Experiments on Extracting Knowledge from a Machine-Readable Dictionary of 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 the wrong word choice from a set of near-synonyms can be imprecise or awkward, or convey unwanted implications. Using Edmonds's model of lexical knowledge to represent clusters of near-synonyms, our goal is to automatically derive a lexical knowledge-base from the *Choose the Right Word* dictionary of near-synonym discrimination. We do this by automatically classifying sentences in this dictionary according to the classes of distinctions they express. We use a decision-list learning algorithm to learn words and expressions that characterize the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS. These results are then used by an extraction module to actually extract knowledge from each sentence. We also integrate a module to resolve anaphors and word-to-word comparisons. We evaluate the results of our algorithm for several randomly selected clusters against a manually built standard solution, and compare them with the results of a baseline algorithm.

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. Examples of near-synonymic variations are given in Table 1 (Edmonds 1999).

There are very few absolute synonyms, if they exist at all. The 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 the each cluster. They 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 consumption and they are available only on paper.

DiMarco, Hirst, and Stede (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 (DiMarco and Hirst 1993).

Table 1. Examples of near-synonymic variations

Type of variation	Example
Collocational	*task* : *job*
Stylistic, formality	*pissed* : *drank* : *inebriated*
Stylistic, force	*ruin* : *annihilate*
Expressed attitude	*skinny* : *thin* : *slim*
Emotive	*daddy* : *dad* : *father*
Continuousness	*seep* : *drip*
Emphasis on different subcomponents	*enemy* : *foe*
Fuzzy boundary	*woods* : *forest*

We hypothesize that the language of the entries contains enough regularities 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).[1] A page from this dictionary is presented in Figure 1.

2 The Clustered 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 defines words in terms of necessary and sufficient truth-conditions; therefore it cannot account for indirect expressions of meaning or for fuzzy differences between near-synonyms.

Edmonds and Hirst (2000) modify 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 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*. Each 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

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

abjure. Do not confuse the verb *abjure* (renounce under oath) with the verb *adjure* (urge solemnly).

abrogate. Do not confuse the verb *abrogate* (cancel or repeal) with the verb *arrogate* (claim a power, privilege, etc., unduly).

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.*

Fig. 1. A page from *Choose the Right Word* by S.I. Hayakawa. Copyright ©1987. Reprinted by arrangement with HarperCollins Publishers, Inc.

by means of peripheral concepts (for denotational nuances) or attributes (for nuances of style and attitude).

In this model, a cluster includes the following fields:

1. `syns` — A list of near-synonyms in the cluster.
2. `core` — The core denotation, or essential shared meaning of the near-synonyms in the cluster. It is represented as a configuration of concepts.
3. `periph` — A set of peripheral concepts that extend the core denotation, and pertain to the differentiation of the near-synonyms.
4. `distinctions` — The actual distinctions between near-synonyms.

For example, the (slightly simplified) structure for the near-synonyms of the word *error*, built by hand by Edmonds (1999), is shown in Figure 2.

Building such representations by hand is difficult and time-consuming, and Edmonds completed only nine of them. We want to automatically extract the content of all the entries in the CTRW dictionary of near-synonyms. In order to build a practical lexical knowledge-base, we use a simplified form of Edmonds's representation for the content of a cluster.

3 Preprocessing CTRW

After OCR scanning 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 actual 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. The markup is quite complex, but only part of it is relevant for the purpose of this paper. An example of segmented content from Figure 1 is given in Figure 3.

In the text of the entry for a cluster, the first occurrence of a synonym is in bold face, and all the other occurrences are in italic. We marked each occurrence with the tag `<near_syn>`. This is useful when we want to extract general patterns, when we don't care what the near-synonym is but just need to know that it is a near-synonym.

We determine sentence boundaries and clearly label examples, which will not presently be considered. We use general heuristics to detect the end of sentences, plus specific heuristics for this particular dictionary; e.g., examples are in square brackets or after a colon. The tag `<s>` marks sentences which describe the nuances of meaning, and the tag `<eg>` marks examples using the near-synonyms. The `<s>` tags have the attribute `punctuation`, taking as value the punctuation mark ending the sentence (full stop, question mark, exclamation mark, etc.). If the value is ':', that means that the sentence ended, but one or more examples immediately follow. Examples are of two kinds: those that follow a colon, and those contained between square brackets. Therefore, the `<eg>` tag has, besides the attribute `punctuation`, the attribute `type` (':' or ']').

4 Relating CTRW to Edmonds's Representation

From each sentence of the dictionary we need to extract the information relevant to the representation. We rely on the relative regularity of the language of the entries.

```
(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)
     (mistake_l low concreteness)
     (howler_l low formality)
    )
)
```

Fig. 2. Simplified cluster for *error*, *mistake*, *blunder*, *slip*, *lapse*, *howler*.

4.1 Core Denotation

Usually, the first sentence in a CTRW entry expresses the core meaning shared by all members of the family. An example is: *These verbs all refer to rather forceful communications between a speaker and another person*. In some entries, this information is skipped, and the first sentence is about the first near-synonym in the cluster.

4.2 Denotational Distinctions

Near-synonyms can differ in the frequency with which they express a component of their meaning (e.g., ***hard up*** *often implies only a temporary shortage of money*; *Occasionally,* ***invasion*** *suggests a large-scale but unplanned* ***incursion***), in the 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*

```
<s punctuation="."> These verbs, all relatively formal, indicate
  the taking in of one thing by another </s>
<s punctuation="."> <near_syn><b>Absorb</b></near_syn> is
  slightly more informal than the others and has, perhaps, the
  widest range of uses </s>
<s punctuation=":"> In its most restricted sense it suggests the
  taking in or soaking up specifically of liquids </s>
<eg type=":" punctuation="."> the liquid <near_syn><i>absorbed
  </i></near_syn> by the sponge </eg>
<s punctuation=":"> In more general uses <near_syn><i>absorb</i>
  </near_syn> may imply the thoroughness of the action </s>
<eg type=":" punctuation="."> not merely to read the chapter,
  but to <near_syn><i>absorb</i></near_syn> its meaning </eg>
<s punctuation="."> To <near_syn><b>digest</b></near_syn> is to
  alter food chemically in the digestive tract so that it can be
  <near_syn><i>absorbed</i></near_syn> into the bloodstream </s>
<s punctuation="."> In other uses, <near_syn><i>digest</i>
  </near_syn> is like <near_syn><i>absorb</i></near_syn> in
  stressing  thoroughness, but is even more emphatic </s>
<eg type="]" punctuation="."> You may completely <near_syn><i>
  absorb </i></near_syn> a stirring play in one evening, but you
  will be months <near_syn><i>digesting</i></near_syn> it </eg>
```

Fig. 3. Example of text from CTRW with XML markup

either benevolent rule or a style of government determined to keep the governed helpless and dependent).

For *denotational distinctions* we extract tuples of 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 the obvious words *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, among others. `Peripheral-concepts` form the basic vocabulary of fine-grained denotational distinctions. In Edmonds (1999) they are structures of concepts defined in the same ontology as the core denotations. Every peripheral concept extends the core denotation in some way, but they represent non-necessary and indirect aspects of meaning. In DiMarco and Hirst (1993) peripheral concepts can be a binary, continuous, or discrete dimension. However, in our first stage of extracting knowledge, we treat the peripheral concepts as strings, without analyzing them more deeply. From a sentence fragment such as ***precipice*** *usually suggests danger*, we can extract `precipice usually medium suggestion danger`. Default values are used when `strength` and `frequency` are not specified in entry.

4.3 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`. Examples of sentences in CTRW expressing attitudes are: ***Blurb*** *is also used pejoratively to denote the extravagant and insincere praise common in such writing* and ***Placid*** *may have an unfavorable connotation in suggesting an unimaginative, bovine dullness of personality*. Both contain information about the pejorative attitude, though they also contain information about denotational distinctions.

The information we extract for attitudinal distinctions has the form *`near-synonym frequency strength attitude`*, where *`strength`* and *`frequency`* have the same values and significance as in the case of denotational distinctions. Edmonds (1999) has an extra element, a pointer to the entity the attitude is directed towards.

We are not always able to extract from the beginning absolute values for the strength of the feature, because they are presented as comparisons between near-synonyms in the cluster, which must be resolved after extraction. The initial information extracted may have the form *`near-synonym frequency strength comparison-term attitude near-synonym`*. For example, from the sentence ***Sycophantic*** *is sometimes used interchangeably with* ***obsequious****, but is more strongly pejorative* we extract `sycophantic usually more pejorative obsequious`. After the comparison is resolved (see Section 7.3) this becomes `sycophantic usually high pejorative`.

4.4 Stylistic Distinctions

In the absence of a comprehensive theory of style, the representation uses a basic approach to representing the stylistic variation of near-synonyms. The information we need to extract 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.

Because the stylistic attributes are expressed by adjectives (or nouns derived from these adjectives), the information is rarely absolute, but is relative to the other near-synonyms in the cluster. Comparatives and superlatives are very frequent. So, in the first phase we may extract information of the form `near-synonym comparison-term stylistic-feature near-synonym`. From a sentence such as ***Designate*** *is the most formal of all these terms*, the information extracted in the end (after the comparisons are resolved) is `designate high formality`.

Words that signal `formality` include *formal*, *informal*, *formality*, and *slang*. `Concreteness` is signaled by words such as *abstract*, *concrete*, and *concretely*. From the sentence ***Center****, most concretely, indicates such a point within the circumference of a circle or a sphere*, we should extract `center high concreteness`. The third kind of stylistic information we extract is `force` (`floridity` and `familiarity` will be dealt with in future work). From the sentence ***Request*** *is considerably weaker than any* *sense of* ***demand***, we should extract `request less force demand`, and in the end `request low force`.

A sentence in CTRW can contain more than one piece of information. Consider the sentence ***Coworker*** *is the least formal of these and applies as readily to manual labor as to more highly skilled occupations*. We need to extract information both about the stylistic feature `formality` and about the denotations of manual labor and highly skilled occupations.

5 The Class Hierarchy of Distinctions

Following Edmonds's analyses 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 class ATTITUDE-STYLE DISTINCTIONS because they present similar behavior from the point of view of this research (see explanation in Section 6).

The leaf classes are those that we classify CTRW sentences into. The leaves of DENOTATIONAL DISTINCTIONS are SUGGESTION, IMPLICATION, and DENOTATION; those of ATTITUDE are FAVORABLE, NEUTRAL, and PEJORATIVE; those of STYLE are FORMALITY, CONCRETENESS, FORCE, FLORIDITY, and FAMILIARITY. 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`.

In order to automatically create near-synonym representations, we must be able 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) we have 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.

6 The Decision-List Learning Algorithm

Unsupervised learning may be used when annotated corpora are not available. Yarowsky's (1995) work on word sense disambiguation using an unsupervised decision-list algorithm inspired many researchers. He classified the senses of a word on the basis of other words that given word co-occurs with. Collins and Singer (1999) classified proper names as `Person`, `Organization`, or `Location` using contextual rules (other words appearing in the context of the proper names). Starting with a few spelling rules (some proper-name features) in the decision list, their algorithm learns new contextual rules; using these rules then it 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

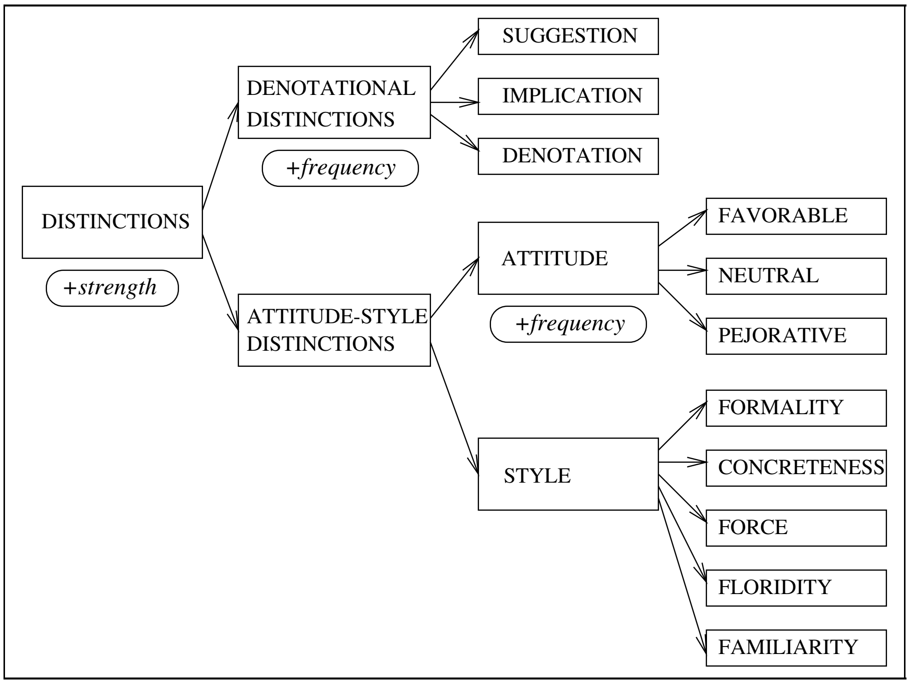

Fig. 4. The class hierarchy of Distinctions (rectangles represent classes, ovals represent attribute a class and its descendents have).

technique to alternately select the best extraction pattern for a category and add its extractions to the semantic lexicon, which is the basis for selecting the next-best extraction pattern.

Our decision-list (DL) algorithm (Figure 5) is tailored for extraction from CTRW. It takes ideas from the previously mentioned work. Like Collins and Singer, we learn two kinds of rules, main rules and auxiliary rules. We also extract patterns and relevant words for the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS, similar to the domain-specific lexicon extracted by Riloff and Jones.

In order to obtain input data, we chunk the text with Abney's chunker (1996). The training set E is formed from all the verb phrases, noun phrases, adjectival 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.)

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

We compute the confidence of a word x with the formula:

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

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 $x \to h(x)$, 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.

Fig. 5. The decision-list learning algorithm.

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 versus the total number of occurrences in the training data.

We learn 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), we can select examples containing them and learn a few auxiliary rules. Using these we can select more examples and learn new main rules. We keep iterating until no more new rules are learned.

We apply the DL algorithm for each of the classes DENOTATIONAL DISTINCTIONS and ATTITUDE-STYLE DISTINCTIONS.

For the class DENOTATIONAL DISTINCTIONS 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 class ATTITUDE-STYLE DISTINCTIONS the input to the algorithm is: the set E of all chunks, the main seed words (*formal*, *informal*, *pejorative*, *disapproval*, *favorable*, *abstract*, *concrete*), the restriction that the pos for words in main rules be adjectives and nouns, and the restriction that the pos for words in auxiliary rules be adverbs.

For example, for the class DENOTATIONAL DISTINCTIONS, starting with the rule *suggest* $\rightarrow 0.99$, we select 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
```

We learn auxiliary rules 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*, we select more examples such as these:

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

From these we learn new main rules, for the words *refer* and *imply*. Using new main rules we select more auxiliary rules for the word *may*, and so on.

We considered extracting patterns for each leaf class in the hierarchy, but after analyzing part of the text, we reached the conclusion that there are just two kinds of patterns: for DENOTATIONAL DISTINCTIONS and for ATTITUDE-STYLE DISTINCTIONS. That is, patterns are characteristic for each of the two top-nodes in the class hierarchy. The following examples show some of the patterns that describe the DENOTATIONAL DISTINCTIONS class:

```
[vx [rb often] [vbz suggests]]---65
[vx [md may] [vb indicate]]---65
[vx [md may] [rb also] [vb refer]]---62
```

The auxiliary words that form the pattern (i.e, *often*, *also*, *may*) are the same for the subclasses DENOTATION, IMPLICATION, SUGGESTION.

For these examples from the DENOTATIONAL class, we derive main rules that *suggests*, *indicate*, and *refer* are significant for the class (with a computed confidence). Auxiliary rules say that *often*, *also*, and *may* are significant for `frequency` or `strength` (with a computed confidence). The decisions for the subclasses are made later.

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

```
[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*.

7 Extracting Knowledge from CTRW

7.1 Classification

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 in the class OTHER. We repeat the same procedure for `frequencies` and `strengths` with the words in auxDL. The words classified as OTHER and the patterns which 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. `Frequencies` can be computed from the auxDL list. STYLE is split into FORMALITY, CONCRETENESS, FORCE. `Strengths` will be computed by the module which resolves comparisons.

7.2 Extraction

The knowledge extraction module 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 has to also be extracted. This module is very minimal for the moment. It relies on tuples ⟨*verb, subject, object*⟩ extracted by the chunker. Simple 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. Simple heuristics are used to extract the subject and object in this case. In order to decide the leaf class, we use the manual partitions of the rules in the mainDL of the two classes.

For example, consider three sentences from the cluster *imbibe, digest, absorb, ingest, assimilate, incorporate* depicted in Figure 1. The results of extraction at this stage are the following:

```
near_syn(absorb) is slightly more informal than the others and
has, perhaps, the widest range of uses
Patterns:
Tuples:    informal :than others :subj near_syn(absorb)
RESULT: subj: near_syn(absorb) class: FORMALITY(informal)
than: others freq:  degree:

in its most restricted sense it suggests the taking in or
soaking up specifically of liquids
Patterns: [vx [vbz suggests]]
Tuples:   suggests :obj taking :in in :subj it
RESULT: subj: near_syn(absorb) class: SUGGESTION(suggests)
periph: the taking freq:  degree:

in more general uses near_syn(absorb) may imply the thoroughness
of the action
Patterns: [vx [md may] [vb imply]]
Tuples: imply   :subj thoroughness
RESULT: subj: near_syn(absorb) class: IMPLICATION(imply) periph:
the thoroughness of the action freq: sometimes(may) degree:
```

7.3 Anaphors and Comparisons

The final module resolves anaphors and comparisons. Anaphora resolution is very simplistic. The last-mentioned near-synonym is recorded in a stack. When *it* is a subject, it most probably refers to the near-synonym on top of the stack. There are cases when the sentence is about two or more near-synonyms or is about all the words in a class. There are other anaphoric expressions we need to resolve (such as *both verbs* and *the remaining nouns*).

CTRW often expresses stylistic or attitudinal features relative to other near-synonyms in the cluster (see examples in Sections 4.3 and 4.4). 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 `medium` value of CONCRETENESS).

8 Results and Evaluation

CTRW contains 912 clusters, with a total of 14138 sentences, from which we derive the lexical knowledge base. Our program is able to extract knowledge from 7450 of the sentences.

An example of final results, corresponding to the example in the previous section, is the following:

```
absorb usually low FORMALITY
absorb usually medium SUGGESTION the taking
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. As the baseline algorithm, we choose 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 *labeled precision* and *labeled recall*. These measures are usually used to evaluate parse trees obtained by a parser (Charniak 1997); they allow for rewarding good partial results. 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). Labeled 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. Labeled recall is the total number of correct constituents found divided by the number of constituents in the standard solution.

Table 2. Labeled precision and labeled recall of the baseline and of our algorithm and the error reduction.

	Baseline algorithm		Our algorithm		Error reduction rate	
	Labeled precision	Labeled recall	Labeled precision	Labeled recall	Labeled precision	Labeled recall
All constituents	.40	.23	.61	.43	35%	26%
Class only	.49	.28	.68	.48	37%	28%

For example, for the sentence ***artificer** still can suggest its earliest meaning of a worker who possesses mechanical facility*, the program obtains: `artificer sometimes medium SUGGESTION its earliest meaning`, while the solution is `artificer sometimes medium SUGGESTION a worker who possesses mechanical facility`. The labeled precision is .80 (4 correct out of 5 found), and the labeled recall is .80 (4 correct out of 5 in the standard solution).

Table 2 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 algorithm increases labeled precision by .21 and labeled recall by .20. Because the baseline algorithm has very low values for labeled precision and recall, it is useful to calculate the error reduction rate, defined as the difference of the error of the baseline algorithm and the error of our algorithm, divided by the error of the baseline algorithm. Our algorithm reduces the error rate of the baseline algorithm by 35% in labeled precision and by 26% in labeled recall. The second row of the table gives the results when only the (leaf) class of the distinctions expressed in CTRW is considered. By considering only whether the class is right, we evaluate more directly the results of the DL learning algorithm. In this case our algorithm attains higher labeled precision, and so does the baseline algorithm (probably because the default class DENOTATION is the most frequent in CTRW).

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. Because the anaphors are resolved only minimally, sometimes the near-synonym is extracted incorrectly or is missing, misleading the alignment. This is one possible explanation of the relatively low figures for recall in Table 2.

A more-realistic evaluation of the lexical knowledge-base would have to be done in the context of a machine translation (or natural language generation) system.

9 Conclusion and Future Directions

This paper presents a first step towards building a lexical knowledge-base by automatically extracting knowledge from a dictionary of near-synonym discriminations.

The methods used in this work still need to be improved. We need to use a more reliable method of extracting subjects and objects from multi-clause sentences. We need to implement better anaphor resolution. We need a better understanding of the nuances of the language of CTRW itself.

One of the next steps is to treat peripheral concepts, to decide what the peripheral concept involved is and what its place in the ontology is.

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 we cannot fit in the current representation are information about generic versus specific near-synonyms and literal versus figurative meanings of near-synonyms.

Acknowledgments

We thank Suzanne Stevenson for useful discussions and comments on earlier versions of this paper. 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

Abney, Steven: Partial parsing via finite-state cascades. Proceedings of the ESSLLI '96 Robust Parsing Workshop (1996)

Charniak, Eugene: Statistical techniques for natural language parsing. AI Magazine 18(4) (1997) 33–44

Collins, Michael and Singer, Yoram: Unsupervised models for named entity classification. In Proceedings of Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99) (1999)

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

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

Edmonds, Philip: Semantic representations of near-synonyms for automatic lexical choice. Ph.D Thesis, University of Toronto (1999)

Edmonds, Philip and Hirst, Graeme: 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, Colorado (2000)

Gove, P.B. (ed.): Webster's New Dictionary of Synonyms. G.&C. Merriam Co. (1984)

Hayakawa, S.I., Ehrlich Eugene (revising ed.): Choose the Right Word. HarperCollins Publishers, Second edition (1994)

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

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

Riloff, Ellen and Jones, Rosie: Learning dictionaries for information-extraction by multi-level bootstrapping. In Proceedings of the Sixteenth Conference on Artificial Intelligence (AAAI-99) (1999) 474–479

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