations for unification-based generation (McKeown and Radev 2000). Statistical NLG systems (such as Nitrogen [Langkilde and Knight 1998]) make good use of the most frequent words and their collocations, but such systems cannot choose a less-frequent synonym that may be more appropriate for conveying desired nuances of meaning if the synonym is not a frequent word.

Turney (2001) used mutual information to choose the best answer to questions about near-synonyms in the Test of English as a Foreign Language (TOEFL) and English as a Second Language (ESL). Given a problem word (with or without context) and four alternative words, the question is to choose the alternative most similar in meaning to the problem word (the problem here is to detect similarities, whereas in our work differences are detected). His work is based on the assumption that two synonyms are likely to occur in the same document (on the Web). This can be true if the author needs to avoid repeating the same word, but not true when the synonym is of secondary importance in a text. The alternative that has the highest pointwise mutual information for information retrieval (PMI-IR) with the problem word is selected as the answer. We used the same measure in Section 3.3—the mutual information between a collocation and a collocate that has the potential to discriminate between near-synonyms. Both works use the Web as a corpus, and a search engine to estimate the mutual information scores.

5.3 Near-Synonyms

As noted in the introduction, our work is based on that of Edmonds and Hirst (2002) and Hirst (1995), in particular the model for representing the meaning of the near-synonyms presented in Section 1.2 and the preference satisfaction mechanism used in Section 7.

[15] http://www.nml.nih.gov/research/umls/

Other related research involving differences between near-synonyms has a linguistic or lexicographic, rather than computational, flavor. Apresjan built a bilingual dictionary of synonyms, more specifically a dictionary of English synonyms explained in Russian (Apresjan et al. 1980). It contains 400 entries selected from the approximately 2,500 entries from *Webster's New Dictionary of Synonyms*, but reorganized by splitting or merging clusters of synonyms, guided by lexicographic principles described by Apresjan (2000). An entry includes the following types of differences: semantic, evaluative, associative and connotational, and differences in emphasis or logical stress. These differences are similar to the ones used in our work.

Gao (2001) studied the distinctions between near-synonym verbs, more specifically Chinese physical action verbs such as verbs of cutting, putting, throwing, touching, and lying. Her dissertation presents an analysis of the types of semantic distinctions relevant to these verbs, and how they can be arranged into hierarchies on the basis of their semantic closeness.

Ploux and Ji (2003) investigated the question of which words should be considered near-synonyms, without interest in their nuances of meaning. They merged clusters of near-synonyms from several dictionaries in English and French and represented them in a geometric space. In our work, the words that are considered near-synonyms are taken from CTRW; a different dictionary of synonyms may present slightly different views. For example, a cluster may contain some extra words, some missing words, or sometimes the clustering could be done in a different way. A different approach is to automatically acquire near-synonyms from free text. Lin et al. (2003) acquire words that are related by contextual similarity and then filter out the antonyms by using a small set of manually determined patterns (such as "either X or Y") to construct Web queries for pairs of candidate words. The problem of this approach is that it still includes words that are in relations other than near-synonymy.

6. Customizing the Lexical Knowledge Base of Near-Synonym Differences

The initial LKB of NS built in Sections 2 to 4 is a general one, and it could, in principle, be used in any (English) NLP system. For example, it could be used in the lexical-analysis or lexical-choice phase of machine translation. Figure 9 shows that during the analysis phase, a lexical knowledge base of near-synonym differences in the source language is used, together with the context, to determine the set of nuances that are expressed in the source-language text (in the figure, the source language is French and the target language is English). In the generation phase, these nuances become *preferences* for the lexical-choice process. Not only must the target-language text express the same meaning as the source-language text (necessary condition), but the choice of words should satisfy the *preferences* as much as possible.

In order to be integrated with the other components of the NLP system, the LKB will probably need some adaptation—in particular, the core denotations and the peripheral concepts will need to be mapped to the ontology that the system employs. This might be a general-purpose ontology, such as Cyc (Lenat 1995) and WordNet, or an ontology built specially for the system (such as Mikrokosmos (Mahesh and Nirenburg 1995) or domain-specific ontologies). In this section, we focus on the generation phase of an interlingual machine translation system, specifically the lexical-choice process, and show how the LKB was adapted for Xenon, a natural language generation system. Xenon is a general-purpose NLG system that exploits our LKB of NS. To implement Xenon,

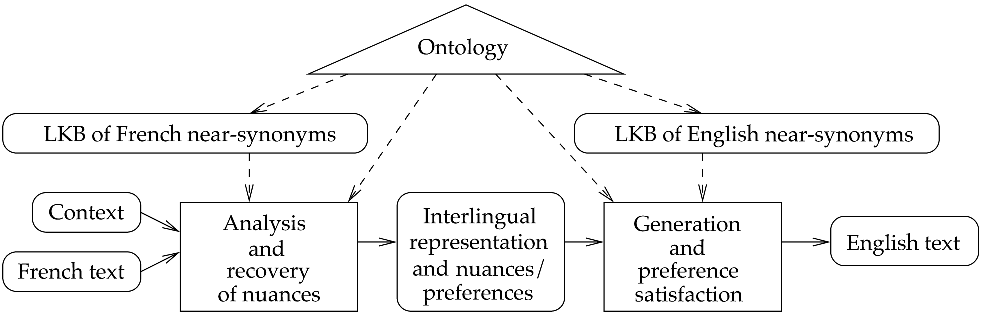

Ontology
LKB of French near-synonyms
LKB of English near-synonyms
Context
French text
Analysis and recovery of nuances
Interlingual representation and nuances/ preferences
Generation and preference satisfaction
English text

Figure 9
Lexical analysis and choice in machine translation; adapted from Edmonds and Hirst (2002). The solid lines show the flow of data: input, intermediate representations, and output; the dashed lines show the flow of knowledge from the knowledge sources to the analysis and the generation module. The rectangles denote the main processing modules; the rest of the boxes denote data or knowledge sources.

we modified the lexical-choice component of a preexisting NLG system, HALogen (Langkilde 2000; Langkilde and Knight 1998), to handle knowledge about the near-synonym differences. (Xenon will be described in detail in Section 7.) This required customization of the LKB to the Sensus ontology (Knight and Luk 1994) that HALogen uses as its representation.

Customization of the core denotations for Xenon was straightforward. The core denotation of a cluster is a metaconcept representing the disjunction of all the Sensus concepts that could correspond to the near-synonyms in a cluster. The names of metaconcepts, which must be distinct, are formed by the prefix `generic`, followed by the name of the first near-synonym in the cluster and the part of speech. For example, if the cluster is *lie, falsehood, fib, prevarication, rationalization, untruth*, the name of the cluster is `generic_lie_n`.

Customizing the peripheral concepts, which are initially expressed as strings, could include parsing the strings and mapping the resulting syntactic representation into a semantic representation. For Xenon, however, we implemented a set of 22 simple rules that extract the actual peripheral concepts from the initial peripheral strings. A transformation rule takes a string of words part-of-speech tagged and extracts a main word, several roles, and fillers for the roles. The fillers can be words or recursive structures. In Xenon, the words used in these representations are not sense-disambiguated. Here are two examples of input strings and extracted peripheral concepts:

```
"an embarrassing breach of etiquette"
=> (C / breach :GPI etiquette :MOD embarrassing)

"to an embarrassing or awkward occurrence"
=> (C / occurrence :MOD (OR embarrassing awkward))
```

The roles used in these examples are MOD (modifier) and GPI (generalized possession inverse). The rules that were used for these two examples are these:

```
Adj Noun1 of Noun2 => (C / Noun1 :GPI Noun2 :MOD Adj)
Adj1 or Adj2 Noun => (C / Noun :MOD (OR Adj1 Adj2))
```

We evaluated our customization of the LKB on a hand-built standard solution for a set of peripheral strings: 139 strings to be used as a test set and 97 strings to be used as a development set. The rules achieved a coverage of 75% on the test set with an accuracy of 55%.[16] In contrast, a baseline algorithm of taking the first word in each string as the peripheral concept covers 100% of the strings, but with only 16% accuracy.

Figure 10 shows the full customized representation for the near-synonyms of *error*, derived from the initial representation that was shown earlier in Figure 8. (See Inkpen [2003] for more examples of customized clusters.) The peripheral concepts are factored out, and the list of distinctions contains pointers to them. This allows peripheral concepts to be shared by two or more near-synonyms.

7. Xenon: An NLG System that Uses Knowledge of Near-Synonym Differences

This section presents Xenon, a large-scale NLG system that uses the lexical knowledge-base of near-synonyms customized in Section 6. Xenon integrates a new near-synonym choice module with the sentence realization system HALogen[17] (Langkilde 2000; Langkilde and Knight 1998). HALogen is a broad-coverage general-purpose natural language sentence generation system that combines symbolic rules with linguistic information gathered statistically from large text corpora. The internal architecture of HALogen is presented in Figure 11. A forest of all possible sentences (combinations of words) for the input is built, and the sentences are then ranked according to an *n*-gram language model in order to choose the most likely one as output.

Figure 12 presents the architecture of Xenon. The input is a semantic representation and a set of preferences to be satisfied. The final output is a set of sentences and their scores. A concrete example of input and output is shown in Figure 13. Note that HALogen may generate some ungrammatical constructs, but they are (usually) assigned lower scores. The first sentence (the highest ranked) is considered to be the solution.

7.1 Metaconcepts

The semantic representation input to Xenon is represented, like the input to HALogen, in an interlingua developed at University of Southern California/Information Sciences Institute (USC/ISI).[18] As described by Langkilde-Geary (2002b), this language contains a specified set of 40 roles, whose fillers can be either words, concepts from Sensus (Knight and Luk 1994), or complex interlingual representations. The interlingual representations may be underspecified: If some information needed by HALogen is not present, it will use its corpus-derived statistical information to

[16] We found that sometimes a rule would extract only a fragment of the expected configuration of concepts but still provided useful knowledge; however, such cases were not considered to be correct in this evaluation, which did not allow credit for partial correctness. For example, if the near-synonym *command* denotes `the/TD stated/VB demand/NN of/IN a/TD superior/JJ`, the expected peripheral concept is `(C1 / demand :GPI superior :MOD stated)`. If the program extracted only `(C1 / demand :GPI superior)`, the result was not considered correct, but the information might still help in an NLP system.

[17] http://www.isi.edu/licensed-sw/halogen/

[18] http://www.isi.edu/licensed-sw/halogen/interlingua.html

```
(defcluster generic_mistake_n
  :syns (mistake blooper blunder boner contretemps error faux_pas goof
         slip solecism)
  :senses (mistake#n#1 blooper#n#1 blunder#n#1 boner#n#1 error#n#1
           faux_pas#n#1 slip#n#1 slip#n#2 solecism#n#1)
  :core (ROOT GENERIC_MISTAKE (OR |fault,error| |boner| |gaffe| |slipup|))
  :periph (
     (P1  (C1 / deviation))
     (P2  (C1 / sin))
     (P3  (C1 / assessment :MOD (*OR* ignorant uninformed)))
     (P4  (C1 / accidental))
     (P5  (C1 / careless))
     (P6  (C1 / indefensible))
     (P7  (C1 / occurrence :MOD (*OR* embarrassing awkward)))
     (P8  (C1 / (*OR* action opinion judgment)))
     (P9  (C1 / (*OR* gross stupid)))
     (P10  (C1 / belief))
     (P11  (C1 / manners))
   )
  :distinctions
  ((error usually medium Implication P1)
   (error usually medium Denotation P2)
   (blunder usually medium Implication P3)
   (slip usually high Denotation P4)
   (slip usually medium Denotation P5)
   (blooper low Formality)
   (goof usually medium Denotation P6)
   (goof medium Formality)
   (contretemps usually medium Denotation P7)
   (mistake usually medium Denotation P8)
   (blunder usually medium Denotation P9)
   (error usually medium Denotation P10)
   (faux_pas usually medium Denotation P11)
   (mistake usually medium Favourable :agent)
   (blunder usually medium Favourable :agent)
   (boner usually medium Pejorative :agent)
   (contretemps usually medium Favourable :agent)
   (error usually medium Favourable :agent)
   (faux_pas usually medium Pejorative :agent)
   (goof usually medium Favourable :agent)
   (slip usually medium Favourable :agent)
   (solecism usually medium Favourable :agent)))
```

Figure 10
The final representation of the *error* cluster.

make choices. Xenon extends this representation language by adding metaconcepts that correspond to the core denotation of the clusters of near-synonyms. For example, in Figure 13, the metaconcept is generic_lie_n. As explained in Section 6, metaconcepts may be seen as a disjunction of all the senses of the near-synonyms of the cluster.

7.2 Near-Synonym Choice

The near-synonym choice module has to choose the most appropriate near-synonym from the cluster specified in the input. It computes a satisfaction score that becomes a weight (to be explained in section 7.3) for each near-synonym in the cluster. HALogen makes the final choice by adding these weights to *n*-gram probabilities from its language model (more precisely, the negative logarithms of these values) and choosing

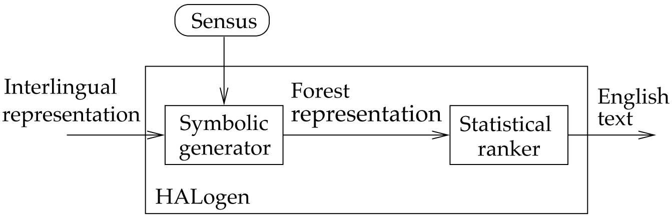


Figure 11
The architecture of the sentence realizer HALogen.

the highest-ranked sentence. For example, the expanded representation of the input in Figure 13 is presented in Figure 14. The near-synonym choice module gives higher weight to *fib* because it satisfies the preferences better than the other near-synonyms in the cluster, *lie, falsehood, fib, prevarication, rationalization,* and *untruth.*

7.3 Preferences and Similarity of Distinctions

The preferences that are input to Xenon could be given by the user, or they could come from an analysis module if Xenon is used in a machine translation system (corresponding to nuances of near-synonyms in a different language, see Figure 9). The preferences, like the distinctions expressed in the LKB of NS, are of three types: attitudinal, stylistic, and denotational. Examples of each:

```
(low formality)
(disfavour :agent)
(imply (C / assessment :MOD ( M / (*OR* ignorant uninformed)).
```

The formalism for expressing preferences is from I-Saurus (Edmonds 1999). The preferences are transformed internally into pseudodistinctions that have the same form as the corresponding type of distinctions so that they can be directly compared with the distinctions. The pseudodistinctions corresponding to the previous examples are these:

```
(-- low Formality)
(- always high Pejorative :agent)
(- always medium Implication
   (C/assessment :MOD (M/(OR ignorant uninformed)).
```

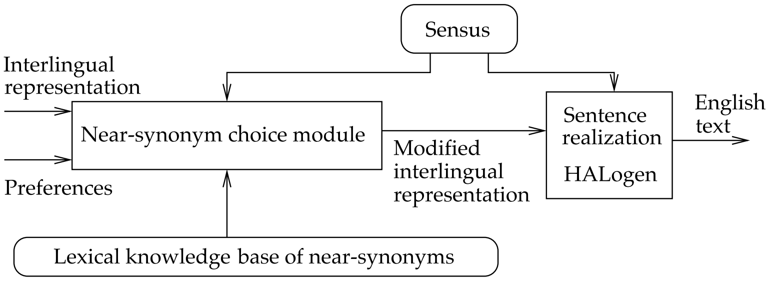

Sensus
Interlingual representation
Near-synonym choice module
Preferences
Modified interlingual representation
Sentence realization
HALogen
English text
Lexical knowledge base of near-synonyms

Figure 12
The architecture of the natural language generation system Xenon.

```
Input:
(A9 / tell
   :agent (V9 / boy)
   :object (O9 / generic_lie_n))

Input preferences:
 ((DISFAVOUR :AGENT) (LOW FORMALITY) (DENOTE (C1 / TRIVIAL)))

Output sentences:
The boy told fibs.    -40.8177
Boy told fibs.    -42.3818
Boys told fibs.    -42.7857
A boy told fibs.    -43.0738
Some boys told fibs.    -46.4388
Any boy told fibs.    -50.0306
An boy told fibs.    -50.15
Were told fibs by the boy.    -55.3801
```

Figure 13
Example of input and output of Xenon.

```
(A9 / tell
   :agent (V9 / boy)
   :object
    (OR (e1 / (:CAT NN :LEX "lie") :WEIGHT 1.0e-30)
        (e2 / (:CAT NN :LEX "falsehood") :WEIGHT 6.937703e-8)
        (e3 / (:CAT NN :LEX "fib") :WEIGHT 1.0)
        (e4 / (:CAT NN :LEX "prevarication") :WEIGHT 1.0e-30)
        (e5 / (:CAT NN :LEX "rationalization") :WEIGHT 1.0e-30)
        (e6 / (:CAT NN :LEX "untruth") :WEIGHT 1.3875405e-7))
```

Figure 14
The interlingual representation of the input in Figure 13 after expansion by the near-synonym choice module.

For each near-synonym NS in a cluster, a weight is computed by summing the degree to which the near-synonym satisfies each preference from the set P of input preferences:

$$\text{Weight}(\text{NS}, P) = \sum_{p \in P} \text{Sat}(p, \text{NS}). \tag{2}$$

The weights are transformed through an exponential function so that numbers are comparable with the differences of probabilities from HALogen's language model:

$$f(x) = \frac{e^{x^k}}{e - 1}. \tag{3}$$

We set $k = 15$ as a result of experiments with a development set.

For a given preference $p \in P$, the degree to which it is satisfied by NS is reduced to computing the similarity between each of NS's distinctions and a pseudodistinction $d(p)$ generated from p. The maximum value over i is taken (where $d_i(w)$ is the ith distinction of NS):

$$\text{Sat}(p, \text{NS}) = \max_i \ \text{Sim}(d(p), d_i(\text{NS})). \tag{4}$$

where the similarity of two distinctions, or of a distinction and a preference (transformed into a distinction), is computed with the three types of similarity measures that were used by Edmonds and Hirst (2002) in I-Saurus:

$$\mathrm{Sim}(d_1, d_2) = \begin{cases} \mathrm{Sim}_{\mathrm{den}}(d_1, d_2) & \text{if } d_1 \text{ and } d_2 \text{ are denotational distinctions} \\ \mathrm{Sim}_{\mathrm{att}}(d_1, d_2) & \text{if } d_1 \text{ and } d_2 \text{ are attitudinal distinctions} \\ \mathrm{Sim}_{\mathrm{sty}}(d_1, d_2) & \text{if } d_1 \text{ and } d_2 \text{ are stylistic distinctions} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Distinctions are formed out of several components, represented as symbolic values on certain dimensions, such as frequency (`seldom`, `sometimes`, etc.) and strength (`low`, `medium`, `high`). In order to compute a numeric score, each symbolic value is mapped into a numeric one. The numeric values are not as important as their relative difference. If the two distinctions are not of the same type, they are incommensurate and their similarity is zero. The formulas for $\mathrm{Sim}_{\mathrm{att}}$ and $\mathrm{Sim}_{\mathrm{sty}}$ involve relatively straightforward matching. However, $\mathrm{Sim}_{\mathrm{den}}$ requires the matching of complex interlingual structures. This boils down to computing the similarity between the main concepts of the two interlingual representations, and then recursively mapping the shared semantic roles (and compensating for the roles that appear in only one). When atomic concepts or words are reached, we use a simple measure of word/concept similarity based on the hierarchy of Sensus. All the details of these formulas, along with examples, are presented by Inkpen and Hirst (2003).

7.4 Integrating the Knowledge of Collocational Behavior

Knowledge of collocational behavior is not usually present in NLG systems. Adding it will increase the quality of the generated text, making it more idiomatic: The system will give priority to a near-synonym that produces a preferred collocation and will not choose one that causes an anti-collocation to appear in the generated sentence.

Unlike most other NLG systems, HALogen already incorporates some collocational knowledge implicitly encoded in its language model (bigrams or trigrams), but this is mainly knowledge of collocations between content words and function words. Therefore, in its integration into Xenon, the collocational knowledge acquired in Section 3 will be useful, as it includes collocations between near-synonyms and other nearby content words. Also, it is important whether the near-synonym occurs before or after the collocate; if both positions are possible, both collocations are in the knowledge base.

The architecture of Xenon extended with the near-synonym collocation module is presented in Figure 15. The near-synonym collocation module intercepts the forest structure, modifies its weights as necessary, and then forwards it to the statistical ranking module. If a potential anti-collocation is seen in the forest structure, the weight of the near-synonym is discounted by $W_{\mathrm{anti\_colloc}}$; if a less preferred collocation is seen, the weight of the near-synonym is discounted by $W_{\mathrm{less\_pref\_colloc}}$. For preferred collocations, the weight is unchanged. If the collocate is not the only alternative, the other alternatives should be discounted, unless they also form a preferred collocation. Section 7.5.2 explains how the discount weights were chosen.

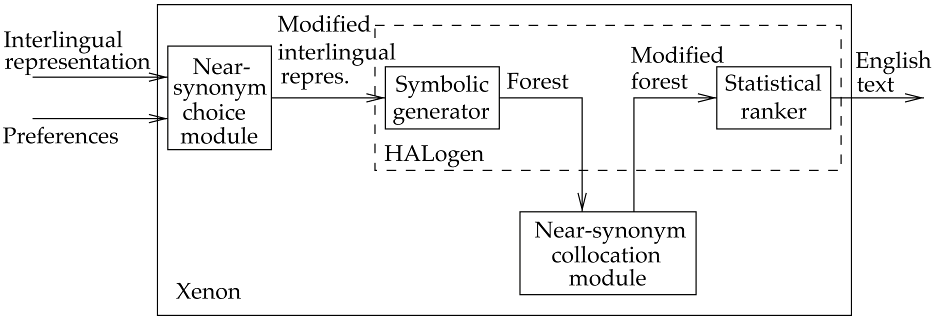

Interlingual representation
Preferences
Near-synonym choice module
Modified interlingual repres.
Symbolic generator
Forest
Modified forest
Statistical ranker
English text
HALogen
Near-synonym collocation module
Xenon

Figure 15
The architecture of Xenon extended with the near-synonym collocation module. In this figure, the knowledge sources are not shown.

7.5 Evaluation of Xenon

The components of Xenon to be evaluated here are the near-synonym choice module and the near-synonym collocation module. We evaluate each module in interaction with the sentence-realization module HALogen,[19] first individually and then both working together.[20]

An evaluation of HALogen itself was presented by Langkilde-Geary (2002a) using a section of the Penn Treebank as test set. HALogen was able to produce output for 80% of a set of 2,400 inputs (automatically derived from the test sentences by an input construction tool). The output was 94% correct when the input representation was fully specified, and between 94% and 55% for various other experimental settings. The accuracy was measured using the BLEU score (Papineni et al. 2001) and the string edit distance by comparing the generated sentences with the original sentences. This evaluation method can be considered as English-to-English translation via meaning representation.

7.5.1 Evaluation of the Near-Synonym Choice Module. For the evaluation of the near-synonym choice module, we conducted two experiments. (The collocation module was disabled for these experiments.) Experiment 1 involved simple monolingual generation. Xenon was given a suite of inputs: Each was an interlingual representation of a sentence and the set of nuances that correspond to a near-synonym in the sentence (see Figure 16). The sentence generated by Xenon was considered correct if the expected near-synonym, whose nuances were used as input preferences, is chosen. The sentences used in this first experiment were very simple; therefore, the interlingual representations were easily built by hand. In the interlingual representation, the near-synonym was replaced with the corresponding metaconcept. There was only one near-synonym in each sentence. Two data sets were used in Experiment 1: a development set of 32 near-synonyms of the five clusters presented in Figure 17 in order to set the exponent k of the scaling function in equation (3), and a test set of 43 near-synonyms selected from six clusters, namely, the set of English near-synonyms shown in Figure 18.

[19] All the evaluation experiments presented in this section used HALogen's trigram language model. The experiments were repeated with the bigram model, and the results were almost identical.

[20] Preliminary evaluation experiments of only the near-synonym choice module were presented by Inkpen and Hirst (2003).

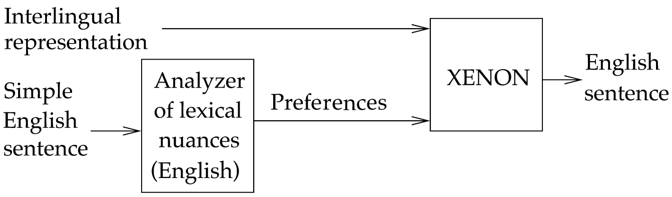

Interlingual representation
Simple English sentence
Analyzer of lexical nuances (English)
Preferences
XENON
English sentence

Figure 16
The architecture of Experiment 1.

1. benefit, advantage, favor, gain, profit
2. flow, gush, pour, run, spout, spurt, squirt, stream
3. deficient, inadequate, poor, unsatisfactory
4. afraid, aghast, alarmed, anxious, apprehensive, fearful, frightened, scared, terror-stricken
5. disapproval, animadversion, aspersion, blame, criticism, reprehension

Figure 17
Development data set used in Experiment 1.

Some of Xenon's choices could be correct solely because the expected near-synonym happens to be the default one (the one with the highest probability in the language model). So as a baseline (the performance that can be achieved without using the LKB of NS), we ran Xenon on all the test cases, but without input preferences.

The results of Experiment 1 are presented in Table 6. For each data set, the second column shows the number of test cases. The column labeled "Total correct" shows the number of answers considered correct (when the expected near-synonym was chosen). The column labeled "Ties" shows the number of cases when the expected near-synonym had weight 1.0, but there were other near-synonyms that also had weight 1.0 because they happen to have identical nuances in the LKB of NS. The same column shows in parentheses how many of these ties caused an incorrect near-synonym choice. In such cases, Xenon cannot be expected to make the correct choice, or, more precisely, the other choices are equally correct, at least as far as Xenon's LKB is concerned. Therefore, the

English: mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism
French: erreur, égarement, illusion, aberration, malentendu, mécompte, bévue, bêtise, blague, gaffe, boulette, brioche, maldonne, sophisme, lapsus, méprise, bourde

English: alcoholic, boozer, drunk, drunkard, lush, sot
French: ivrogne, alcoolique, intempérant, dipsomane, poivrot, pochard, sac à vin, soûlard, soûlographe, éthylique, boitout, imbriaque

English: leave, abandon, desert, forsake
French: abandonner, délaisser, déserter, lâcher, laisser tomber, planter là, plaquer, livrer, céder

English: opponent, adversary, antagonist, competitor, enemy, foe, rival
French: ennemi, adversaire, antagoniste, opposant, détracteur

English: thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy, wiry
French: mince, élancé, svelte, flandrin, grêle, fluet, effilé, fuselé, pincé

English: lie, falsehood, fib, prevarication, rationalization, untruth
French: mensonge, menterie, contrevérité, hâblerie, vanterie, fanfaronnade, craque

Figure 18
Test data set used in Experiment 1 (English only) and Experiment 2 (English and French).

Table 6
Results of Experiment 1 (boldface indicates best results).

Data set	No. of cases	Total correct	Correct by default	Ties	Base-line %	Accuracy (no ties) %	Accuracy %
Development	32	27	5	5 (4)	15.6	84.3	**96.4**
Test	43	35	6	10 (5)	13.9	81.3	**92.1**

accuracies computed without considering these cases (the seventh column) are underestimates of the real accuracy of Xenon. The last column presents accuracies taking the ties into account, defined as the number of correct answers divided by the difference between the total number of cases and the number of incorrectly resolved ties.

Experiment 2 is based on machine translation. These experiments measure how successful the translation of near-synonyms is, both from French into English and from English into English. The experiments used pairs of French and English sentences that are translations of one another (and that contain near-synonyms of interest), extracted from sentence-aligned parallel text, the bilingual Canadian Hansard. Examples are shown in Figure 19.[21] For each French sentence, Xenon should generate an English sentence that contains an English near-synonym that best matches the nuances of the French original. If Xenon chooses exactly the English near-synonym used in the parallel text, then Xenon's behavior is correct. This is a conservative evaluation measure because there are cases in which more than one of the possibilities would be acceptable.

As illustrated earlier in Figure 9, an analysis module is needed. For the evaluation experiments, a simplified analysis module is sufficient. Because the French and English sentences are translations of each other, we can assume that their interlingual representation is essentially the same. So for the purpose of these experiments, we can use the interlingual representation of the English sentence to approximate the interlingual representation of the French sentence and simply add the nuances of the French near-synonym to the representation. This is a simplification because there may be some sentences for which the interlingual representation of the French sentence is different because of translation divergences between languages (Dorr 1993). For the sentences in our test data, a quick manual inspection shows that this happens very rarely or not at all. This simplification eliminates the need to parse the French sentence and the need to build a tool to extract its semantics. As depicted in Figure 20, the interlingual representation is produced with a preexisting input construction tool that was previously used by Langkilde-Geary (2002a) in her HALogen evaluation experiments. In order to use this tool, we parsed the English sentences with Charniak's parser (Charniak 2000).[22] The tool was designed to work on parse trees from the Penn TreeBank, which have some extra annotations; it worked on parse trees produced by Charniak's parser, but it failed on some parse trees probably more often than it

[21] The sentences were obtained from USC/ISI (http://www.isi.edu/natural-language/download/hansard/) (approximately one million pairs of sentences). Other sources of parallel text, such as parallel translations of the Bible (http://benjamin.umd.edu/parallel/) (Resnik 1999) and a collection of Web pages (Resnik, Olsen, and Diab 1999), happened to contain very few occurrences of the near-synonyms of interest.

[22] ftp://ftp.cs.brown.edu/pub/nlparser/

⟨en⟩ Canada must send a clear message to international governments to **abandon** such atrocities if they wish to carry on cooperative relations with our country. ⟨/en⟩
⟨fr⟩ Le Canada doit faire savoir clairement aux gouvernements étrangers qu'ils doivent **abandonner** de telles pratiques atroces s'ils veulent entretenir des relations de coopération avec notre pays. ⟨/fr⟩

⟨en⟩ Populism is the natural **enemy** of representative democracy. ⟨/en⟩
⟨fr⟩ Le populisme est l'**ennemi** naturel de la démocratie représentative. ⟨/fr⟩

Figure 19
Examples of parallel sentences used in Experiment 2.

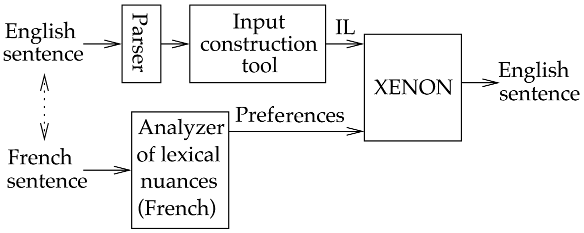


Figure 20
The architecture of Experiment 2 (French to English).

did in HALogen's evaluation experiments. We replaced each near-synonym with the metaconcept that is the core meaning of its cluster. The interlingual representation for the English sentence is semantically shallow; it does not reflect the meaning of the French sentence perfectly, but in these experiments we are interested only in the near-synonyms from the test set; therefore, the other words in the French sentence are not important.

The analyzer of French nuances of Figure 20 needs to extract nuances from an LKB of French synonyms. We created by hand an LKB for six clusters of French near-synonyms (those from Figure 18) from two paper dictionaries of French synonyms, Bénac (1956) and Bailly (1973). For each peripheral string, in French, an equivalent concept is found in Sensus by looking for English translations of the words and then finding Sensus concepts for the appropriate senses of the English words. Figure 21 presents a fragment of a cluster of French near-synonyms. For example, if we are told that *erreur* denotes *fausse opinion*, the equivalent peripheral concept is `(P8 (c8 / |view<belief| :mod |false>untrue|))`. If we are told that *gaffe* denotes *bêvue grossiere*, then the equivalent peripheral concept is `(P7 (c7 / |glaring,gross|))`.

A similar experiment, translating not from French into English but from English into English, is useful for evaluation purposes. An English sentence containing a near-synonym is processed to obtain its semantic representation (where the near-synonym is replaced with a metaconcept), and the lexical nuances of the near-synonym are input as preferences to Xenon. Ideally, the same near-synonym as in the original sentence would be chosen by Xenon (we consider it to be the correct choice). The percentage of times this happens is an evaluation measure. The architecture of this experiment is presented in Figure 22.

It happens that not all the near-synonyms in the test data set were found in Hansard—in fact, only 13 distinct pairs occur as translations of each other. Some of these pairs are very frequent, and some are rare. In order to evaluate the system for all these near-synonyms, both with and without regard to their frequency, we prepared two

```
(defcluster generic_erreur_n
    :syns (erreur egarement illusion aberration malentendu mecompte
           bevue betise blague gaffe boulette brioche maldonne sophisme
           lapsus meprise bourde)
:periph ((P1 (c1 / |take amiss| :object thing))
          ...
         (P7 (c7 / |glaring,gross|))
         (P8 (c8 / |view<belief| :mod |false>untrue|))
          ... )
:distinctions ( ...
               (meprise usually medium Denotation P1)
               ; "prend une chose pour une autre"
               (gaffe usually medium Denotation P7)
               ; "bevue grossiere"
               (erreur usually medium Denotation P8)
               ; "fausse opinion"
              )
)
```

Figure 21
Fragment of a cluster of French near-synonyms.

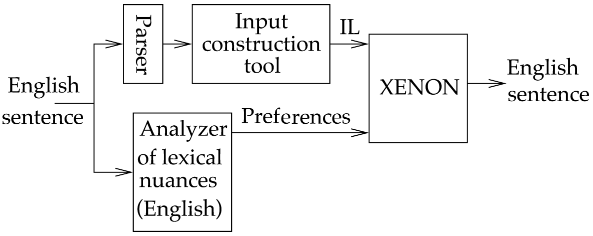


Figure 22
The architecture of Experiment 2 (English to English).

data sets by sampling Hansard in two different ways. Sentence data set 1 contains, for each French and English near-synonym pair in the test data set, two pairs of sentences in which the English word appears as a translation of the French. The sentences selected for each pair were the first two for which the input construction tool produced a valid interlingual representation. Sentence data set 2 is similar to set 1, but instead of having two sentences for a near-synonym pair, it contains all the sentence pairs in a large fragment of Hansard in which the near-synonyms of interest occurred. Therefore, this data set has the advantage of a natural frequency distribution. It has the disadvantage that the results for the less-frequent near-synonyms, which tend to be the "harder" and more-interesting cases (see below), may be swamped by the more-frequent, relatively "easy" cases. Initially there were 564 pairs of sentences, but the input construction tool worked successfully only on 298 English sentences. The interlingual representations that it produced are quite complex, typically several pages long.

The results of Experiment 2 are presented in Table 7;[23] the interpretation of the columns is the same as for Table 6. For each set of sentences, the baseline is the same

[23] The improvement over baseline is statistically significant for all the results in Table 7, except the third line.

Table 7
Results of Experiment 2 (boldface indicates best results).

Data set and condition	No. of cases	Total correct	Correct by default	Ties	Base-line %	Accuracy (no ties) %	Accuracy %
Sentence set 1 French to English	26	13	10	5 (3)	38.4	50.0	**56.5**
Sentence set 1 English to English	26	26	10	2 (0)	38.4	100	**100**
Sentence set 2 French to English	298	217	214	7 (1)	71.8	72.8	**73.0**
Sentence set 2 English to English	298	296	214	2 (0)	71.8	99.3	**99.3**

for the French-to-English and English-to-English experiments because no nuances were used as input for the baseline experiments. The baseline for data set 2 is quite high (71.8%), because it contains sentences with frequent near-synonyms, which happen to be the ones that Xenon chooses by default in the absence of input preferences. Xenon's performance is well above baseline, with the exception of the French-to-English condition on sentence data set 2.

In the English-to-English experiments, there are two reasons to expect Xenon's accuracy to be less than 100% even if the input nuances are the nuances of a particular English near-synonym. The first reason is that there are cases in which two or more near-synonyms get an equal, maximal score because they do not have nuances that differentiate them (either they are perfectly interchangeable, or the LKB of NS does not contain enough knowledge) and the one chosen is not the expected one. The second reason is that sometimes Xenon does not choose the expected near-synonym even if it is the only one with maximal weight. This may happen because HALogen makes the final choice by combining the weight received from the near-synonym choice module with the probabilities from the language model that is part of HALogen. Frequent words may have high probabilities in the language model. If the expected near-synonym is very rare, or maybe was not seen at all by the language model, its probability is very low; yet it is exactly those cases where a writer chooses a rare near-synonym over a more-frequent alternative that the choice is the most telling. When combining the weights with the probabilities, a frequent near-synonym may win even if it has a lower weight assigned by the near-synonym choice module. In such cases, the default near-synonym (the most frequent of the cluster) wins. Sometimes such behavior is justified because there may be other constraints that influence HALogen's choice.

In the French-to-English experiments, the performance of Xenon is lower than in the English-to-English experiments. There are two explanations. First, there is some overlap between the nuances of the French and the English near-synonyms, but less than one would expect. For example, the English adjective *alcoholic* is close in nuances to the French adjective *alcoolique*, but they have no nuance in common in Xenon's knowledge bases simply because of the incompleteness of the explanations given by lexicographers in the dictionary entries.

The second explanation is related to what is considered the "correct" choice of near-synonym. Sometimes more than one translation of a French near-synonym could be correct, but in this conservative evaluation, the solution is the near-synonym that was used in the equivalent English sentence. Therefore, test cases that would be considered correct by a human judge are harshly penalized. Moreover, the near-synonym choice module always chooses the same translation for a near-synonym, even if the near-synonym is translated in Hansard in more than one way, because Xenon does not consider the context of the near-synonym in the sentence. (The context is taken into account only when the collocation module is enabled and a preferred collocation is detected in the sentences.) For example, the French noun *erreur* is translated in Hansard sometimes as *error*, sometimes as *mistake*. Both have nuances in common with *erreur*, but *mistake* happened to have higher overlap with *erreur* than *error*; as a result, the near-synonym choice module always chooses *mistake* (except when the collocation module is enabled and finds a preferred collocation such as *administrative error*). All the cases in which *error* was used as translation of *erreur* in Hansard are penalized as incorrect in the evaluation of the near-synonym choice module. A few of these cases could be indeed incorrect, but probably many of them would be considered correct by a human judge.

Another way to look at the performance of Xenon is to measure how many times it makes appropriate choices that cannot be made by HALogen—that is, cases that make good use of the nuances from the LKB of NS. This excludes the test cases with default near-synonyms—those in which Xenon makes the right choice simply because of its language model—and cases of ties in which Xenon cannot make the expected choice. Accuracies for nondefault cases vary from 84.3% to 100%.

7.5.2 Evaluation of the Near-Synonym Collocation Module. For the evaluation of the near-synonym collocation module, we collected sentences from the BNC that contain preferred collocations from the knowledge base of near-synonym collocational behavior. The BNC was preferred over Hansard for these evaluation experiments because it is a balanced corpus and contains the collocations of interest, whereas Hansard does not contain some of the collocations and near-synonyms of interest. The sentences were collected from the first half of the BNC (50 million words). Sentence data sets 3 and 4 contain collocations for the development set of near-synonyms in Figure 17; sentence data sets 5 and 6 contain collocations for the English near-synonyms in Figure 18. Sets 3 and 5 include at most two sentences per collocation (the first two sentences from the corpus, except in cases when the input construction tool failed to produce valid interlingual representations); sets 4 and 6 include all the sentences with collocations as they occurred in the fragment of the corpus (except the sentences for which the input construction tool failed). For example, for set 4 there were initially 527 sentences, and the input construction tool succeeded on 297 of them. Set 3 was used for development—to choose the discount weights (see below)—and the others only for testing. The architecture of this experiment is the same as that of the English-to-English experiments (Figure 22), except that in this case it was the near-synonym choice module that was disabled.

We observe that the sentence data sets may contain collocations for the wrong senses of some near-synonyms because, as explained in Section 3.4, the near-synonym collocations knowledge base may contain, for a cluster, collocations for a different sense. For example, the collocation *trains run* appears in the cluster *flow, gush, pour, run, spout, spurt, squirt, stream*, when it should appear only in another cluster. In this case the near-synonym *run* should not be replaced with the metaconcept `generic_flow_v` because it corresponds to a different metaconcept. These sentences should be eliminated from

Table 8
The results of the evaluation of Xenon's collocations module (boldface indicates best results).

		HALogen only (baseline)			HALogen + collocations		
Sentence data set	No. of cases	Correct NS choice	Pref. collocs	Anti-collocs	Correct NS choice	Pref. collocs	Anti-collocs
Set 3	105	62%	88%	12%	70%	100%	**0%**
Set 4	297	83%	91%	9%	87%	99%	**1%**
Set 5	44	59%	80%	20%	75%	100%	**0%**
Set 6	185	58%	68%	32%	86%	99%	**1%**

the data sets, but this would involve disambiguation or manual elimination. However, they do not affect the evaluation results because they are unlikely to produce anti-collocations. This is because *trains run* is a frequent bigram, whereas *trains flow* is not; Xenon will make the correct choice by default.

Sentence data set 3 was used to choose the best values of the discount weights $W_{\text{anti\_colloc}}$ and $W_{\text{less\_pref\_colloc}}$. In fact, the latter could be approximated by the former, treating less preferred collocations as anti-collocations, because the number of less preferred collocations is very small in the knowledge base. As the value of the discount weight $W_{\text{anti\_colloc}}$ increased (from 0.0 and 1.0), the number of anti-collocations generated decreased; there were no anti-collocations left for $W_{\text{anti\_colloc}} = 0.995$.

Table 8 presents the results of the evaluation experiments. These results refer to the evaluation of Xenon with the near-synonym collocations module enabled and the near-synonym choice module disabled (lexical nuances are ignored in this experiment). The baseline used for comparison is obtained by running HALogen only, without any extension modules (no knowledge of collocations). For each test, the first four columns contain the number of test cases, the number of near-synonyms correctly chosen by the baseline system, the number of preferred collocations, and the number of anti-collocations produced by the baseline system. The remainder of the columns present results obtained by running Xenon with only the near-synonym collocations module enabled (i.e., HALogen and the collocations module): the number of near-synonyms correctly chosen, the number of preferred collocations produced, and the number of anti-collocations produced. The number of anti-collocations was successfully reduced to zero, except for sentence sets 4 and 6 where 1% of the anti-collocations remained. The sixth column (correct choices or accuracy) differs from the seventh column (preferred collocations) in the following way: The correct choice is the near-synonym used in the original BNC sentence; sometimes the generated sentence can choose a different near-synonym that is not the expected one but which participates in a preferred collocation (this happens when more than one near-synonym from the same cluster collocates well with the collocate word). For example, both *serious mistake* and *serious blunder* are preferred collocations, while only one of *mistake* and *blunder* is the correct choice in any particular context. The number of correct choices is relevant in this experiment only to show that the collocations module does not have a negative effect on correctness; it even increases the accuracy.[24]

[24] The accuracy without ties was used here; therefore the numbers are conservative.

7.5.3 Evaluation of the Two Modules in Interaction. The interaction between the near-synonym choice module and the collocations module increases Xenon's performance. To prove this, we repeated the experiments of the previous section, but this time with input preferences (the nuances of the near-synonym from the original sentence). The architecture of this test is the same as that of the English-to-English experiments in Section 7.5.1, depicted in Figure 22. Table 9 shows the number of correct near-synonym choices (and the percent accuracy) for the baseline case (no nuances, no collocation module; i.e., HALogen by itself), for the collocations module alone (i.e., HALogen and the collocations module only; this column is also part of Table 8), for the near-synonym choice module alone (i.e., HALogen and the nuances module only), and for Xenon with both modules enabled. When both modules are enabled there is a slight increase in accuracy on sentence data sets 4, 5, and 6; the accuracy on set 3 is the same as using the near-synonyms module only.

7.6 Summary

This section presented Xenon, an NLG system capable of choosing the near-synonym that best satisfies a set of input preferences (lexical nuances). The input preferences could come from an analysis module for a different language; in this case the translation into English would preserve not only the meaning but also nuances of meaning. The evaluation of Xenon's two new modules shows that they behave well, both independently and in interaction.

The evaluation showed that we were successful in dealing with lexical nuances in general. One weak point of the evaluation was the relatively small overlap in coverage of the French and English knowledge bases. Another bottle-neck was the need for a language-neutral ontology.

8. Conclusion

We have presented a method for extracting knowledge from dictionaries of near-synonym discrimination. The method can potentially be applied to any dictionary

Table 9
Correct near-synonym choices for the baseline system (HALogen only), for HALogen with each module of Xenon separately, and for HALogen with both modules of Xenon (boldface indicates best results).

		HALogen (baseline)		Xenon	
Sentence data set	Number of cases	Correct NS	Correct NS collocations module	Correct NS nuances module	Correct NS nuances + collocations
Set 3	105	62%	70%	93%	**93%**
Set 4	297	83%	87%	95%	**95%**
Set 5	44	59%	75%	93%	**95%**
Set 6	185	58%	86%	91%	**95%**

of near-synonym discrimination for any language for which preprocessing tools, such as part-of-speech taggers and parsers, are available. We built a new lexical resource, a lexical knowledge base of differences among English near-synonyms, by applying the extraction method to *Choose the Right Word*. The precision and recall of the extracted knowledge was estimated to be in the range of 70–80%. If higher precision and recall are needed for particular applications, a human could validate each extraction step. We enriched the initial lexical knowledge base of near-synonyms with distinctions extracted from machine-readable dictionaries.

We have presented Xenon, a natural language generation system that uses the LKB of NS to choose the near-synonym that best matches a set of input preferences. Xenon extends a previous NLG system with two new modules: a module that chooses near-synonyms on the basis of their lexical nuances, and a module that chooses near-synonyms on the basis of their collocations. To evaluate Xenon, we manually built a small LKB of French near-synonyms. The test set consisted of English and French sentences that are mutual translations. An interlingual representation (with the near-synonym replaced by the core denotation of its cluster) was input to Xenon, together with the nuances of the near-synonym from the French sentence. The generated sentence was considered correct if the chosen English near-synonym was the one from the original English sentence. We also evaluated the near-synonym collocation module and the interaction of the two modules.

Short-term future work includes overcoming some limitations and extending the current work, such as extending the near-synonym representation with other types of distinctions such as information about more general and more specific words, and information about special meanings that some words have in particular contexts or domains (e.g., in a legal context).

Longer-term future work directions include the following:

Analysis of lexical nuances A full-fledged analysis module could be developed. Sense disambiguation would be required when a near-synonym is a member of more than one cluster. It is more difficult to model the influence of the context and the complex interaction of the lexical nuances. Such an analysis module could be used in an Machine Translation (MT) system that preserves lexical nuances. It could also be used to determine nuances of text for different purposes. For example, a system could decide if a text is positive, neutral, or negative in its semantic orientation. Then Xenon could be used to generate a new text that has the same meaning as the original text but a different semantic orientation. This could be useful, for example, in an application that sends letters to customers: If the initial draft of the text is found to be too negative, it could be transformed into a more positive text before it is sent to the customer.

Lexical and conceptual associations The method presented in Section 3 could be extended to acquire *lexical associations* (i.e., longer-distance collocations) of near-synonyms. Words that strongly associate with the near-synonyms can be useful, especially those that associate with only one of the near-synonyms in the cluster. These strong associations could provide additional knowledge about nuances of near-synonyms.

An experiment similar to that presented in Section 3 could look for words that co-occur in a window of size $K > 2$ to acquire lexical associations, which would include the collocations extracted in Section 3.2. Church et al. (1991) presented associations for the near-synonyms *ship* and *boat*; they suggest that a lexicographer looking at these associations can infer that *boats* are generally smaller than *ships* because they are found in *rivers* and *lakes* and are used for small jobs (e.g., *fishing*, *police*, *pleasure*), whereas *ships* are found in *seas* and are used for serious business (e.g., *cargo*, *war*). It could be possible to automatically infer this kind of knowledge or to validate already acquired knowledge.

Words that do not associate with a near-synonym but associate with all the other near-synonyms in a cluster can tell us something about its nuances of meaning. For example, *terrible slip* is an anti-association, whereas *terrible* associates with *mistake, blunder, error*. This is an indication that *slip* is a minor error. By further generalization, the associations could become conceptual associations. This may allow the automatic learning of denotational distinctions between near-synonyms from free text. The concepts that are common to all the near-synonyms in a cluster could be part of the core denotation, whereas those that associate only with one near-synonym could be part of a distinction.

Cross-lingual lexical nuances The method presented in Section 2 could be used to automatically build a lexical knowledge base of near-synonym differences for other languages, such as French, for which dictionaries of synonym discriminations are available (in paper form) along with other resources, such as part-of-speech taggers and parsers. In order to use the French and the English knowledge bases in the same system, a study of the cross-lingual lexical nuances will be needed.

Analysis of types of peripheral nuances Linguists and lexicographers have looked at differences between particular types of near-synonyms. For example, Gao (2001) studied the semantic distinctions between Chinese physical action verbs; one type of distinctive peripheral nuance is the manner in which the movement is made for each verb. This kind of study could help to develop a list of the main types of peripheral nuances (peripheral concepts). In our work, the form that the peripheral nuances can take is not restricted, because the list of peripheral nuances is open-ended. However, it may be possible to keep the form unrestricted but add restrictions for the most important types of peripheral nuances.

Intelligent thesaurus The acquired lexical knowledge base of near-synonym differences could be used to develop an intelligent thesaurus that assists a writer not only with a list of words that are similar to a given word but also with explanations about the differences in nuances of meaning between the possible choices. The intelligent thesaurus could order the choices to suit a particular writing context. The knowledge about the collocational behavior of near-synonyms can be used in determining the order: Near-synonyms that produce anti-collocations would be ranked lower than near-synonyms that produce preferred collocations.

Automatic acquisition of near-synonyms This work considered only the near-synonyms and distinctions that were listed by the lexicographers of CTRW. Other dictionaries of synonym discrimination may have slightly different views. Merging clusters from different dictionaries is possible. Also, near-synonym clusters could be acquired from free text. This would distinguish near-synonyms from the pool of related words. As mentioned in Section 5.3, Lin et al. (2003) acquired words that are related by contextual similarity, and then filtered out the antonyms. Words that are related by relations other than near-synonymy could also be filtered out. One way to do this could be to collect signatures for each potential near-synonym—words that associate with it in many contexts. For two candidate words, if one signature is contained in the other, the words are probably in an IS-A relation; if the signatures overlap totally, it is a true near-synonymy relation; if the signatures overlap partially, it is a different kind of relation. The acquisition of more near-synonyms, followed by the acquisition of more distinctions, is needed to increase the coverage of our lexical knowledge base of near-synonym differences.

Acknowledgments

We thank Irene Langkilde-Geary for making the input construction tool available and for her advice on how to connect our preference satisfaction mechanism to HALogen. We thank Phil Edmonds for making available the source code of I-Saurus. We thank Suzanne Stevenson, Gerald Penn, Ted Pedersen, and anonymous reviewers for their feedback on various stages of this work. We thank Ol'ga Feiguina for being an enthusiastic research assistant and for implementing the programs from Section 4. Our work is financially supported by the Natural Sciences and Engineering Research Council of Canada, the University of Toronto, and the University of Ottawa.

References

Abney, Steven. 1996. Partial parsing via finite-state cascades. In *Proceedings of the 8th European Summer School in Logic, Language and Information (ESS-LLI'96), Robust Parsing Workshop*, pages 124–131, Prague, Czech Republic.

Apresjan, J. D., V. V. Botiakova, T. E. Latiskeva, M. A. Mosiagina, I. V. Polik, V. I. Rakitina, A. A. Rozenman, and E. E. Sretenskaia. 1980. *Anglo-Russkii Sinonimicheskii Slovar*. Izdatelstvo Russkii Jazik.

Apresjan, Juri. 2000. *Systematic Lexicography*. Translation, Oxford University Press.

Bailly, René, editor. 1973. *Dictionnaire des Synonymes de la Langue Française*. Larousse, Paris.

Barrière, Caroline and Fred Popowich. 1996. Concept clustering and knowledge integration from a children's dictionary. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'96)*, pages 65–70, Copenhagen, Denmark.

Bénac, Henri, editor. 1956. *Dictionnaire des Synonymes*. Librarie Hachette, Paris.

Bernard, J. R. L., editor. 1986. *The Macquarie Thesaurus—Companion to The Macquarie Dictionary*. Macquarie Library, Sydney, Australia.

Caraballo, Sharon. 1999. Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126, College Park, MD.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing (NAACL-ANLP 2000)*, pages 132–139, Seattle, WA.

Church, Kenneth, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using statistics in lexical analysis. In Uri Zernik, editor, *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, pages 115–164. Lawrence Erlbaum.

Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz.

Collins, Michael and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 100–110, College Park, MD.

Delbridge, A., J. R. L. Bernard, D. Blair, W. S. Ramson, and Susan Butler, editors. 1987. *The Macquarie Dictionary*. Macquarie Library, Sydney, Australia.

DiMarco, Chrysanne, Graeme Hirst, and Manfred Stede. 1993. The semantic and stylistic differentiation of synonyms and near-synonyms. In *Proceedings of AAAI Spring Symposium on Building Lexicons for Machine Translation*, pages 114–121, Stanford, CA.

Dorr, Bonnie J. 1993. *The Use of Lexical Semantics in Interlingual Machine Translation*. The MIT Press, Cambridge, MA.

Edmonds, Philip. 1999. *Semantic representations of near-synonyms for automatic lexical choice*. Ph.D. thesis, University of Toronto.

Edmonds, Philip and Graeme Hirst. 2002. Near-synonymy and lexical choice. *Computational Linguistics*, 28(2):105–145.

Gao, Hong. 2001. *The Physical Foundation of the Patterning of Physical Action Verbs*. Ph.D. thesis, Lund University.

Gove, Philip B., editor. 1984. *Webster's New Dictionary of Synonyms*. G.&C. Merriam Co.

Hayakawa, S. I., editor. 1994. *Choose the Right Word*. Second Edition, revised by Eugene Ehrlich. HarperCollins Publishers.

Hearst, Marti. 1992. Automatic acquisition of hyponyms from large corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 538–545, Nantes, France.

Hirst, Graeme. 1995. Near-synonymy and the structure of lexical knowledge. In *Working Notes, AAAI Symposium on Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, pages 51–56, Stanford University.

Hovy, Eduard. 1990. Pragmatics and language generation. *Artificial Intelligence*, 43:153–197.

Ide, Nancy and Jean Véronis. 1994. Knowledge extraction from machine-readable dictionaries: An evaluation. In P. Steffens, editor, *Machine Translation and the Lexicon*, pages 19–34, Springer-Verlag.

Inkpen, Diana. 2003. *Building a Lexical Knowledge Base of Near-Synonym Differences*. Ph.D. thesis, University of Toronto.

Inkpen, Diana Zaiu and Graeme Hirst. 2001. Building a lexical knowledge base of near-synonym differences. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 47–52, Pittsburgh, PA.

Inkpen, Diana Zaiu and Graeme Hirst. 2002. Acquiring collocations for lexical choice between near-synonyms. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition, 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 67–76, Philadelphia, PA.

Inkpen, Diana Zaiu and Graeme Hirst. 2003. Near-synonym choice in natural language generation. In *Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing)*, pages 204–211, Borovets, Bulgaria.

Knight, Kevin and Steve Luk. 1994. Building a large knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 773–778, Seattle, WA.

Langkilde, Irene. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing (NAACL-ANLP 2000)*, pages 170–177, Seattle, WA.

Langkilde, Irene and Kevin Knight. 1998. The practical value of N-grams in generation. In *Proceedings of the 9th International Natural Language Generation Workshop*, pages 248–255, Niagara-on-the-Lake, Canada.

Langkilde-Geary, Irene. 2002a. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 12th International Natural Language Generation Workshop*, pages 17–24, New York, NY.

Langkilde-Geary, Irene. 2002b. *A Foundation for a General-Purpose Natural Language Generation: Sentence Realization Using Probabilistic Models of Language*. Ph.D. thesis, University of Southern California.

Lenat, Doug. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lesk, Michael. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC Conference*, pages 24–26, Toronto, Canada.

Lin, Dekang, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proceedings of the Eighteenth Joint International Conference on Artificial Intelligence (IJCAI-03)*, pages 1492–1493, Acapulco, Mexico.

Lindberg, Donald A. B., Betsy L. Humphreys, and Alexa T. McCray. 1993. The unified medical language system. *Methods of Information in Medicine*, 32(4):281–289.

Mahesh, Kavi and Sergei Nirenburg. 1995. A situated ontology for practical NLP. In *Proceedings of Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada.

Manning, Christopher and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

McKeown, Kathleen and Dragomir Radev. 2000. Collocations. In Robert Dale, Hermann Moisl, and Harold Somers, editors, *Handbook of Natural Language Processing*, pages 507–524. Marcel Dekker.

Miller, George A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. Technical report RC22176, IBM Research Division, Thomas J. Watson Research Center.

Pearce, Darren. 2001. Synonymy in collocation extraction. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second meeting of the North*

American Chapter of the Association for Computational Linguistics, pages 41–46, Pittsburgh, PA.

Pedersen, Ted and Satanjeev Banerjee. 2003. The design, implementation, and use of the ngram statistical package. In *Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2003)*, pages 370–381, Mexico City, Mexico.

Ploux, Sabine and Hyungsuk Ji. 2003. A model for matching semantic maps between languages (French/English, English/French). *Computational Linguistics*, 29(2):155–178.

Resnik, Philip. 1999. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, College Park, MD.

Resnik, Philip, Mari Broman Olsen, and Mona Diab. 1999. The bible as a parallel corpus: Annotating the 'Book of 2000 Tongues'. *Computers and the Humanities*, 33(1–2):129–153.

Richardson, Stephen, William Dolan, and Lucy Vanderwende. 1998. MindNet: Acquiring and structuring semantic information from text. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics joint with 17th International Conference on Computational Linguistics (ACL-COLING'98)*, pages 1098–1102, Montreal, Quebec, Canada.

Riloff, Ellen and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, FL.

Roget, Peter Mark, editor. 1852. *Roget's Thesaurus of English Words and Phrases*. Longman Group Ltd., Harlow, Essex, England.

Room, Adrian, editor. 1981. *Room's Dictionary of Distinguishables*. Routhledge & Kegan Paul, Boston.

Sim, Julius and Chris C. Wright. 2005. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. *Physical Therapy*, 85(3):257–268.

Stone, Philip J., Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and associates. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press, Cambridge, MA.

Turney, Peter. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML 2001)*, pages 491–502, Freiburg, Germany.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.