Building a Lexical Knowledge-Base of Near-Synonym Differences

Diana Zaiu Inkpen and **Graeme Hirst**
Department of Computer Science,
University of Toronto
Toronto, Ontario, Canada M5S 3G4
{dianaz,gh}@cs.toronto.edu

Abstract

In machine translation and natural language generation, making a poor choice from a set of near-synonyms can be imprecise or awkward, or convey unwanted implications. Our goal is to automatically derive a lexical knowledge-base from a dictionary of near-synonym discriminations. We do this by classifying sentences according to the classes of distinctions they express, on the basis of words selected by a decision-list algorithm. Improvements on previous results are due in part to the addition of a coreference module.

1 Near-synonyms

Near-synonyms are words that are almost synonyms, but not quite. They are not fully inter-substitutable, but rather 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.

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* (Hayakawa, 1994), which list clusters of similar words and explicate the differences between the words in each cluster. As a matter of terminology, we use the word *cluster* to denote the set of near-synonyms in a dictionary entry, plus the differences among the near-synonyms of that entry. These dictionaries are in effect dictionaries of near-synonym discriminations. 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.

Near-synonyms are important for fine-grained distinctions in MT systems. For example, when translating the French word *erreur* to English, one of the near-synonyms *error, mistake, blunder, blooper, contretemps, goof, slip, solecism* could be chosen, depending on the context and on the nuances that need to be conveyed.

DiMarco and Hirst (1993) analyzed the type of differences adduced in dictionaries of near-synonym discriminations. They found that only a limited number of types were used, making it possible to formalize the entries in a computational form. Edmonds (1999) designed a model to represent near-synonyms, and he constructed by hand the representations for nine clusters.

Our goal is to automatically derive a lexical knowledge base (LKB) of near-synonyms from a dictionary of near-synonym discriminations. We present our results for the extraction of knowledge from the text of the dictionary, and sketch our approach for the next step of dealing with the concepts in the representations and a reorganization of an existing ontology. Our goal is not only to automatically extract knowledge from one dictionary of synonym discriminations, but also to discover a general methodology which can be applied to any such dictionary with minimal adaptation.

2 Edmonds's model of lexical knowledge

Edmonds (1999) and Edmonds and Hirst (2000) 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. Such a model cannot account for indirect expressions of meaning or for fuzzy differences between near-synonyms.

Edmonds (1999) modifies 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. Thus the meaning of a word consists of both a core sense that allows 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*. Each

```
(defcluster error_C
 :syns (error_l mistake_l blunder_l slip_l
        lapse_l howler_l)
 :core (ROOT Generic-Error)
 :periph ((P1 Stupidity) (P2 Blameworthiness)
          (P3 Criticism (ATTRIBUTE (P3-1 Severity)))
          (P4 Misconception) (P5 Accident)
          (P6 Inattention))
 :distinctions
 ((blunder_l usually medium implication P1)
  (mistake_l sometimes medium implication
                                (P2 (DEGREE 'medium)))
  (blunder_l sometimes medium implication
                                  (P2 (DEGREE 'high)))
  (mistake_l always medium implication
                                 (P3-1 (DEGREE 'low)))
  (error_l always medium implication
                              (P3-1 (DEGREE 'medium)))
  (blunder_l always medium implication
                                (P3-1 (DEGREE 'high)))
  (mistake_l always medium implication P4)
  (slip_l always medium implication P5)
  (mistake_l always low implication P5)
  (lapse_l always low implication P5)
  (lapse_l always medium implication P6)
  (blunder_l always medium pejorative)
  (blunder_l high concreteness)
  (error_l low concreteness) (howler_l low formality)
  (mistake_l low concreteness)))
```

Figure 1: Edmonds's representation for the cluster *error*, *mistake*, *blunder*, *slip*, *lapse*, *howler*.

cluster has a core denotation that represents the essential shared denotational meaning of its near-synonyms. The internal structure of each cluster is complex, representing semantic (or denotational), stylistic, and expressive (or attitudinal) differences between 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). For example, the structure for the near-synonyms of the word *error*, built by hand by Edmonds (1999), is shown in Figure 1.

In this model, a cluster includes the following fields: `syns` – a list of near-synonyms in the cluster; `core` – the core denotation, or essential shared meaning of the near-synonyms in the cluster, represented as a configuration of concepts; `periph` – a set of peripheral concepts that extend the core denotation, and pertain to the differentiation of the near-synonyms; and `distinctions` – the actual distinctions between near-synonyms.

Building such representations by hand is difficult and time-consuming, and Edmonds completed only nine of them. Our goal is to automatically extract the content of all the entries in a dictionary of near-synonym discriminations, using a slightly simplified form of Edmonds's representation for the content of a cluster. We hypothesize that the language of the entries is sufficiently regular to allow automatic extraction of knowledge from them. The dictionary of near-synonym differences that we use is *Choose the Right Word* (Hayakawa, 1994) (hereafter CTRW). An example of text from this dictionary is presented in Figure 2. After OCR scanning of CTRW and error correction, we have marked up the structure of the dictionary in XML.

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.]

Figure 2: Part of an entry in CTRW. Copyright ©1987. Reprinted by arrangement with HarperCollins Publishers, Inc.

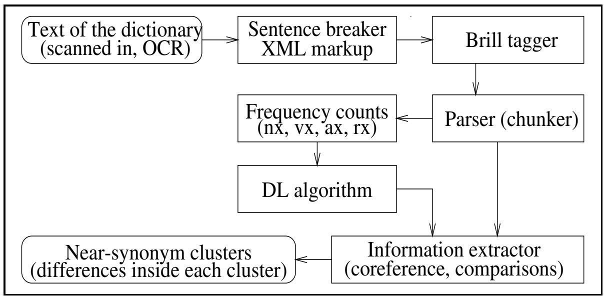

Figure 3: The architecture of the extraction system.

Figure 3 presents the architecture of the extraction module, which is described in the next sections. The extraction component obtains the relevant information from each sentence and produces the initial clusters, containing the peripheral concepts as simple strings. Some of our preliminary results were presented in (Inkpen and Hirst, 2001). The results we present here are improved, the extraction component has been revised, and a coreference resolution module specific to CTRW has been added.

3 Distinctions among near-synonyms

¿From each sentence of the dictionary, the program needs to extract the information relevant to the representation. Following Edmonds's analysis of the distinctions among near-synonyms, we derived the class hierarchy of distinctions presented in Figure 4. The top-level class DISTINCTIONS consists of DENOTATIONAL DISTINCTIONS, ATTITUDE, and STYLE. The last two are grouped together in a single class, ATTITUDE-STYLE DISTINCTIONS, be-

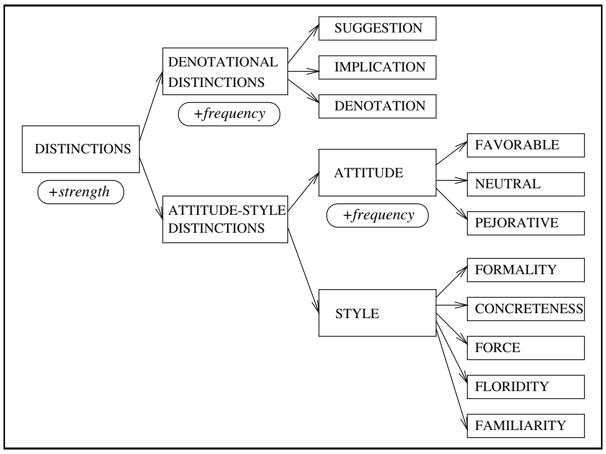

Figure 4: The class hierarchy of distinctions. Rectangles represent classes; ovals represent an attribute that a class and its descendents have.

cause they present similar behavior from the point of view of this research.

3.1 Denotational distinctions

Near-synonyms can differ in the frequency with which they express a component of their meaning (e.g., <u>Occasionally</u>, ***invasion*** *suggests a large-scale but unplanned* ***incursion***), in the indirectness of the expression of the component (e.g., ***Test*** <u>*strongly implies*</u> *an actual application of these means*), and in fine-grained variations of the idea itself (e.g., ***Paternalistic*** *may suggest* <u>*either*</u> *benevolent rule* <u>*or*</u> *a style of government determined to keep the governed helpless and dependent*).

For *denotational distinctions*, the tuples to be extracted have the form ⟨`near-synonym`, `frequency`, `strength`, `indirectness`, `peripheral-concept`⟩. The `indirectness` takes the values `suggestion`, `denotation`, `implication`. It is signaled by many words in CTRW, including *suggests*, *denotes*, *implies*, and *connotes*. `Strength` takes the values `low`, `medium`, `high`, and it is signaled by words such as *strongly* and *weakly*. `Frequency` takes the values `always`, `usually`, `sometimes`, `seldom`, `never` and is signaled by the corresponding English words. Default values are used when `strength` and `frequency` are not specified.

3.2 Attitudinal distinctions

A word can convey different attitudes of the speaker towards an entity of the situation. The three attitudes represented in the model are `pejorative`, `neutral`, and `favorable`. An example of a sentence in CTRW expressing attitudes is: ***Blurb*** *is also used* <u>*pejoratively*</u> *to denote the extravagant and insincere praise common in such writing*. This contains information about the pejorative attitude, in addition to its information about denotational distinctions.

The information extracted for attitudinal distinctions has the form ⟨`near-synonym`, `frequency`, `strength`, `attitude`⟩, where `strength` and `frequency` have the same values and significance as in the previous section.

3.3 Stylistic distinctions

The information extracted from CTRW about stylistic variations has the form ⟨`near-synonym`, `strength`, `stylistic-feature`⟩, where the `stylistic feature` has the values `formality`, `force`, `concreteness`, `floridity`, and `familiarity` (Hovy, 1990). The `strength` has the values `low`, `medium`, `high`, indicating the level of the stylistic attribute. 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*.

4 The decision-list learning algorithm

In order to automatically create near-synonym representations, the program needs to extract relevant portions of the text that are informative about these attributes. Therefore, the goal is to learn for each leaf class in the hierarchy a set of words or expressions in CTRW that characterizes descriptions of the class. When classifying a sentence (or fragment of sentence) the program has to decide which leaf class it expresses, and also with what `strength` and what `frequency`. We use a decision-list algorithm to learn sets of words and patterns for the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS.

Our decision-list (DL) algorithm (Figure 5) is tailored for extraction from CTRW. Like that of Collins and Singer (1999), our program learns two kinds of rules: main rules (for words that are significant for distinction classes) and auxiliary rules (for frequency words, strength words, and comparison words). We also extract patterns and relevant words for the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS, similar to the domain-specific lexicon extraction of Riloff and Jones (1999).

In order to obtain input data, we replace all the near-synonyms in the text of the dictionary with the term *NS*; 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 a threshold t times (where $t = 3$ in our experiments). (We prefer to use a chunker rather than a parser, because the sentences are long and contain lots of coordinations that a parser cannot reliably handle.)

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

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 5: The decision-list learning algorithm.

computed with the formula:

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

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

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 rules are learned.

We apply the DL algorithm for each of the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS. For the former, the input to the algorithm is: the set E of all chunks, the main seed words (*suggest*, *imply*, *denote*, *mean*, *designate*, *connote*), the restriction that the part-of-speech (pos) for words in main rules be verbs and nouns, and the restriction that the pos for words in auxiliary rules be adverbs and modals. For the latter, the input to the algorithm is: the set E of all chunks, the main seed words (*formal*, *informal*, *pejorative*, *disapproval*, *favorable*, *abstract*, *concrete*), and the restriction that the pos for words in main rules be adjectives and nouns and in auxiliary rules be adverbs.

For example, for the class DENOTATIONAL DISTINCTIONS, starting with the rule *suggest* $\rightarrow 0.99$, 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* with confidence factors given by the count of these words in the current set of selected examples compared with the count in the rest of the set of examples. Using the new auxiliary rules for the words *sometimes* and *can*, 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*. Using new main 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
```

For this example, main rules contain the words *formal* and *concrete*, and auxiliary rules *much*, *more*, and *most*.

5 Extracting knowledge from CTRW

5.1 Classification and extraction

After we run the DL algorithm for the class DENOTATIONAL DISTINCTIONS, the words in the list mainDL are manually split into three classes: SUGGESTION, IMPLICATION, and DENOTATION. Some words can be insignificant for any class (e.g., the word *also*) or for the given class; therefore they are classified as the class OTHER and filtered out. We repeat the same procedure for *frequencies* and *strengths* with the words in auxDL. The words classified as OTHER and the patterns that do not contain any word from mainDL are ignored in the next processing steps.

After we have run the algorithm for the class ATTITUDE-STYLE DISTINCTIONS, the words in the list mainDL have to be split into two classes: ATTITUDE and STYLE. ATTITUDE is split into FAVORABLE, NEUTRAL, PEJORATIVE. STYLE is split into FORMALITY, CONCRETENESS, FORCE. *Frequencies* can be computed from the auxDL list. *Strengths* will be computed by the module that resolves comparisons.

The knowledge-extraction component takes each sentence in CTRW and tries to extract one or more pieces of knowledge from it. It considers what near-synonyms the sentence fragment is about, what the expressed distinction is, and with what frequency and relative strength. If it is a denotational distinction, then the peripheral concept involved must also be extracted. This module is very minimal for the moment. It relies on tuples ⟨*subject, verb, object*⟩ extracted by the chunker. Heuristics are used to correct cases when the information in the tuple is not accurate. When tuples are not available, it relies on patterns for the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS. Heuristics are used to extract the subject and object in this case. Improvements on our previous work include heuristics to retrieve compound-subjects of the form *NS and NS* and *NS, NS, and NS*. In order to determine the leaf class, we use the manual partitions of the rules in the mainDL of the two classes.

5.2 Coreferences and comparisons

Coreference resolution has been added since our earlier report (Inkpen and Hirst, 2001). We applied the same DL algorithm to retrieve expressions used to refer to near-synonyms or groups of near-synonyms. When running the algorithm with the seeds *noun, word, term, verb, adverb, adjective*, the expressions retrieved look like these:

```
[nx [dtp these] [nns verbs]]--330
[nx [dt the] [jj other] [nns adjectives]]--43
[nx [dt the] [vbg remaining] [nns nouns]]--28
```

The auxiliary words include: *the, three, both, preceding, previous, remaining, other*. By assigning meaning to these auxiliary words, more coreferences are resolved. Any time the subject is one of the main words (*noun, word, term, verb, adverb, adjective, preposition, nouns, words, terms, verbs, adverbs, adjectives, pair*), if there is an auxiliary word, the meaning is modified accordingly. For example, the expression *the remaining verbs* will cause the program to compute the set of near-synonyms of that entry not yet processed at that point.

CTRW often expresses stylistic or attitudinal features relative to other near-synonyms in the cluster. Such comparisons are easy to resolve because we consider only three levels (`low`, `medium`, `high`). We explicitly tell the system which words represent what absolute values of the corresponding feature (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).

6 Results and evaluation

CTRW contains 912 clusters, with a total of 14,138 sentences, from which we derive the lexical knowledge base. Our program is able to extract knowledge from 7450 of the sentences.

Table 1: Precision (P) and recall (R) of the baseline, our earlier system, and our present system.

	Baseline		Earlier system		Present system	
	P	R	P	R	P	R
All constituents	.40	.23	.61	.43	.66	.62
Class only	.49	.28	.68	.48	.71	.68

An example of final results, corresponding to the second, third, and fourth sentences in Figure 2, is this:

```
⟨absorb, usually, low, FORMALITY⟩
⟨absorb, usually, medium, SUGGESTION, the
   taking in of liquids⟩
⟨absorb, sometimes, medium, IMPLICATION,
   the thoroughness of the action⟩
```

In order to evaluate the final results, we randomly selected 25 clusters. We built by hand a standard solution to be compared with the results of our algorithm and with the results of a baseline algorithm. The baseline algorithm chooses the default values whenever it is possible; it is not possible for peripheral concepts (the direct object in the sentence) and for the near-synonyms the sentence is about (the subject in the sentence). The baseline algorithm relies only on tuples extracted by the chunker to extract the subjects and the objects.

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

Table 1 presents the evaluation of the 25 randomly selected clusters. The first row of the table presents the results as a whole (all the constituents of the extracted lexical knowledge-base). Our system increases precision by 0.26 and recall by 0.39 over the baseline. The second row of the table gives the results when only the (leaf) class of the distinctions expressed in CTRW is considered. In this case our system and the baseline algorithm attain higher precision, probably because the default class DENOTATION is the most frequent in CTRW.

Our system attains much better recall (0.21 more) than the earlier system presented in (Inkpen and Hirst, 2001) because it resolves coreferences. It is able to retrieve information about groups of near-synonyms referred to, for example, by the expression *the remaining words*. Small improvements in precision are due to better heuristics in the extractor component.

A problem in comparing the knowledge extracted from a sentence with the corresponding knowledge in the standard solution is the fact that often there are several pieces of knowledge to be aligned with several pieces in the standard solution. Our evaluation method aligns pieces of knowledge that are about the same near-synonym. Sometimes the near-synonym is extracted incorrectly or is missing, misleading the alignment. This is one possible explanation of the relatively low figures in Table 1.

7 Future work

The initial clusters we computed do not include the core denotations and the peripheral concepts. The peripheral concepts are implicitly there, but they are still strings (the literal noun phrases). In the results from the previous section, the peripheral concepts involved are *thoroughness* and *taking_in_liquids*. Peripheral concepts could be more complex: they can have attributes with discrete or numerical values. For each cluster we have to implement the following steps:

1. Deciding which senses of each near-synonym are the ones actually involved in the cluster. (It may be necessary to group these senses together.) This would help to decide what the core denotation of the cluster is. Sometimes the first sentence of the cluster states this. If not, maybe the most general near-synonym can help deciding the core denotation.
2. Deciding which are the peripheral concepts for the cluster. In this step, all distinctions extracted for the cluster are inspected. This step involves deciding what part of the noun phrase is relevant (the head and some of the adjectives).
3. Obtaining the final form of the cluster structure—that is the distinctions for each near-synonym—and adding new lexical items if necessary.
4. Reorganizing the ontology, if necessary.

If we use WordNet the disambiguation problem is difficult. There are too many senses for each word (for example, six senses for *error*, eight for *absorb*). We need to disambiguate to see what are the senses closely related to the peripheral concept. We have to group together senses which are very similar. The WordNet hierarchy has to be reorganized to accommodate the clusters of near-synonyms. From this point of view, Mikrokosmos (Mahesh and Nirenburg, 1995) is better because there are few senses for a word; closely related senses were merged into one sense when the ontology was built.

Other future work will focus on improving the results of the extraction module. Most mistakes are due to the wrong extraction of the subject and direct object of some sentences. We need to experiment with a different parser to see if more reliable subjects and objects can be extracted for this particular type of text. An alternative is to split complex sentences into simple sentences.

Another direction of further research is to extend Edmonds's representation to be able to represent all the distinctions adduced in CTRW. Examples of knowledge which do not fit in the current representation are information about generic versus specific near-synonyms and literal versus figurative meanings of near-synonyms.

Finally, a more-realistic evaluation of the lexical knowledge-base will have to be done in the context of an MT or NLG system.

Acknowledgments

We thank Suzanne Stevenson for useful discussions and comments on this work. We are grateful to HarperCollins Publishers, Inc. for permission to use CTRW in our project. Our work is financially supported by the Natural Sciences and Engineering Research Council of Canada and the University of Toronto.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the 8th European Summer School in Logic, Language and Information, Robust Parsing Workshop*.

Michael Collins 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*.

Chrysanne DiMarco and Graeme Hirst. 1993. Usage notes as the basis for a representation of near-synonymy for lexical choice. In *Proceedings of 9th annual conference of the University of Waterloo Centre for the New Oxford English Dictionary and Text Research*, pp. 33–43.

Philip Edmonds. 1999. *Semantic representations of near-synonyms for automatic lexical choice*. Ph.D. thesis, University of Toronto. *http://www.cs.toronto.edu/compling/Publications/Abstracts/Theses/EdmondsPhD-thabs.html*

Philip Edmonds and Graeme Hirst. 2000. Reconciling fine-grained lexical knowledge and coarse-grained ontologies in the representation of near-synonyms. In *Proceedings of the Workshop on Semantic Approximation, Granularity, and Vagueness*, Breckenridge, CO.

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

S.I. Hayakawa. 1994. *Choose the Right Word*. HarperCollins Publishers, Second edition.

Graeme Hirst. 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*, Stanford, pp. 51–56.

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

Diana Zaiu Inkpen and Graeme Hirst. 2001. Experiments on extracting knowledge from a machine-readable dictionary of synonym differences. In *Computational Linguistics and Intelligent Text Processing*, Alexander Gelbukh (ed.), LNCS 2004, Springer, pp. 265–280.

Kavi Mahesh 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*, Montreal, Canada.

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