Detecting and Correcting Malapropisms with Lexical Chains

by

David St-Onge

Department of Computer Science
University of Toronto
Toronto, Canada

March 1995

A thesis submitted in conformity with the requirements
for the degree of Master of Science at the
University of Toronto

Abstract

Because chains of semantically related words express semantic continuity, such lexical chains can play an important role in the detection of malapropisms. A malapropism is a correctly spelled word that does not fit in the context where it is used because it is the result of a spelling error on a different word that was intended.

I first assume that such a word has much less probability of being inserted in any chain with other words. If this assumption is correct, words that failed to be inserted with other words can be considered as potential malapropisms. A mechanism that generates spelling replacements can then be used to generate replacement candidates. The second assumption is that whenever a spelling replacement can be inserted in a chain with other words, this replacement is likely to be the intended word for which a malapropism has been substituted.

The algorithm proposed here to detect lexical chains uses the on-line thesaurus WordNet to automatically quantify semantic relations between words. Chains identified by the algorithm may have two major problems: over- or under-chaining. Under-chaining—the inability to link a pair of related words—might be caused by an inadequacy of WordNet's set of relations, a lack of connections in WordNet's set of relations, a lack of connections in WordNet, a lack of consistency in the semantic proximity expressed by WordNet's links, and a poor algorithm for chaining. Over-chaining—the linking of two poorly related words—might happen whenever two semantically distant words are close to each other in WordNet's graph. Over-chaining often results in the merging of two chains.

The results of the experiment show the validity of the basic assumptions. However, improvements to the lexical chaining algorithm are required before the malapropism detection algorithm can be integrated into a commercial spelling checker.

Acknowledgments

First of all, I would like to thank Dr. Graeme Hirst for being much more than my thesis supervisor. His personal guidance and devotion to my success have been greatfully appreciated.

Secondly, thanks must be given to my second reader, Dr. Jeffrey Siskind, for his constructive comments.

Thirdly, I am grateful to FCAR (Formation de Chercheurs et Aide à la Recherche) for their research scholarship, and NSERC (Natural Sciences and Engineering Research Council), who financed the equipment used for this research and provided funds to my department for my research assistantship.

Fourthly, I must thank Steve Green, Daniel Marcu, and Philip Edmonds for their help and feedback.

I would also like to take this opportunity to thank my parents for their complete support throughout all my years of study.

Finally, many thanks to Susanne, my German rose, for having reinforced my motivation to complete this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Overview of Lexical Chains

A *lexical chain* is a succession of semantically related words in a text, that creates a context and contributes to the continuity of meaning (Morris and Hirst, 1991). To illustrate this, let us consider the following text:

> We suppose a very long train$_1$ traveling$_2$ along the rails$_1$ with the constant velocity$_2$ v and in the direction$_2$ indicated in Figure I. People traveling$_2$ in this train$_1$ will with advantage use the train$_1$ as a rigid reference-body$_3$; they regard all events in reference$_3$ to the train$_1$. Then every event which takes place along the line$_1$ also takes place at a particular point$_1$ of the train$_1$. Also, the definition of simultaneity can be given relative to the train$_1$ in exactly the same way as with respect to the embankment$_1$. (Einstein, 1939)

Three lexical chains can be identified:

1. {train, rails, train, train, train, *line*, point, train, train, embankment}
2. {traveling, velocity, direction, traveling}
3. {reference-body, reference}

The context created by the first chain enables the identification of the appropriate meaning of the word *line*. Word-sense disambiguation is, in fact, an important application for lexical chaining and will be discussed in Section 1.3.

Another application for lexical chains is Morris and Hirst's (1991) use of the correspondence between lexical chains and structural units within a text as an indicator of the whole text structure. Morris and Hirst based their theory on the following observation:

...when a chunk of text forms a unit within a discourse, there is a tendency for related words to be used. It follows that if lexical chains can be determined, they will tend to indicate the structure of the text. (Morris and Hirst, 1991)

The first step of their work was to identify the lexical chains of a text using their intuition (*i.e.*, common sense and knowledge of English). Then, they used the results to develop a method for identifying lexical chains.

By grouping words that express some semantic continuity, lexical chains provide a new approach for many AI fields such as information retrieval or word-sense disambiguation. Lexical chains also open doors to original applications like intelligent spelling checking (see Chapter 3), identification of text structure, summarization, etc.

The first aim of my research is to provide an entirely automatic tool for identifying lexical chains. This software could be used for a variety of different research areas involving lexical chains. My second aim is to apply this lexical chainer in a novel way to the detection of malapropisms. A malapropism is a correctly spelled word that does not fit in the context where it is used because it is the result of a spelling error on a different word that was intended.

I first assume that such a word has much less probability of being inserted in any chain with other words. If this assumption is correct, words that failed to be inserted with other words can be considered as potential malapropisms. A mechanism that generates spelling replacements can then be used to generate replacement candidates. The second assumption is that whenever a spelling replacement can be inserted in a chain with other words, this replacement is likely to be the intended word for which a malapropism has been substituted.

The next three sections describe the lexical chain–related research that influenced my implementation of a lexical chainer.

1.1 Morris and Hirst's Algorithm

Written by Peter Mark Roget in 1852, *Roget's International Thesaurus* is a classification of words and phrases around ideas and concepts. Its primary purpose is to enable writers to find an accurate word or phrase in specific circumstances. The fifth edition of *Roget's International Thesaurus* (Roget, 1992) classifies 325,000 words and phrases into 1073 semantic categories. These categories are grouped into five classes (e.g., *the body and the* *senses, feelings, place and change of place*). Categories are divided into sub-categories, that are grouped according to the four main grammatical categories: noun, verb, adjective, and adverb. At the end of the book, a comprehensive index lists all key words and key phrases with their respective list of sub-categories. Each sub-category contains words and ideas that express a similar concept or idea, but does not necessarily contain the given key word or key phrase.

Morris and Hirst's (1991) algorithm to construct lexical chains identifies relations between words by using *Roget's International Thesaurus*. Compound words are considered, but words are not merged together into phrases (*i.e.*, *merry-go-round* is accepted but *elementary school* is considered as two separate words). The algorithm considers two words to be related to each other if their stems satisfy any one of these six conditions:

1. they are both identical;
2. they both have an index entry that refers to the same category;
3. they both have an index entry that refers to a different category but one of these two categories has a pointer to the other one;
4. one has an index entry that refers to a category containing the other stem;
5. they are both contained in the same sub-category;
6. they both have an index entry that refers to a different category but these two categories have a common pointer to another category.

When a new word is read, the first step is to make sure that it is suitable for lexical analysis. Words like pronouns, prepositions, verbal auxiliaries, and other high-frequency words (*e.g.*, *good, do*) are filtered out. If the new word is not rejected, then a backward search is made through the whole text to see whether an earlier occurrence of the word is found (first rule). If no such relation is found, then a new backward search is performed to find a word that connects according to one of the other rules (2–6). Such backward searching implies that chains are selected in their most recent order. The scope of the second search is limited to three sentences. However, a *one-word transitivity* is permitted. This transitivity means that if a word is not in the search scope of the current word but has been previously linked to a word located in the search scope, then it is considered for a relation evaluation with the current word.

Due to the lack of a machine-readable thesaurus, Morris and Hirst did not implement their algorithm.

1.2 Okumura and Honda's Algorithm

Okumura and Honda (1994) took a different approach from Morris and Hirst. First, they suggested that chains should not be selected in their most recent order, but in order of *salience*. Salience is defined according to both the recency and the length of a chain. A stack is used to maintain chains in order of salience. Second, lexical cohesion is checked inside the current sentence before popping any word from the stack. As a result, chains are not updated after each word but after each sentence, thereby giving priority to intra-sentential information. The lexical knowledge base used by Okumura and Honda is a Japanese thesaurus similar to *Roget's*.

1.3 Word-Sense Disambiguation

An earlier form of word-sense disambiguation is Hirst's (1988) disambiguation system called Polaroid^{TM} Words (PW). It progressively determines the meaning of a word in a fashion similar to the development of a Polaroid photograph. When a sentence is read, a PW structure is assigned to the current word and is filled with a certain packet of knowledge containing, among other things, the different interpretations of the word. This PW is then compared with the previous PWs of the same sentence that are related to it by some predefined syntactic relations. The different interpretations of the PWs are progressively eliminated as the comparisons occur within the sentence.

Okumura and Honda (1994) used the chains identified by their algorithm to disambiguate words. Each chain is used to provide a context. When a word is added to a chain, its ambiguity is resolved. The sense of this word is therefore determined by rejecting other candidate senses that correspond to other lexical chains. This method is called incremental word sense disambiguation because each time a word is added to a chain, it helps clarify the sense of the other words of the chain.

Chapter 2

Automatic Identification of Lexical Chains

In Morris and Hirst's work, the knowledge source used is not machine readable and a lexical chain structure is simply a set of words. To actually implement a lexical chainer, a machine-readable knowledge base and more complex data structures are required. Relations between words are redefined according to the new knowledge base and a stack of chains is implemented to manage chains during their construction. Stack, chains, and words are all objects that include data structures and functions.

2.1 WordNet 1.4: A Lexical Knowledge Base

An investigation of the different lexical knowledge bases available led me to choose WordNet 1.4[1]. *Roget's thesaurus* exists in two electronic versions: the 1911 version, which has no index and suffers from a lack of new words, and the 1993 version, which cannot be licensed from the publisher.

WordNet (Miller et al., 1990) is a lexical knowledge base developed at Princeton University. It is divided into four data files containing data for adjectives, adverbs, nouns, and verbs respectively. An index file is associated with each data file (see Figure 2.1). Each data file contains a set of tuples. Each tuple corresponds to a *synonym set*, or *synset*, which contains words of similar meaning that refer to a common semantic concept. As words can

[1]WordNet is a trademark of Princeton University.

have more than one meaning, a word may be present in more than one synset. WordNet 1.4 groups 100,665 words and phrases into 70,305 synsets.

A list of pointers is attached to each synset. These pointers express relations between synsets. WordNet 1.4 contains 11 such lexical relations. Table 2.1 lists them along with the direction that I assigned to each of them. A downward relation goes from a given word to a more specific one. For instance, if *soupspoon* is an hypernym of *spoon*, then *spoon* is the initial word and *soupspoon* is a more specific word, resulting in a downward relation from *spoon* to *soupspoon*. An upward relation goes from a given word to a more general one. Finally, a horizontal relation goes from a given word to another word that is neither more specific nor more general. Figure 2.2 gives a sample from an index file while Figure 2.3 gives a sample data token from the noun data file.

A closer look at Figure 2.1 reveals that the four data files are not completely connected together. The following are the existing relations between data files:

- nouns refer to adjectives with *attribute* relations;
- adjectives refer to nouns with *attribute* and *pertain* relations;
- adverbs refer to adjectives with *pertain* relations;

2.2 Semantic Relations

One way to think of the natural language lexicon is to imagine a semantic universe where words are located with respect to each other according to their meaning. Semantically similar words are located more closely to each other than less similar words. Words that have more than one meaning appear at different locations according to their different meanings.

This introduces the notion of *semantic distance*. For instance, *goal* and *objective* have a short semantic distance, *goal* and *achievement* have a larger semantic distance, and *goal* and *thought* have a much larger semantic distance. Words that have short semantic distances have a *semantic proximity*. Each relation used in WordNet corresponds to a certain a semantic proximity between two words. Although it is relatively easy for a reader to determine whether or not two words are semantically similar, it is much harder to quantify this semantic distance. Therefore, when using WordNet to determine whether or not two words are sufficiently similar to form a chain link, a method must be chosen to evaluate the

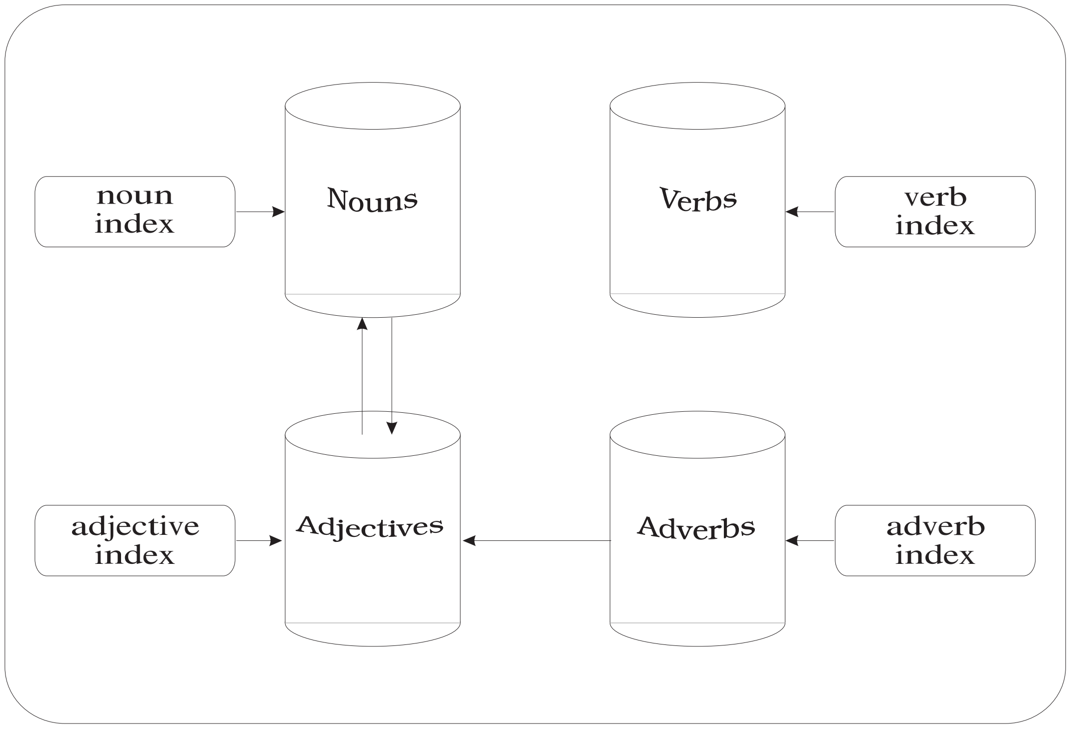
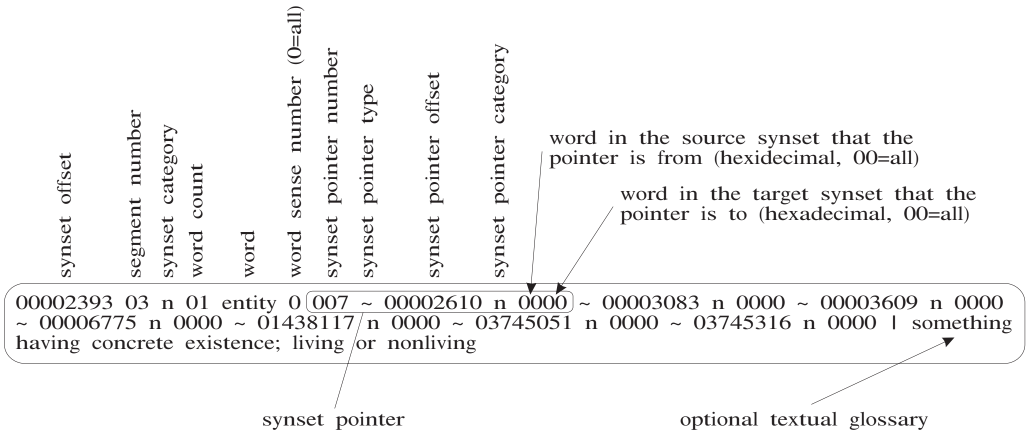


Figure 2.1: WordNet 1.4 Structure

```
prince_edward_island n 0 3 @ #p %p 1 04201200
prince_of_wales n 0 1 @ 1 04712962
prince_of_wales_heath n 0 1 @ 1 05270052
prince_otto_von_bismarck n 0 1 @ 1 04816164
princedom n 2 2 @ ~ 1 04130839
princeship n 0 1 @ 1 00214954
princess n 4 3 @ ~ #m 1 04713071
princess_pine n 0 1 @ 1 05850687
princess_royal n 1 1 @ 1 04713233
princeton n 0 3 @ #p %p 2 04274481 01636405
princeton_university n 0 2 @ #p 1 01636405
principal n 12 3 @ ~ #p 5 05958166 05938454 04767075 04713496
 04713335
principal_axis n 2 1 @ 1 03088623
```

Figure 2.2: Index File Sample



```
00002393 03 n 01 entity 0 007 ~ 00002610 n 0000 ~ 00003083 n 0000 ~ 00003609 n 0000
~ 00006775 n 0000 ~ 01438117 n 0000 ~ 03745051 n 0000 ~ 03745316 n 0000 | something
having concrete existence; living or nonliving
```

synset offset
segment number
synset category
word count
word
word sense number (0=all)
synset pointer number
synset pointer type
synset pointer offset
synset pointer category
word in the source synset that the pointer is from (hexidecimal, 00=all)
word in the target synset that the pointer is to (hexadecimal, 00=all)
synset pointer
optional textual glossary

Figure 2.3: Sample Data Token

Table 2.1: Lexical Relations in WordNet 1.4

also see		(direction: horizontal)
	definition:	Refer to another expression that is not related semantically but that includes a given word.
	example:	*wash* also see *wash_up*, *freshen_up*
antonym		(direction: horizontal)
	definition:	Word that as the opposite meaning.
	example:	*lightness* is the antonym of *darkness*
attribute		(direction: horizontal)
	definition:	Relation of implication between a noun and an adjective, and vice versa.
	example:	*serious* is an attribute of *sincerity*
cause		(direction: down)
	definition:	Cause of another action.
	example:	*anesthesize* is a cause of *sleep*
entailment		(direction: down)
	definition:	Implication of another action.
	example:	*breathe* is an entailment of *inhale*
holonym		(direction: down)
	definition:	Corresponds to the part in a part-whole relation. Three kinds of holonyms are used: by member, by substance, and by part.
	examples:	*verb* is a member holonym of *conjugation*
		lesson is a part holonym of *course*
		cranberry is a substance holonym of *cranberry_sauce*
hypernym		(direction: up)
	definition:	Generalization of a word. Its opposite is the hyponym.
	example:	*canine* is an hypernym of *dog*
hyponym		(direction: down)
	definition:	Specification of a word. The hyponym relation is often called IS-A. Its contrary is the hypernym.
	example:	*soupspoon* is an hyponym of *spoon*
meronym		(direction: up)
	definition:	Corresponds to the whole in a part-whole relation. Three kinds of meronyms are used: by member, by substance, and by part.
	examples:	*plant_kingdom* is a member meronym of *plant*
pertain		(direction: horizontal)
	definition:	Relation from a noun to an adjective, an adjective to a noun, or an adverb to an adjective indicating a morphological relation.
	example:	*alphabetical* pertains to *alphabet*
similar		(direction: horizontal)
	definition:	Refer to another adjective that is very close in terms of meaning to the current adjective, although not enough to be part of the same synset.
	example:	*unquestioning* is similar to *absolute*

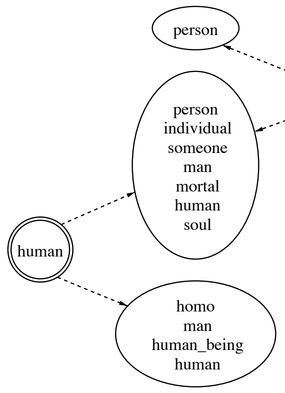


Figure 2.4: Strong Relation I

semantic distance expressed by the path (succession of one or more links) that goes from one word to the other. This is the aim of the next section.

2.3 Algorithm and Implementation

2.3.1 Relation Between Words

As stated earlier, each word in WordNet may appear in more than one synset. Each synset corresponds to a different sense of the word. When attempting to find a relation between two different words, each synset of the first word must be considered with each synset of the second word. Three kinds of relation between words have been defined: extra-strong, strong, and medium-strength. An extra-strong relation corresponds to a word repetition (e.g., *vehicle* and *vehicle, window* and *windows, foot* and *feet*). Extra-strong relations have the highest weight of all relations.

A strong relation has a lower weight than any extra-strong relation and a higher weight than any medium-strength relation. Strong relations have been divided in three sub-categories. The first one, illustrated in Figure 2.4, corresponds to the existence of a common synset between two words. Here, double circles correspond to words while single circles correspond to synsets. The second one, illustrated by Figure 2.5, corresponds to the existence of an horizontal link (e.g., antonymy, similarity, also see) between a synset of each word. Finally, the third sub-category of strong relation, illustrated in Figure 2.6, corresponds to the existence of any kind of link between a synset of each word, if one word is a compound word or a phrase that includes the other one.

Finally, a medium-strength relation between two words corresponds to the existence of

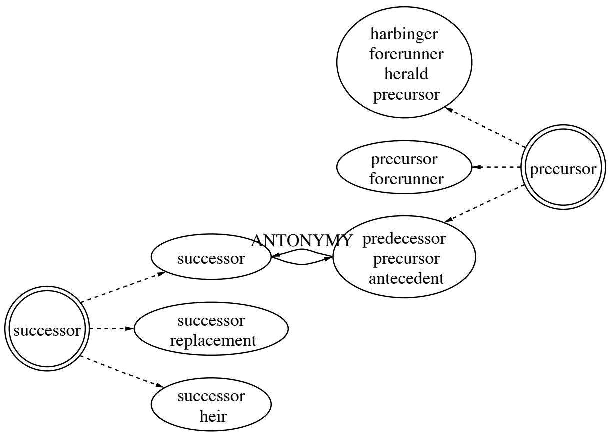


Figure 2.5: Strong Relation II

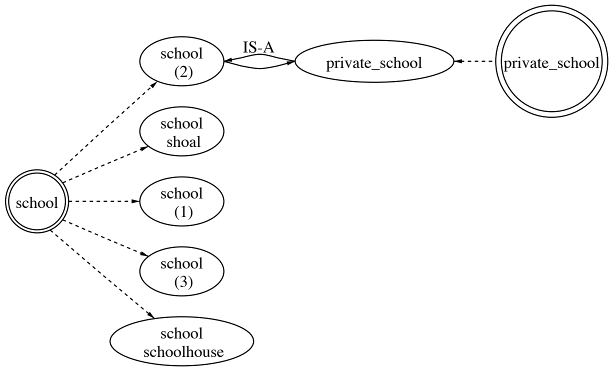


Figure 2.6: Strong Relation III

a special path between a synset of each word. Unlike extra-strong and strong relations, medium-strength relations have different weights. To compute the weight of a medium-strength relation, the shortest path between a pair of synsets is identified. If the length of this path is greater than 5 or if the path shape corresponds to one of the patterns shown in figure 2.7 (a) than its weight is null (i.e., there is no relation). Otherwise, the path shape falls into one of the classes shown in figure 2.7 (b) and the weight of the path is given by

$$weight = C - path\ length - k * number\ of\ changes\ of\ direction \qquad (2.1)$$

(where C and k are constants). Figure 2.8 provides an example of a medium-strength relation between two words.

The rationale for the patterns of Figure 2.7 is as follows: As defined earlier, links can have three possible directions: upward, downward, and horizontal. An upward direction corresponds to a generalization of the context. For instance, an upward link from {*apple*} to {*fruit*} means that {*fruit*} is a semantically more general synset than {*apple*}. Similarly, a downward link corresponds to a specification of the context. Horizontal links are less frequent than upward and downward links. A synset rarely has more than one horizontal link. Such a link corresponds to a specification of the context, usually in a very accurate way. In Figure 2.5, the horizontal link between {*successor*} and {*predecessor, precursor, antecedent*} is a very accurate specification of the meaning of the word *successor*.

While searching for a path between a synset of the source word and a synset of the target word, the semantic transition implied by each lexical relation must be taken into consideration in order to ensure semantic proximity. To ensure that the selected path will correspond to an actual relation between the source and the target word, two rules have been defined.

- (1) No other direction may precede an upward link.

Once a link that narrows down the context has been used (downward or horizontal), it is not permitted to enlarge the context by using an upward link.

- (2) No more than one change of direction is allowed.

Changes of direction constitute large semantic steps. Therefore, they must be limited. This second rule has the following exception:

- It is permitted to use an horizontal direction to make the transition from an upward to a downward direction.

Horizontal links correspond to small semantic distances like for *heat* and *hot* which are linked by an attribute relation. In this case, this exception to the second rule enables connections between subordinates of *heat* and subordinates of *hot*. Thus, I assume that enabling such a connection between two superordinates does not constitute too large a semantic step.

Since the verb file has no relation with the three other files and that adverb file has only unidirectional relations with the adjective file, I was forced to limit the chaining process to nouns. However, no grammatical parsing phase has been integrated into my algorithm in order to avoid the slowdown and the error that would have resulted. Instead, I decided to consider, as nouns, all words that could be found in the noun index as is, as well as those that could be morphologically transformed as nouns. This is based on the assumption that most words that exist as nouns, but that are used in different grammatical categories, are semantically close to their noun form (e.g., *to walk* and *a walk*). My experimentation showed that this assumption was true. However, an attempt to morphologically transform unidentified words as verbs before searching for them in the noun index introduced too much chaining inaccuracy.

2.3.2 Chain Creation and Management

Although a lexical chain may be represented as a set of words, its actual implementation is a much more complex object. Figure 2.9 gives an overview and example of a chain construction process. First, a chain object is allocated and its chain pointer, which points to a linked list of chain word objects, is initialized to *NIL* (see Figure 2.9 (i)). Then, a chain word object is allocated, initialized with the word *economy*, and pushed into the new chain (see Figure 2.9 (ii)). Now, to insert *sectors*, another chain word object is constructed and pushed into the chain's linked list. At the origin of a word insertion into a chain, there must be a relation (extra-strong, strong, or medium-strength) between the new word and a word of the chain. This relation is also stored in the chain word object. In Figure 2.9 (iii), *sectors* precedes *economy* in the chain's linked list and another form of connection, which will be described soon, illustrates its relation with *economy*. In Figure 2.9 (iv), *economic_system* is pushed into the chain's linked list, not from a relation with *sectors*, its immediate successor in the linked list, but from a relation with *economy*. Therefore, the word order in a chain's

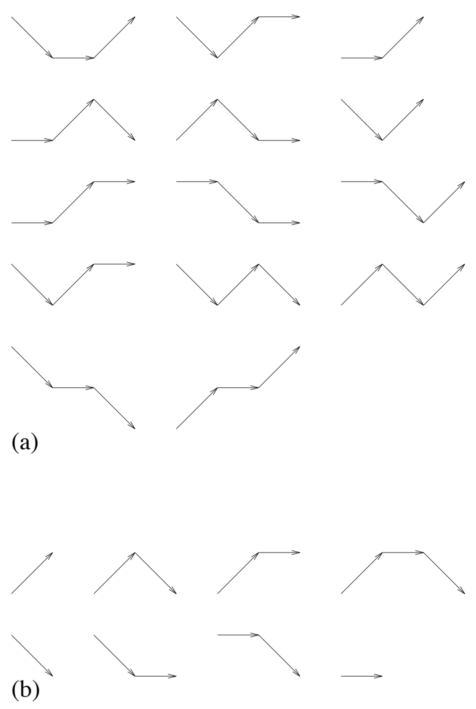


Figure 2.7: (a) Disallowed and (b) Allowed Patterns
(A vector corresponds to one or more links.)

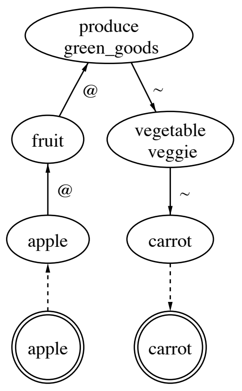


Figure 2.8: Example of Medium-Strength Relation Between Two Words
(@ = hypernymy, $\sim$ = hyponymy)

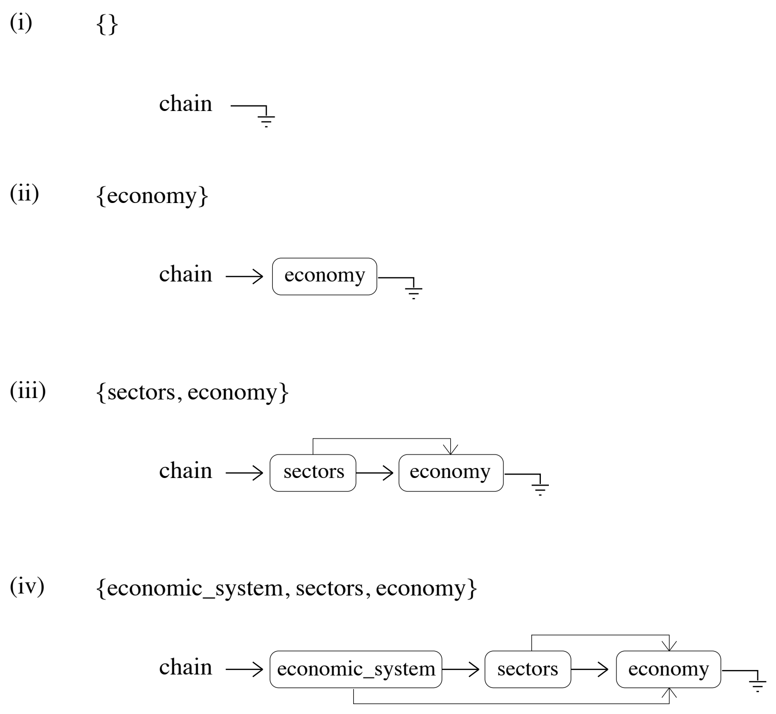


Figure 2.9: Chain Management

linked list does not necessarily correspond to relations between words, or even, as will be seen later, the word order in the input text, but only to the insertion order.

As seen earlier, a word may exist in more than one synset of WordNet, each synset corresponding to a semantic variation of the word. While constructing a new chain word object, a linked list of pointers to every synset of the word is created and attached to the object. When a word starts a new chain, all its synsets are kept since, at this point, no contextual information is available to discriminate among them (see Figure 2.10).

Pushing another word into the chain results in a word connection by linking synsets involved in the relation. When a word is inserted into. a chain with an extra-strong relation, all identical synsets are connected (see Figure 2.11). When the relation involved is strong, all pairs of strongly-related synsets are connected (see Figure 2.12). Finally, when the relation involved is medium-strength, all pair of synsets with a relation of the same weight are connected. In other words, if the largest relation weight found is x, then the weight of every combination of synsets between the two involved words is evaluated and synsets that have a relation of weight exactly equal to x are connected.

After the connection is made, the unconnected synsets of the new word object are deleted and the chain is parsed to remove the ambiguities wherever possible. Synsets that are not

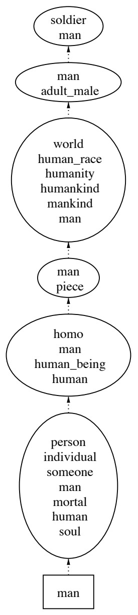


Figure 2.10: Word Starting a New Chain
(The word *man* has 6 synsets.)

implied in the current word connection are removed. Removing synsets while pushing words into a chain progressively disambiguates each word of the chain. A link from a synset of a word to a synset of another word is called an *intersynset link*. Figure 2.13 illustrates the word-sense disambiguation process resulting from the situation illustrated in Figure 2.12. This idea comes from Hirst's Polaroid Words (1988). As words are inserted into a chain, uninvolved interpretations are removed, therefore enabling stronger chaining by narrowing down the context.

While pushing *economic_system* in Figure 2.9 (iv), not only the synset list of *economic_system* and *economy* are updated but also the synset list of *sectors*. This is possible because the chain word object for *economy* contains a pointer to the chain word object for *sectors*. Therefore, each time a word is pushed into a chain, the whole chain is traversed, by following the word connections, to update the synset list of each word of the chain.

2.3.3 Stack Management

Chains are stored in a linked list. A chain object contains a pointer to the next chain object. The chain stack is implemented by a chain stack object that contains a pointer to the first chain of the linked list, the number of chains in the stack, and many stack management functions. In the stack, chains are kept in order of recency, the most recently

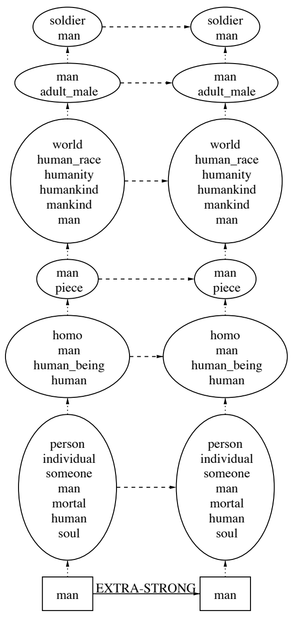


Figure 2.11: Pushing the Same Word

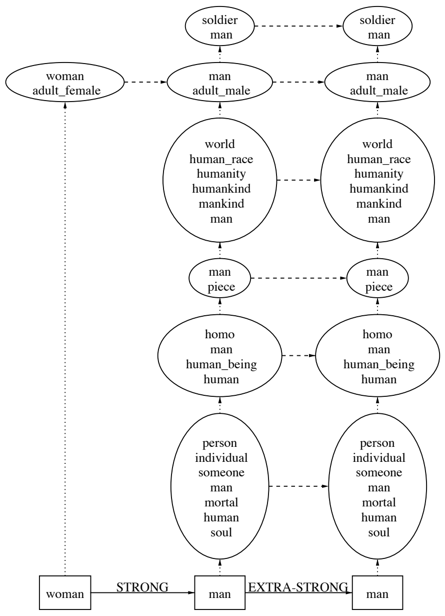


Figure 2.12: Pushing an Antonym

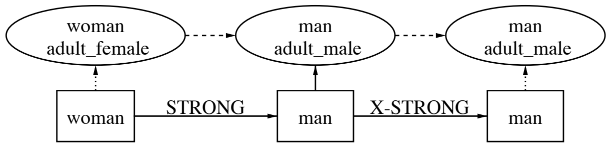


Figure 2.13: Updated Chain After Insertion

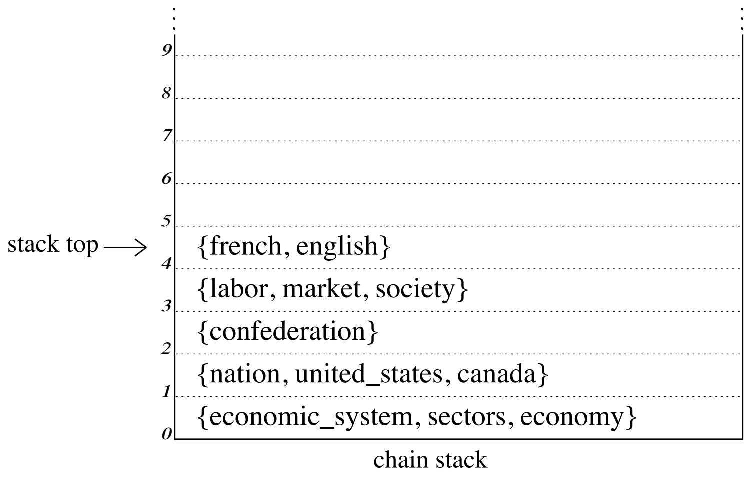


Figure 2.14: Initial Stack State

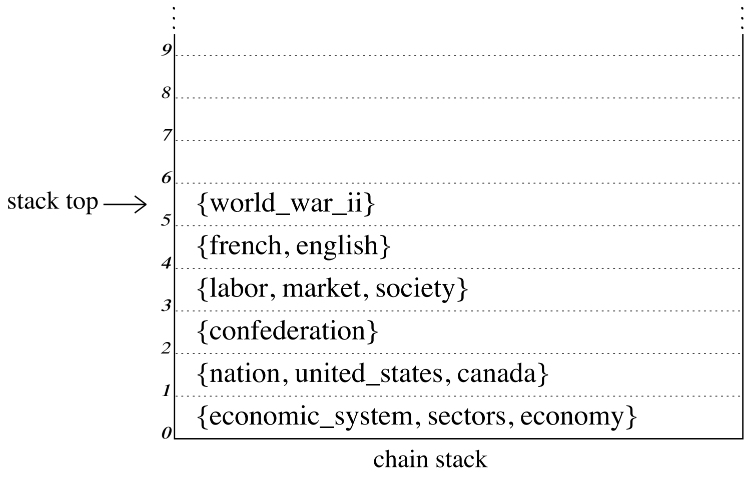


Figure 2.15: Insertion of *World War II*

created or updated chain being on top. The idea of using a stack to manage chains comes from Okumura and Honda (1994) where chains were kept in order of salience. In my implementation, recency is the equivalent of Okumura and Honda's salience.

Let us consider the stack in Figure 2.14 and the expression *World War II*. If the algorithm fails to find a chain into which to insert *World War II* than a new chain is created with this expression and pushed on top of the stack (see Figure 2.15). If the next word to be classified is *country* and the algorithm identifies {*nation, united_states, canada*} as the proper chain to insert this word, then *country* is pushed into this chain and the chain is moved on top of the stack (see Figure 2.16).

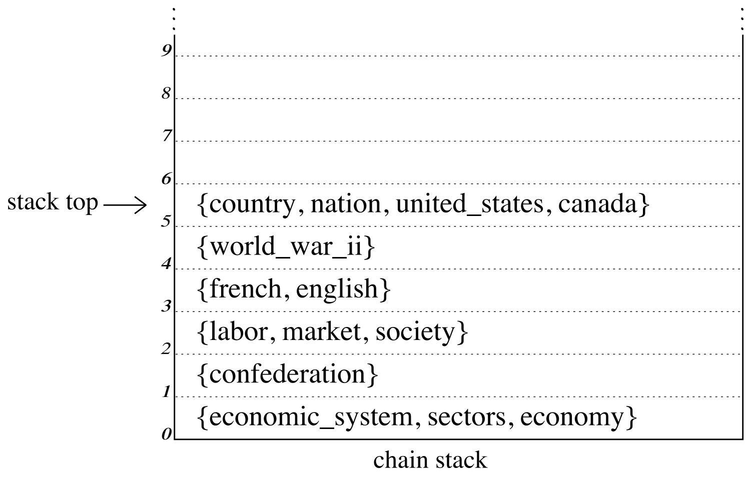


Figure 2.16: Insertion of *country*

2.3.4 Lexical Chaining

The lexical chain identification process, or *lexical chaining*, from a given input text, requires all objects described previously. Function *classify_word* in Algorithm 2.1 illustrates that process. While words are being read one by one, an attempt is made to combine them with their predecessors to form larger compound expressions that exist in WordNet. Compound words and phrases tend to correspond much better to the actual meaning than the words that comprise them, taken separately. For instance, *private school*, which is listed in the noun index as *private_school*, leads to much more accurate chaining than *private* and *school* taken separately. As one can see from figure 2.2, WordNet 1.4 contains many phrases. Concurrent with this phrase identification process, words must also pass a validity test to ensure their suitability for lexical chaining. A word (or a phrase) is valid under two conditions. First, it must not appear in the stop-word list. Secondly, it must be in the WordNet noun base as is, or morphologically transformed as a noun. The stop-word list contains closed-class words, and many vague high-frequency words that tend to weaken chains (e.g., *one, two, dozen, little, relative, right*).

When a word has passed through the phrase identification process and has been accepted as valid, it is sent to the *classify_word* function. While the current sentence has not changed (line 1) *classify_word* pushes the word into a queue (line 2). At the end of each sentence, the queue is processed (lines 3–28).

First, for each word of the queue, an extra-strong relation is sought through the whole chain stack, from top to bottom. Since all extra-strong relations have the same weight, a

Algorithm 2.1 classify_word(new_word)

```
▷ MS_R = Medium-Strength Relation
▷ S_R = Strong Relation
▷ XS_R = Extra Strong Relation
if (new_word.sentence_number = current_sentence_number) then
    queue.push(new_word);
else
    current_sentence_number = new_word.sentence_number;
    for current_word from queue.first to queue.last do
        if (chain_stack.try_to_chain(current_word, XS_R)) then
            queue.remove(current_word);
        end if
    end do
    for current_word from queue.first to queue.last do
        if (chain_stack.try_to_chain(current_word, XS_R)) or
            (chain_stack.try_to_chain(current_word, S_R)) then
                queue.remove(current_word);
        end if
    end do
    for current_word from queue.first to queue.last do
        if (chain_stack.try_to_chain(current_word, XS_R)) or
            (chain_stack.try_to_chain(current_word, S_R)) or
            (chain_stack.try_to_chain(current_word, MS_R)) then
                queue.remove(current_word);
        end if
    end do
    for current_word from queue.first to queue.last do
        chain_stack.create_chain(current_word);
        queue.remove(current_word);
    end do
    queue.push(new_word);
end if
```

search ends as soon as one is found. If the search is successful, the word is added to the chain where the extra-strong relation has been found and then removed from the queue. As seen previously, that chain is moved to the top of the chain stack. Words that cannot be classified with an extra-strong connection remain in the queue (lines 5–7).

Then, strong relations are sought with the remaining words of the queue (lines 10–13). At this point, there might exist an extra-strong relation between two words that are still in the queue. If the algorithm did not take this possibility into consideration, then there could be a situation where the former word is classified with a strong relation in a given chain while the latter word is classified with a strong relation in another chain, thereby violating the priority is the extra-strong relation. To avoid this problem, before seeking a strong relation for a word of the queue, an extra-strong relation is sought in the whole chain stack (line 11 and 12). Therefore, if there are two words in the queue that are related together by an extra-strong relation and the former word is inserted somewhere in the stack by a strong relation, the extra-strong relation search performed before proceeding to the strong relation search of the latter word will enable the linking of both words. Even if it adds several redundant searches, this solution is efficient because the extra-strong relation search process is so fast that there is almost no speed reduction.

When no extra-strong relation is found for a given word, a strong relation is sought in the chain stack. However, for strong relations, the search scope is limited to a *word-chain distance* of 7 sentences. The word-chain distance is defined as the difference between the sentence number of the current word and the sentence number of closest word (the latest word in this case) of the current chain. Since all strong relations also have the same weight, a search ends as soon as a strong relation is found. Here again, words that cannot be classified with a strong connection remain in the queue.

Finally, medium-strength relations are sought with the remaining words of the queue. The attempt to classify the remaining words of the queue with medium-strength relations is shown in lines 16–20. In a fashion similar to the strong relation seek process, for each word of the queue, an extra-strong and a strong relation are sought before seeking a medium-strength relation. For medium-strength relations, the search scope is limited to a word-chain distance of 3. However, since the weight of medium-strength connections varies, all medium-strength connections within the search scope must be found, in order to retain the one with the highest weight.

Both the extra-strong and the strong relation search are very fast processes. However, the medium-strength relation search is the most expensive operation of the whole lexical chaining process in terms of CPU time.

At the end of the classification process, a new chain is created for each word that remains in the queue. These chains are then pushed into the chain stack.

2.4 Analysis of Results

2.4.1 First Analysis

When tested on different input texts, my lexical chaining program identified chains that were sometimes very good and sometimes disappointing. The text studied in this section is the quotation from Einstein given at the beginning of Chapter 1. The resulting chains are the following:

```
[009]    embankment(3)


[004]    given(3), constant(0)


[001]    simultaneity(3), respect_to(3), train(3), train(2),
         reference_to(1), reference(1), advantage(1), train(1),
         train(1), train(1), direction(0), velocity(0),
         train(0)


[008]    definition(3)


[006]    point(2), particular(2), regard(1)


[007]    place(2), place(2), event(2), events(1)


[003]    line(2), rails(0)


[005]    body(1), people(1), figure(0)
```

```
[002]   travelling(1), travelling(0)
```

Numbers between square brackets at the beginning of each chain indicate the chain creation order, [001] corresponding to the first chain inserted in the stack. Chains are displayed according their order of occurrence in the stack, the first one being at the top of the stack. Numbers between parenthesis beside words indicate the word sentence number, 0 corresponding to the first sentence. In each chain, words appear in reverse order of insertion, the last word being the first one inserted.

Here, the first chain has been created with the word *train*, which has 6 senses in WordNet. Later on, *rails* is read but is not associated with *train*, the distance between these two words being too large in WordNet.

Another problem is encountered when *velocity* is connected to *train* by a medium-strength relation. In fact, this has the undesired effect of wrongly disambiguating *train* by selecting the following sense:

> sequence, succession, sequel, train – events that are ordered in time; "in chronological sequence".

Later on, *direction* is connected to *train* by the following medium-strength relation:

> {direction, directionality}
>
> *is a*
>
> {spatial relation}
>
> *is a*
>
> {relation}
>
> *is a*
>
> {temporal relation}
>
> *includes*
>
> {sequence, succession, sequel, train}

The second chain is simply a repetition of the word *travelling* and is, therefore, trivial. The third chain is an example of successful word-sense disambiguation. It has been created with *rails* which has this unique meaning:

> track, rail, rails – a bar or bars of rolled steel making a track along which vehicles can roll.

When *line*, which has 25 senses, has been connected to *train*, the following sense was used:

line, railway line, rail line – railroad track and roadbed.

The medium-strength relation involved is the following:

{line, railway line, rail line}

has part

{railroad track, railway}

has part

{track, rail, rails}

Thus, in this case, WordNet's relations set is sufficient to relate *line* and *rails*.

Chain 5 is another example where words that are not semantically related in the text are put together by using different senses, thereby producing a wrong disambiguation. First, the insertion of *people* forces the sense of *figure* to be the following:

human body, physical body, material body, soma, build, figure, physique, anatomy, shape, bod, chassis, frame, form – alternative names for the body of a human being.

Then, the insertion of *body* can be done by using the following inappropriate sense:

body – people associated by some common tie or occupation.

In this case, the problem of wrong sense disambiguation is hard to avoid because the three synsets respectively involved are close to each other in terms of distance in the graph:

{people}

has member

{person, individual, someone, man, mortal, human, soul}

has part

{human body, physical body, material body, soma, build, figure, physique, anatomy, shape, bod, chassis, frame, form}

{body}

is a

{<u>people</u>}

But small texts do not fairly reflect the performance of my lexical chainer. On one hand, words located at the beginning of a text are disadvantaged by a lack of contextual information. On another hand, words located at the end of a text are disadvantaged by a lack of successors that could disambiguate them further more.

2.4.2 Second Analysis

A much larger input text is now considered. It appears in Appendix A, followed by the resulting lexical chains.

By looking at the output, one can notice that some chains are made mostly by the repetition of a word:

```
[004]   classroom(65), classroom(63), school(62), school(62),
        schools(60), school(60), school(58), school(57),
        schools(57), school(57), school(55), schools(54),
        schools(52), schools(52), kindergarten(50),
        school(49), school(49), school(48), schools(47),
        high(46), school(45), schools(45), high(44),
        school(44), school(44), school(42), school(39),
        school(35), schools(29), school(27), schools(21),
        public_schools(17), schools(15), school(11),
        school(9), public_school(5), elementary_school(4),
        private_school(0), public_school(0)
```

Other chains, like the following one, seem quite coherent:

```
[015]   science(62), philosophy(48), english(39), math(30),
        science(30), philosophy(6), geography(2), science(2),
        english(2), mathematics(2), science(2), geography(2)
```

However, two major problems can be observed. On one hand, many "chains" contain only a single word, especially at the end of the stack. This under-connection reflects an inability to find relations between these words and other words in the text. On the other hand, at the beginning of the stack, some chains seem to be the result of an over-connection.

A failure to connect a word might be explained by one of these four reasons:

1. the word is a figure of speech (*e.g.*, a metaphor)
2. an inadequacy of WordNet's set of relations
3. a lack of consistency in the semantic proximity expressed by WordNet's links
4. a lack of connections in WordNet
5. a poor algorithm for chaining

The third sentence illustrates the first case:

> She opposes the curriculum prescribed by the Ontario government that bundles subjects such as history, geography, science and mathematics into an integrated program – a <u>cornucopia</u> of intersubject learning in which a thematic exploration of autumn, for example, might touch on English, French, art, science and geography.

Here, *cornucopia* is used as a metaphor. Since the actual meaning of *cornucopia* has nothing to do with the context, it is perfectly reasonable that this word is not chained to any other word.

The following sentence gives an example of the second problem.

> School administrators say these same taxpayers expect the schools to provide <u>child care</u> and school lunches, to integrate immigrants into the community, to offer special classes for adult students, to present sports, music and vocational programs, to instill a set of values, to teach conflict resolution and AIDS prevention — all without sacrificing the 3Rs.

Here, one would want *child care* to be semantically related to *school*. However, WordNet does not have a sufficient set of relations to relate these two words. In fact, relations such as antonymy, holonymy, or meronymy are not appropriate to link *child care* to *school*. The relation that exists between these two words is rather a *situational relation*. In other words, *child care* could be related to *school* because children go to school and school cares for them.

The second sentence illustrates the third problem:

The cost means no holiday trips and more stew than steak, but she is satisfied that her children, now in Grades 3 and 4, are being properly taught.

Here, *stew* and *steak* are obviously somehow related. However, according to the output, these two words have not been linked to each other (or any other words). Here is their mutual relation in WordNet:

{stew}

is a

{dish}

is a

{aliment}

includes

{meat}

includes

{cut, cut of meat}

includes

{piece, slice}

includes

{steak}

The inter-synset distance between *stew* and *steak* is 6 synsets, which is greater than the limit of 5 that is set in the lexical chainer. In general, the greater the distance limit, the greater the number of weak connections. However, links in WordNet do not all reflect the same *semantic distance*. In other words, there are situations, as with *stew* and *steak*, where words have an obvious semantic proximity but are distant in WordNet's graph.

There are also situations were words are close to each other in WordNet's graph while being quite distant semantically. This introduces the problem of over-chaining. Chain [028] is an example of over-chaining. It has been created with *pathologists*. Right after, *professionals* has been inserted into it. Then, *taxpayers* has been linked to *professionals*. However, *public* has then been inserted in the chain by the following relation with *professionals*:

{populace, public, world}

is a

{people}

has member

{person, individual, someone, man, mortal, human, soul}

includes

{adult}

includes

{<u>professional</u>, professional person}

This connection results in the merging of two chains in one. On one hand, *teacher* is connected to *professionals*, *educator* is connected to *teacher*, *principal* is connected to *educator*, ... On the other hand, *group* is connected to *public*, *provincial* and *national* are connected to *group*, ...

The *stew* and *steak* problem could also be classified as an example of the fourth problem. In fact, *stew* is linked to 16 examples of stew: *Brunswick stew*, *olla podrida*, *Hungarian goulash*, *beef stew*, etc. However, none of these synsets has a substance holonym. Such links could have set a shorter path between *stew* and *steak*.

Among the four problems regarding failures to connect, the first one is not a fault. It is perfectly acceptable that a word which has a special use (*e.g.*, a metaphor) cannot be connected to any other word.

Despite the fact that WordNet contains many semantic relations, the second problem corresponds to the inadequacy of the set of relations to link some words that are obviously semantically related. However, many semantic relations are hard to classify (*e.g.*, the situational relation between *physician* and *hospital*). These relations are often expressed, without being labeled, in categories of a thesaurus. Thus, one way to improve WordNet would be to use a thesaurus to add new relations that could simply be labeled as *thesaural relations*. Nevertheless, the lack of correspondence between WordNet's synsets and thesaural categories make this solution hard to apply.

The third problem is not just WordNet's problem. In fact, it is reasonable to find more concepts in some subjects and therefore a higher density of synsets. This problem could be solved if each link in WordNet had some kind of scale factor quantifying the semantic distance corresponding to the link. However, the semantic proximity of two synsets is rather hard to quantify.

Finally, the fourth problem might be partly solved if more connections are made in a future version of WordNet.

Furthermore, the wrong disambiguation problem is hard to avoid, especially when the weight of the connections involved is large. If it is not the case, solving the third and the fourth problem could to find a connection with the appropriate sense that has a larger weight than the one involved with the wrong sense.

2.5 Similar Research

In his master's thesis, *Lexical Chains, WordNet and Information Retrieval*, Mark Stairmand (1994) proposes a way to identify lexical chains for information retrieval purposes. He first decided to implement Morris and Hirst's algorithm by using the 1911 on-line version of Roget's Thesaurus. He concluded that this thesaurus was an inappropriate knowledge source for three reasons: (*i*) it suffers from a lack of modern vocabulary and (*ii*) many typographical errors can be found.

Therefore, Stairmand decided to develop his own lexical chainer based on the use of WordNet. His algorithm includes five steps:

1. A tagger is used to identify and assign a part-of-speech label to nouns, adjectives, and verbs of the input text. Words belonging to any of these three categories are stored on disk with their respective label.

2. A program written in C reads the input text and assigns a paragraph number to each word selected at the previous step.

3. For each of the selected words, a list of all related words in WordNet is generated and output to disk. This process is called the *expansion of a term* and corresponds to finding synonyms, subordinate terms, terms with a common immediate superordinate term, and meronyms by part or by member.

4. The expansion terms found in step 3 are then used to find every possible link among the candidate words. The results are stored into a file.

5. The file resulting from the previous step is then sorted to find words that are linked together. These groups of words form the chains.

In addition to these five steps, the density of each chain is calculated. High-density chains are retained for information retrieval queries while low-density chains are rejected. Some chains, called *spurious*, are semantically useless either because the head term is too general or because some terms are used in a different context than their context in the text.

To calculate the density of a chain, its words are considered in their order of appearance in the text. This computation involves three steps:

1. The sum of the distance between successive paragraphs is calculated for all paragraphs involved in the chain.

2. The sum of the distance between successive paragraphs is calculated for all words of the chain. This is different from step 1 since the possibility that there might be more than one word per paragraph is taken into account. In other words, if two words are in the same paragraph then the distance with the previous paragraph is computed once in step 1 and twice in step 2.

3. The results of step 1 and step 2 are averaged and divided by the number of elements in the chain minus one.

The lexical chainer developed by Stairmand is oriented toward information retrieval purposes. Its goal is not to chain as many words as possible but to identify chains that could reflect the structure of a text as proposed by Morris and Hirst (1991). Unlike my lexical chainer, Stairmand's chainer deals with verbs and adjectives. He uses the *pertains to* relation to link adjectives and nouns (like *Jewish* and *Jew*).

Nevertheless, Stairmand's chainer does not deal with compound words and expressions that are in WordNet. Furthermore, searches for relations between words are limited to an inter-synset distance of one, except for the search of terms with common super-ordinate (inter-synset distance of two).

Chapter 3

Automatic Detection of Malapropisms

A *malapropism* is the confounding of one word with another word of similar sound and/or similar spelling that has a quite different meaning, *e.g.*, *an ingenuous (*for *ingenious) machine for peeling oranges.* In this example, there is a one-letter difference between the malapropism and the correct word. Ignorance or simply a typing mistake might cause such errors. However, since *ingenuous* is a correctly spelled word, traditional spelling checkers cannot detect this kind of mistake. The aim of this chapter is to propose a malapropism detection algorithm based on the identification of lexical chains.

3.1 Overview of Spelling Error Detection and Correction

There exist two kinds of spelling errors: non-word and real-word errors. A non-word spelling error is a word that is not correctly spelled (*e.g., swi*m*ing* instead of *swi*mm*ing*). A real-word spelling error is a word that is correctly spelled as the result of a spelling error on a different word that was intended (*e.g.,* a*uspicious* instead of s*uspicious*).

3.1.1 Non-word Error Detection and Correction

The two main techniques for non-word error detection are lexicon lookup and n-gram analysis. In the former, each word of the input text is sought in a lexicon and considered to be an error if not found. The size of the lexicon is an important issue: an insufficient lexicon will result in too many false rejections while too-large a lexicon (with rare or unusual words) will result in too many false acceptances. False rejections might also be caused by the use of a lexicon that is not adapted to a specific area of application. The second detection method, called the n-gram analysis, is based on the probability of a given sequence of letters of length n, usually two (digram) or three (trigram). Under a certain threshold, an error is signaled.

In a survey paper on correction techniques, Karen Kukich (1992) classifies replacement techniques for error correction into six groups:

1. The minimum edit distance technique (Wagner, 1974) selects, in a lexicon, words that require fewer transformations, in terms of basic operations such as insertion or deletion, to get to the misspelled word.

2. The similarity key technique uses sets of similarly spelled strings to generate replacement candidates and then uses a lexicon to verify the validity of each of candidate.

3. The rule-based technique uses a set of rules for transforming misspelled strings and then filters out the improper suggestions with a lexicon.

4. The n-gram technique uses statistics on the probability of a given sequence of characters to replace misspelled strings. Here again, a lexicon filters out the improper replacement suggestions.

5. The probabilistic technique is based on the probability that a given character will be followed by (transition probabilities) or mistaken (confusion probabilities) for another given character to generate replacement candidates.

6. Neural network techniques.

3.1.2 Real-word Error Handling

Real-word errors are much more difficult to detect than non-word errors and even more difficult to correct. Kukich (1992) classifies real-word errors into four categories:

1. syntactic errors (*e.g.*, *The students are doing* **there** *homework.*);

2. semantic errors or malapropisms (*e.g.*, *He spent his summer travelling around the* **word**.);

3. structural errors (*e.g., I need* **three** *ingredients: red wine, sugar, cinnamon, and cloves.*);

4. pragmatic errors (*e.g., He studies at the University of Toronto in* **England** *and she studies at Cambridge.*).

Errors that belong to the first category can be detected by using a natural-language parser or by performing n-gram analysis to detect words that have a low probability of succession. The same tools could be used to suggest replacements. However, errors that belong to one of the three other categories are much harder to detect and even harder to correct.

Few studies have been made on the frequency of real-word errors. Mitton (1987) studied 925 essays written by high-school students and found that 40 percent of all errors were real-word errors. He noticed that most of these real-word errors belonged to the first category. Atwell and Elliot (1987) with the University of Lancaster Unit for Computer Research on the English Language (UCREL) (Garside, Leech and Sampson, 1987) analyzed three different kinds of text that had not previously been automatically proofread: published texts, 11- and 12-year-old students' essays, and text written by non-native English speakers. They found that the corresponding amount of real-word errors were 48%, 64%, and 96% respectively. Among the real-word errors, 25%, 16%, and 38% respectively belonged to the semantic category.

This chapter presents an algorithm that attempts to detect semantic errors (malapropisms) and to suggest correction replacements.

3.2 Algorithm for Detecting Likely Malapropisms

As seen in Chapter 1, each discourse unit of a text tends to use related words (Morris and Hirst, 1991). Therefore, in our semantic universe of words (see Section 2.2), discourse units would tend to form clusters of words. My hypothesis is that the more distant a word is from any word cluster of a text, the higher the probability that this word is a malapropism. In other words, if a word does not seem to belong to any word cluster, then this word has a higher probability than the other words of being used improperly.

In my algorithm, I use lexical chains as an implementation of word clusters and consider words that cannot be inserted in any chain as potential malapropisms. Therefore, my first basic hypothesis is the following:

> A word that is valid for chain insertion but cannot be inserted in a chain with any other words has a much higher probability of being a malapropism than a word that can be inserted into a chain.

Such a word will be called a potential malapropism.

Assuming that I have a tool that produces a set of replacement words for which an initial word was a plausible mistyping, my second basic hypothesis is the following:

> Among potential malapropisms, actual malapropisms have a higher probability of having a replacement candidate that can be inserted in a chain with other words than non-malapropisms.

These two hypothesis are the basis of the following algorithm devised by Hirst (personal communication):

> It is assumed that (as in all but the simplest spelling checkers) there is already in place in the program a mechanism that, given a character string w, can produce a set $P(w)$ of all words in the program's lexicon for which w is a plausible mistyping.
>
> Such programs may then detect real-word errors by incorporating the following technique.
>
> 1. The program filters out (that is, ignores) 'stop words': high-frequency words of low semantic content, such as closed-class words, and vague common verbs such as *do, thing*, and so on. (The erroneous occurrence of these words cannot be detected by this method.)
> 2. The program then constructs lexical chains between the remaining words in the text, using any method, such as one of those described below. Words that appear in the same chain have a close semantic relation to one another.
> 3. The program then hypothesizes that a word w is in error, even though it is a correctly spelled word, if w is not a member of any of the lexical chains, but there is a word $w' \in P(w)$ that would be in a lexical chain were it to have appeared in the text instead of w. It is then likely enough that w' was

intended where w appears that the user should be alerted to the possibility of the error.

(It is not essential that the spelling checker deals with *in*correctly spelled words before applying this method; rather, they may simply be ignored. However, correcting them first will increase the accuracy of the technique.)

3.3 Experiment

One way of testing my algorithm is to insert malapropisms in a text and see what proportion of them are identified as potential malapropisms. This constitutes the first detection stage. Then, for each potential malapropism, a set of replacement candidates is established. Whenever a potential malapropism has a replacement candidate that can be inserted into a chain with other words, it is likely enough that this replacement candidate was intended instead of the potential malapropism. If this potential malapropism does not correspond to an actual malapropism, it is called a *false alarm*. Otherwise, it is considered as a detected malapropism and the corresponding replacement candidate is considered as a proper correction. This alarm generation phase constitutes the second detection stage.

3.3.1 First Detection Stage

As illustrated by Algorithm 3.1, the experiment for the first detection stage is divided into four steps. First, a given input text is read and malapropisms are inserted to create a new text for the experiment (line 1). These malapropisms are also stored on disk in a relation called GENERATION that has four entities: *file_name*, *sentence_number*, *original_word*, and *malapropism*.

In the second step, the lexical chains of the modified text are identified by using the algorithm described in Chapter 2 (line 2). These chains are stored in a chain stack. At this point, the chain stack also contains potential malapropisms that are in atomic chains[1].

In the third step, atomic chains are extracted from the chain stack and stored in a list of potential malapropisms (line 3). This is in accordance with my hypothesis that words

[1]Since a *chain* is, by definition, a series, the concept of an *atomic chain* may seem awkward. However, one must understand that, in this case, *chain* is not intended to refer to a linguistic entity but only to the implementation of a lexical chain.

Algorithm 3.1 Experiment

```
1   generate_malap(original_text_file, modified_text_file,);
2   identify_chains(modified_text_file, chain_stack);
3   chain_stack.extract_atomic_chains(potential_malap_list);
4   output_potential_malap(original_text_file, potential_malap_list);
```

that cannot be inserted into any chain have higher probability of being malapropisms.

Finally, the list of potential malapropisms is stored on disk in a relation called POTENTIAL_MALAP (line 4). This relation has three entities: *file_name*, *sentence_number*, and *guess*.

Algorithm 3.2 inserts malapropisms into a given input text and stores the new text on disk. To insert malapropisms, the input text is read from disk, word by word (line 2). When the value of the word counter is a multiple of 200 (line 3), the current word is selected for replacement. If the current word is not valid for replacement, the following words are considered, one by one, until a valid one is found. A word is valid for replacement under three conditions. First, the word must not be in the stop-word list. Secondly, the word or its stem must be in the noun database. Finally, a replacement word must be found. If all these conditions are satisfied, the current word is replaced by a malapropism and is output to disk in the new text file (line 18). The GENERATION relation is also updated (line 19).

Algorithm 3.3 presents the *create_malap* function. First, a list of valid replacements for the given word is generated (line 1) by calling the *suggest_replacement* function. If the resulting list is not empty (line 2) one word is randomly selected (line 3) and returned as a malapropism.

Algorithm 3.4 describes the *suggest_replacement* function. First, spelling replacements for the input word are generated (line 1) by using code adapted from ispell[2] 1.123. Then, these spelling replacements, which are kept in a replacement queue, are popped one by one (line 2) and pushed into a malapropism list (line 8) if they satisfy a validity test (lines 3–7). A spelling replacement is a valid malapropism if it satisfies four criteria. First, it must be different from the original word (line 3). Secondly, it must not be in the stop-word list (line 4). Thirdly, it must be in the noun database (line 5). Finally, it must not be a morphological derivation of the original word (line 6 and 7).

[2]See Section 3.4 for an overview of the ispell word replacement algorithm.

Algorithm 3.2 generate_malap(input_file, output_file)

```
word_count = 1;
while (input_file.read_word(current_word)) do
    if ((word_count mod 200) = 0) then
        modification_mode = TRUE;
        while (modification_mode) do
            if not(stop_word_list.includes(current_word)) then
                find_stem(current_word, current_word_stem);
                if ((WordNet_noun.includes(current_word) or
                    WordNet_noun.includes(current_word_stem)) and
                    (create_malap(current_word_stem, malap))) then
                        modification_mode = FALSE;
                end if
            end if
            if (modification_mode) then
                output_file.write_word(current_word);
                input_file.read_word(current_word);
            else
                output_file.write_word(malap);
                GENERATION.append(input_file, current_word, malap);
            end if
            word_count = word_count + 1;
        end do
    else
        output_file.write_word(current_word);
        word_count = word_count + 1;
    end if
end do
```

Algorithm 3.3 create_malap(original_word, malap)

```
suggest_replacement(original_word, replacement_list);
if (replacement_list.count $>$ 0) then
    malap = replacement_list[random() mod replacement_list.count];
end if
return(replacement_list.count);
```

Algorithm 3.4 suggest_replacement(original_word, malap_list)

```
compute_spelling_replacement(original_word, replacement_queue);
while (replacement_queue.pop(current_suggestion)) do
    if ((current_suggestion $\neq$ original_word) and
        (not(stop_word_list.includes(current_suggestion))) and
        (WordNet_noun.includes(current_suggestion)) and
        ((!find_stem(current_suggestion, stem)) or
        (stem $\neq$ original_word)) ) then
            malap_list.push(current_suggestion);
    end if
end do
```

3.3.2 Second Detection Stage

In the second detection stage, a set of replacement candidates is found for each potential malapropism. Then, an attempt is made to find a relation between any of these replacement candidates and a word of the chain stack. If more than one replacement candidate has a relation with a word of the chain stack, all of them are retained. This chaining process is done by using the normal chaining mechanism except that the word-chain search scope is both backward and forward and limited to the same word-chain distance in both directions.

If a potential malapropism has a chainable replacement candidate, I shall say that an alarm is raised. Each alarm is stored in a relation called ALARM that has four fields: *file_name*, *sentence_number*, *guess*, and *correction*. If this alarm does not correspond to one of the malapropisms that have been inserted initially, it is called a false alarm. Otherwise, it is called a true-alarm, or detected malapropism. However, this detected malapropism might or might not correspond to the original word (the "intended" word).

3.3.3 Performance Measurement

In a highly redundant, connected text, all atomic chains would contain malapropisms and no non-atomic chain would contain malapropisms. This can be expressed as follows:

$$(card(chain_y) = 1) \Rightarrow (\exists x | malapropism_x \in chain_y) \tag{3.1}$$

where $card()$ returns the cardinality of a given chain.

However, my first basic hypothesis does not state that malapropisms will satisfy this equation but only that they have a higher probability of being in an atomic chain than the other chained words. Therefore, I should expect that some atomic chains will not contain a malapropism and that some malapropisms will be in non-atomic chains. This is expressed as follows:

$$\exists x, y | (card(chain_y) > 1) \wedge (malapropism_x \in chain_y) \tag{3.2}$$

$$\exists y \ \nexists x | (card(chain_y) = 1) \wedge (malapropism_x \in chain_y) \tag{3.3}$$

As expressed before, whenever a word is in atomic chain, it is considered as a potential malapropism. According to Equation 3.2 and 3.3, some potential malapropisms will correspond to actual malapropisms and some will not. These two kinds of potential malapropisms are called correct and incorrect guesses respectively. The performance of my algorithm is evaluated as follows:

$$performance_1 = \frac{(\frac{number\ \ of\ \ correct\ \ guesses}{number\ \ of\ \ malapropisms})}{(\frac{number\ \ of\ \ incorrect\ \ guesses}{number\ \ of\ \ non-malapropisms})} \tag{3.4}$$

If my first basic hypothesis is true, then

$$performance_1 > 1 \tag{3.5}$$

A performance of 2 would mean that malapropisms have twice as much chance of being in atomic chains than non-malapropisms.

The number of malapropisms is obtained by performing the following SQL query:

```
select count(*)
from GENERATION
```

The following query returns the number of correct guesses:

```
select count(*)
from GENERATION, POTENTIAL_MALAP
where GENERATION.file_name = POTENTIAL_MALAP.file_name
    and GENERATION.sentence_number = POTENTIAL_MALAP.sentence_number
    and GENERATION.malap = POTENTIAL_MALAP.guess
```

The number of incorrect guesses is obtained by subtracting the number of correct guesses from the number of tuples in POTENTIAL_MALAP. The number of non-malapropisms is calculated by subtracting the number of malapropisms from the number of words in the chain stack.

To test my second basic hypothesis, I need to verify whether correct guesses have a higher probability of becoming alarms than incorrect guesses. Here, the performance of my algorithm is evaluated as follows:

$$performance_2 = \frac{(\frac{number\ \ of\ \ detected\ \ malapropisms}{number\ \ of\ \ correct\ \ guesses})}{(\frac{number\ \ of\ \ false-alarms}{number\ \ of\ \ incorrect\ \ guesses})} \tag{3.6}$$

The number of detected malapropisms is given by the following SQL query:

```
select count(distinct g.file_name, g.sentence_number, g.malap)
from GENERATION g, ALARM
where GENERATION.file_name = ALARM.file_name
    and GENERATION.sentence_number = ALARM.sentence_number
    and GENERATION.malap = ALARM.guess
```

The number of false-alarms can be obtained by subtracting the number of detected malapropisms from the number of alarms. The number of alarms is given by the following query:

```
select count(distinct file_name, sentence_number, guess)
from ALARM
```

3.3.4 Overall Performance

The overall detection performance of the algorithm can be obtained by multiplying the performance of the first detection stage by the performance of the second detection stage:

$$performance_O = performance_1 \times performance_2 \tag{3.7}$$

This corresponds to the probability of a malapropism of becoming an alarm over the probability of a non-malapropism of becoming an alarm.

The correction performance of the algorithm is given by the proportion of detected malapropisms that have the original word among their correction suggestions. The number of these detected malapropisms can be found by the following query:

```
select count(*)
from GENERATION, ALARM
where GENERATION.file_name = ALARM.file_name
    and GENERATION.sentence_number = ALARM.sentence_number
    and GENERATION.malap = ALARM.guess
    and GENERATION.original_word = ALARM.correction
```

3.4 Ispell Word-Replacement Algorithm

To understand the nature of the malapropisms used in this experiment, a brief overview of the basis of the malapropism generator, ispell, will be given. Written by Geoff Kuenning, ispell is a spelling checker for non-word errors that uses a dictionary to find misspelled words. Whenever a word is not found in its dictionary, ispell looks in its lexicon for *near misses* among the six following categories:

1. a missing letter *e.g.*, fla**m**able → fla**mm**able;
2. an extra letter *e.g.*, ca**ll**ibrate → ca**l**ibrate;
3. a pair of transposed letters *e.g.*, retr**ei**ve → retr**ie**ve;
4. a letter substitution *e.g.*, Gre**a**k → Gre**e**k;
5. a missing space *e.g.*, filterout → filter out;
6. a missing hyphen *e.g.*, threedimensional → three-dimensional.

3.5 Analysis of Results

To test my algorithm, I used 500 articles on many different topics selected randomly from the *Wall Street Journal* between 1987 and 1989. Whenever an atomic chain was made of a compound word (*e.g.*, *black-and-white*) or a phrase (*e.g.*, *elementary school*), it was not considered as a potential malapropism. The probability that the algorithm puts two words together to form a compound word or a phrase is so low that such a merging can be considered as a special kind of chaining.

As seen in Subsection 3.3.1, malapropisms were created by using an algorithm that generates replacement suggestions for a spelling checker. This algorithm returns words that differ only slightly (usually by a single letter or a transposition) from the original one. Applied to malapropism creation, this technique has the disadvantage of sometimes generating a new word that is semantically very close to the original one (*e.g.*, *billion* → *million*, *unit* → *unity*). These new words are not actual malapropisms but are considered to be malapropisms in the computation of the results. Fortunately, such replacements are rare.

Table 3.1 displays a sample of the GENERATION relation created during the experiment while Tables 3.2 and 3.3 give a sample of the POTENTIAL_MALAP and ALARM relation respectively.

Table 3.1 shows that *today*, *crash*, and *tender* were substituted by the malapropisms *toady*, *crush*, and *tenter* respectively. The corresponding original sentences are the following:

1. "<u>Today</u> all anyone needs is basic data on a small company ...and a Lotus spread sheet to do most of the research", says Scott Emerich, chairman of Market Guide, a Glen Head, N.Y.-based OTC information service.

2. But most of yesterday's popular issues were small out-of-the-limelight technology companies that slipped in price a bit last year after the <u>crash</u>, although their earnings are on the rise.

3. Among the largest OTC issues, Farmers Group, which expects B.A.T Industries to launch a hostile <u>tender</u> offer for it, jumped 2 3/8 to 62 yesterday.

Table 3.2 shows that both *toady* and *crush* have been detected as potential malapropisms as well as *basic* and *treasure*, which are not actual malapropisms. However, *tenter* was not detected but was rather inserted into the following chain:

```
[003]   tenter(28), stocks(24), stock(19), brush(17), limelight(17),
        stock(15), shoe(12), stock(11), stock(7), stocks(5),
        stocks(1), stocks(0), stocks(0)
```

The relation implied to insert *tenter* into this chain is the following:

{<u>tenter</u>}

is a

{framework, frame}

includes

{handbarrow}

has part

{handle, grip, hold}

includes

{<u>stock</u>}

Table 3.3 shows that all potential malapropisms of Table 3.2 resulted in an alarm. The correction sugestion for *toady* is the original word, *today*, which was inserted in the following chain:

```
[004]   yesterday(28), million(25), million(25), market_value(25),
        ...
        turnarounds(11), today(10), spread(8), needs(8), today(8),
        things(7), single(7), month(7), past(7), purchases(7),
        values(6), million(5), million(5), yesterday(5),
        buying(5), million(4), yesterday(4), volume(4), points(3),
        points(3), close(3), measures(3), buying(2), measure(2),
        yesterday(2), weeks(1), volume(0), yesterday(0)
```

In the case of *crush*, the correction suggestion is *brush* rather than the original word, *crash*. Therefore, this is an example of real alarm with an improper correction suggestion. The chain where *brush* was inserted is the same one as for *tenter* (chain 003). An example of false-alarm is given by *treasure* which has *treasury* as correction suggestion. The chain where *treasury* was inserted is the following:

```
[017]   concern(25), concern(25), concerns(23), concern(22), concerns(22),
        firm(20), concern(18), companies(17), concern(12), treasury(11),
        companies(11), service(9), company(8), institutions(7), hands(5)
```

Another example of false-alarm is given by *basic* which has three correction suggestions: *basis*, *basin*, and *basil*.

Table 3.4 displays the results of the experiment. Since malapropisms are inserted into each input text at about every 200 words, some articles did not have any malapropisms due to their small size. However, whenever a text is too small to provide enough context, the algorithm used to identify lexical chains is not valid. Therefore, for this experiment, a text was considered as being too small if it did not enable the insertion of at least one malapropism. Eighteen such articles were found and removed from every relation. The 482 articles retained for the experiment included a total of 322,645 words, of which 33.9% were inserted into the chain stack. 1,409 malapropisms were generated and 31.4% of them were placed in atomic chains by the chainer, resulting in 442 correct guesses. 7.01% of

Table 3.1: Sample from GENERATION

file_name	*sentence _number*	*original_word*	*malapropism*
art.358	9	today	toady
art.358	18	crash	crush
art.358	29	tender	tenter

Table 3.2: Sample from POTENTIAL_MALAP

file_name	*sentence _number*	*guess*
art.358	9	toady
art.358	9	basic
art.358	12	treasure
art.358	18	crush

non-malapropisms were also inserted in atomic chains, resulting in 7,572 incorrect guesses. Therefore, actual malapropisms are 4.47 times more likely to be inserted in an atomic chain.

89.8% of the correct guesses resulted in alarms (*i.e.*, detected malapropisms) while 36.6% of the incorrect guesses resulted in alarms (*i.e.*, false-alarms). Therefore, correct guesses are 2.46 times more likely to result in alarms than incorrect guesses. The proportion of alarms that are false is 87.5%. The average number of replacement suggestions per alarm was 2.66.

To summarize my results, an alarm was generated for 28.2% of the malapropisms. Furthermore, an alarm with the original word (the word for which a malapropism was substituted) as one of the replacement suggestions was generated for 24.8% of the malapropisms. Malapropisms are 11 times more likely to result in an alarm than other words. However, this is at the cost of 25.3 false-alarms per 1000 words eligible for chaining, or 8.59 false-alarms per any 1000 words of the input text.

Table 3.3: Sample from ALARM

file_name	*sentence _number*	*guess*	*correction*
art.358	9	toady	today
art.358	9	basic	basis
art.358	9	basic	basin
art.358	9	basic	basil
art.358	12	treasure	treasury
art.358	18	crush	brush

Table 3.4: Results

number of words	322,645
number of words in chain stack	109,407
number of non-malapropisms	107,998
number of malapropisms	1,409
number of atomic chains	8,014
number of correct guesses	442
number of incorrect guesses	7,572
performance 1	4.47
number of alarms	3,167
number of detected malapropisms	397
number of false-alarms	2,770
performance 2	2.46
overall performance	11.0
number of perfectly detected malapropisms	349

Chapter 4

Conclusion

4.1 Automatic Identification of Lexical Chains

An appropriate knowledge base and an algorithm making an optimal use of that knowledge base were required to accurately evaluate semantic relations between words in order to identify lexical chains.

Since my aim was to develop an entirely automatic tool for identifying lexical chains, I had to rely on an on-line knowledge-base. WordNet appeared to be the best compromise. It does not express as many relations as Roget's 1911 Thesaurus (which includes, for instance, situational relations) and has few connections between the different parts of speech, but has the benefits from a modern vocabulary (*e.g.*, *psycholinguistics*, *supernova*, *laser printer*), many phrases and expressions (*e.g.*, *capital of Canada*, *health maintenance organization*, *black-and-white*), many proper nouns (*e.g.*, *Isaac Newton*, *Victor Hugo*, *British Columbia*), a comprehensive index, and high reliability.

The lexical chaining algorithm proposed here borrowed ideas from other lexical chain-related work. As for Morris and Hirst (1991), word duplication has priority over any other kind of relation and has an unlimited search scope. Other relations have a limited search scope. As in Okumura and Honda's work (1994), chains are stored and managed in a stack. Finally, as for Hirst's Polaroid Words (1988), a progressive word-sense disambiguation is performed as words are read. WordNet's attribution of many senses to each word appeared to be appropriate for word-sense disambiguation. Problems to select the appropriate word-sense were caused by the difficulty to accurately evaluate semantic relations between words.

A mechanism that quantitatively evaluates semantic relations was required. Word duplication and strong relations such as synonymy or antonymy were given first and second priority respectively. Other relations were evaluated according to the length and the pattern of the path between synset pairs (one synset from each word involved in the connection).

To avoid tagging errors and to ensure an optimal execution speed, no grammatical parser was used. The algorithm considers only nouns, and defines as a noun, every word that can be found in WordNet's noun base either as is or after a morphological transformation as a noun.

The actual implementation of a lexical chain is a complex object that contains words in their reverse order of insertion, keeps track of which word is connected to which, and what senses are involved in each connection. Whenever a word cannot be inserted in any chain, it begins a new chain. Chains are kept in a stack by order of recency (*i.e.*, most recent updated first).

Before the insertion phase, words must pass through a selection phase. Closed-class and high-frequency words are filtered out, and whenever possible, words are grouped together to form compound words and expressions listed in WordNet.

The chains identified by this algorithm have two major problems: under- and over-chaining. Under-chaining might be caused by four reasons:

1. An inadequacy of WordNet's set of relations. For instance, *child care* and *school* cannot be related by using WordNet's set of relations.

2. A lack of connections in WordNet. For instance, WordNet does have a proper set of relations to link *beef stew* and *beef* with a single relation (substance meronym/holonym) but no such link exists in WordNet's graph.

3. A lack of consistency in the semantic proximity expressed by WordNet's links. For example, in WordNet's graph, the shortest path between *stew* and *steak* has 6 links while the shortest path between *Australian* and *millionaire* has 4 links.

4. A poor algorithm for chaining.

Over-chaining might be caused whenever two words are very close to each other in WordNet's graph while being distant semantically. This particular case of the lack of consistency in the semantic proximity expressed by WordNet's links often results in the merging of two chains.

It is perfectly expectable that a lexical chainer fails to connect words that are used in a text as figures of speech. To reduce over-chaining and under-chaining of the other words, a more accurate way to compute semantic proximity is required. One possibility would be to combine another knowledge base, like Roget's 1911 Thesaurus, with WordNet in order to get the best out of both knowledge bases. However, such an attempt would require a strategy to match corresponding word senses of each knowledge-base.

4.2 Automatic Detection of Malapropisms

Spelling errors can be classified into two categories: non-word and real-word errors. Contrary to non-word errors, real-word errors are words that are correctly spelled. They are the result of a spelling mistake on a different word than was intended. A malapropism is a semantic real-word error (*i.e.*, a word that does not make sense in the context where it appears).

Some studies made on the frequency of real-word errors showed a significant quantity of semantic errors in different kinds of text. To detect them, the idea invested in this thesis was to use the property of lexical chains of expressing semantic continuity in order to detect words that did not fit in their context. Whenever a spelling replacement for such a word can be inserted in a chain, it is considered likely enough to be the intended word for which the current word has been mistaken.

To verify this hypothesis, two assumptions needed to be verified: first, that malapropisms have higher probability of not fitting in any lexical chain than other words, and second, that malapropisms have higher probability of having a spelling replacement that fits into a chain of the chain stack.

The experiment performed consisted in taking 500 journal articles and substituting one word out of every 200 with a malapropism. Then, the proportion of these malapropisms that was found in atomic chains was computed and an attempt was made to insert spelling replacements of these malapropisms in the chain stack. Whenever an atomic chain was made of a phrase or an expression, it was not considered to be atomic. The proportion of successful insertion of these replacements was also computed. The mechanism for generating spelling replacements was taken from ispell.

Nearly one malapropism out of three was inserted in an atomic chain and about 90% of these atomic chains resulted in an alarm. However, nearly one correct word out of fourteen was also inserted in an atomic chain and about 36% of these atomic chains resulted into an alarm.

The overall performance of the algorithm shows that malapropisms were 11 times more likely to result in an alarm with replacement suggestions than any other words. Consequently, an alarm with the original word as a replacement suggestion was generated for one word out of four, at the cost of almost 9 false-alarms per 1000 words of the input text.

A more strict chaining would result in a worse detection while a less strict chaining would result in more false-alarms. A lexical chainer that quantifies semantic relations more accurately should enable a higher malapropism detection while decreasing the number of false-alarms.

One way to improve malapropism detection would be to combine the algorithm proposed here with a mechanism that performs real semantical analysis, or uses statistics to estimate the probability of two words to be substituted for each other. Also, a weight could be attributed to each chain, indicating whether or not it provides enough context for a given replacement candidate. In this case, when attempting to insert a given replacement candidate, perhaps atomic chains should be considered only if an extra-strong connection can be made.

Full malapropism detection with no false-alarms at all cannot be expected with the approach proposed here. However, I believe that better performances and the integration of this malapropism detection algorithm to a spelling checker could make a commercializable product.

Appendix A

An Example of Lexical Chaining

This section displays an article provided to my lexical chainer, followed by the lexical chains identified. Written by Harvey Enchin, the article is entitled *Education on trial: Conflict over the classroom*. It was published in the December 29th, 1992, issue of *The Globe and Mail*[1].

A.1 Article Analysed

Fed up with having to teach her two daughters what she thought they should have been learning at public school, Judy Sumner pulled them out two years ago and sent them to private school.

The cost means no holiday trips and more stew than steak, but she is satisfied that her children, now in Grades 3 and 4, are being properly taught.

She opposes the curriculum prescribed by the Ontario government that bundles subjects such as history, geography, science and mathematics into an integrated program — a cornucopia of intersubject learning in which a thematic exploration of autumn, for example, might touch on English, French, art, science and geography.

"I'd had enough," she says. "They keep talking about teaching elementary-school children critical-thinking skills, but kids in Grades 4, 5 and 6 can't even read."

Her complaints about her local public school in Middlesex County in Southwestern Ontario are familiar to parents across Canada who are questioning the effectiveness of the

[1]Reprinted with permission from *The Globe and Mail*. Copyright © 1992 *The Globe and Mail*.

state-run education system.

Many are suspicious of current teaching philosophy that has largely replaced drilling, lecturing and rote with co-operative learning activities. And some are uneasy with the "spiral" curriculum, in which concepts are repeated year after year, growing in depth and complexity as students develop skills and knowledge.

These parents are more comfortable with direct instruction that aims at mastery of one skill before moving on to the next, with a greater focus on content than on interrelationships of subjects.

Still others fear that government belt-tightening will lead to the loss of school services provided by speech pathologists, psychologists, social workers and other professionals. Some taxpayers, meanwhile, are beginning to doubt that they are getting much in return for the $4-billion in public funds spent annually on education.

Parents who want their children to learn something at school are often labelled back-to-basics zealots by supporters of teaching techniques such as whole language, which stresses meaning and context over phonetics, and child-centred learning, which puts the focus of instruction on the student rather than the teacher.

But the idea that teachers should teach and students should learn is appealing to those parents who find that their children can neither read nor write near the end of the elementary years.

Some have found support in grassroots organizations of parents, educators, school board trustees and other interested members of the public who are determined to make the educational system more effective, responsive and accountable.

Ms.Sumner, for example, is a vice-president of the Organization for Quality Education, an incorporated 600-member group in Ontario, which declares in its mission statement that it is dedicated to improving outcomes in publicly financed schools. "We believe that the so-called new methods of play learning, whole language and inventive spelling are lowering children's achievement," the year-old organization says.

It seeks public schools that reflect the will of parents, a province-wide sequential curriculum for each subject and grade, objective evaluations in each subject area set against provincial, national and international standards and wide dissemination of information about effective instructional techniques.

According to OQE president Malkin Dare, there are about two dozen like-minded groups in Ontario alone and she keeps in touch with others in Manitoba, British Columbia and Nova Scotia.

“All the parents had tried to make changes in their own school boards and were unsuccessful,” Ms.Dare says. “The structure of the system is such that the incentive, motivation and hierarchy all bring to bear strong pressure for schools to be a certain way.”

The OQE group holds that genuine self-esteem comes as a result of genuine achievement. That message is echoed by the Quality Education Network, an umbrella organization of 6,000 parents, teachers and taxpayers formed a year ago to ensure that reading, writing and arithmetic remain the base of a good education and to press for regular evaluation of student achievement.

Neither group advocates a return to the rigid schooling methods of the past and both say they welcome programs that stimulate creativity, resourcefulness and love of learning. They urge parents to become actively involved in their children’s schooling, they say.

In some cases, parents have successfully fought instructional methodology that had teachers abandon their traditional role and instead act as facilitators as children moved from one so-called learning centre to another. But without a dynamic principal or ever-vigilant parents group, the school would revert, Ms.Dare says.

Frustrated by the short-lived gains its members have won in individual schools, the OQE is pressing for political action. On that front, the group has mailed letters to members of the legislature, making recommendations on improving Ontario’s low standings in math and science by adopting some of the methods used in the Japanese system.

Last March the organization sent a letter to Ontario Education Minister Tony Silipo in response to proposed legislation governing education. There was no reply. A second letter, mailed in October, was also ignored, Ms. Dare says.

Maybelle Durkin, executive director of the Canadian Home and School Parent Teacher Federation, says this is the wrong way to bring about change.

“These people are bypassing the system,” she says. “And that represents a certain factionalization of values within the community.”

She says that parents would be more successful in their social activism working within the system. In Ontario, that means joining the organization for francophone parents, the organization for English Catholic parents or the Ontario Federation of Home and School Associations — all recognized by the provincial government.

Many advances that parents now take for granted came about, Ms.Durkin says, "because there was an organized parent body within the school system working over time." Among these, she includes French immersion, parent-teacher nights, guidance services, special education for children with learning disabilities and in-school libraries.

There is no uniform approach to parents' participation across the country. Since Quebec introduced school committees in 1971 — after a provincial commission of inquiry found that high-quality education depended on a high degree of parental involvement — most provinces have experimented with parent advisory committees or school councils.

In its 1989 Schools Act, British Columbia created parent advisory councils, requiring each school to have one, with the onus on parents to organize themselves. The legislation deems parents equal partners with teachers and administrators and they sit on high-level committees convened by the deputy minister.

In some provinces, alternative schools within the public system enjoy the most parental input. Parents have much more control over philosophy, curriculum and methodology than in a typical school. In midtown Toronto, for example, parents determined the unique character of the Hawthorne Bilingual School, which goes far beyond provincial guidelines for core French, although it is not a French-immersion school. French is introduced in kindergarten with songs, stories and simple instructions. Toward the end of the elementary years, up to half the instruction is in French.

Parents with children at alternative schools are used to fund-raising events that pay for the additional staff, field trips and other extras, but even regular schools are growing more dependent on private financing to narrow the gap between their budgets and what they receive from penurious governments.

"Fund-raising is not encouraged," said Fiona Nelson, long-serving trustee on the Toronto Board of Education. "It's not forbidden, but we don't want enormous disparities among the schools based on the ability of parents to raise big bucks."

While frowning on fund-raising, school boards are beginning to meet resistance from taxpayers over the rising cost of the education bureaucracy.

Trustees in many jurisdictions have been forced to freeze their pay and, in some cases, roll back salary increases in the face of opposition from local ad hoc groups such as STOP (Stop The Over Payment) that have mounted media, letter-writing and telephone campaigns to protest against the escalating cost of public education.

School administrators say these same taxpayers expect the schools to provide child care and school lunches, to integrate immigrants into the community, to offer special classes for adult students, to present sports, music and vocational programs, to instill a set of values, to teach conflict resolution and AIDS prevention — all without sacrificing the 3Rs.

"These are people who see the school as an avenue for social change," Ms.Durkin says. "So they have been demanding more and more from the school and schools haven't been given the funding to do it."

Taxpayers should be made to see that their best interests are tied up in having an educational system that they understand and support, she says. There must be a dialogue between those who see the school as the agent of social change and those who feel the school's job is to equip children to succeed in a world dominated by science and technology.

"The public doesn't see the connection between what goes on in the classroom and the economic well-being of the country," Ms.Durkin says. "Until we make the connection between the classroom and the economic future of Canada, and until the public accepts the idea that the level of social programs we have here cannot continue unless we can compete successfully, then we won't make too much headway."

A.2 Lexical Chains Identified

```
[044]    headway(65)

[003]    future(65), programs(65), idea(65), technology(62),
         system(61), given(60), present(57), values(57),
         programs(57), education(56), writing(56), cases(56),
         education(55), bucks(54), ability(54), want(54),
         education(53), field(52), alternative(52), end(51),
         character(49), example(49), control(48),
         methodology(48), input(47), alternative(47),
         system(47), inquiry(44), high_quality(44),
         education(44), special_education(42), learning(42),
         time(41), school_system(41), activism(38),
         system(38), values(37), system(36), october(33),
```

education(31), education(31), front(30), system(30), methods(30), abandon(26), methodology(26), cases(26), learning(26), urge(25), love(24), creativity(24), past(24), learning(24), programs(24), methods(24), return(24), base(23), arithmetic(23), writing(23), reading(23), evaluation(23), education(23), education(23), quality(23), self-esteem(22), motivation(21), incentive(21), structure(21), system(21), standards(17), subject_area(17), evaluations(17), objective(17), techniques(17), subject(17), methods(16), language(16), whole(16), learning(16), quality(15), education(15), example(15), system(13), end(12), idea(12), meaning(11), language(11), whole(11), techniques(11), basics(11), want(11), focus(11), learning(11), return(10), education(10), speech(9), lead(9), content(8), focus(8), subjects(8), skill(8), knowledge(7), concepts(7), skills(7), learning(6), system(5), education(5), skills(4), thinking(4), art(2), example(2), program(2), history(2), subjects(2), learning(2), learning(0), thought(0)

[028] social(65), public(65), public(63), agent(62), social(62), taxpayers(61), social(58), people(58), adult(57), immigrants(57), classes(57), set(57), community(57), taxpayers(57), administrators(57), public(56), groups(56), bureaucracy(55), taxpayers(55), private(52), dependent(52), extras(52), staff(52), regular(52), core(49), bilingual(49), provincial(49), public(47), deputy(46), administrators(46), partners(46),

```
        equal(46), minister(46), committees(46),
        teachers(46), councils(45), councils(44),
        committees(44), committees(44), commission(44),
        provincial(44), uniform(43), libraries(42),
        teacher(42), body(41), associations(39),
        catholic(39), provincial(39), home(39),
        federation(39), social(38), community(37),
        people(36), federation(35), home(35),
        executive_director(35), teacher(35), minister(31),
        japanese(30), legislature(30), group(30),
        individual(29), principal(27), group(27),
        centre(26), facilitators(26), teachers(26),
        group(24), regular(23), good(23), network(23),
        taxpayers(23), teachers(23), group(22), bear(21),
        hierarchy(21), groups(18), national(17),
        provincial(17), set(17), mission(15), group(15),
        educators(13), public(13), teachers(12),
        teacher(11), public(10), taxpayers(10),
        professionals(9), pathologists(9)

[011]   level(65), events(52), governments(52), level(46),
        disabilities(42), government(39), grade(17),
        context(11), doubt(10), government(9), grades(4),
        government(2), grades(1)

[013]   canada(65), country(63), jurisdictions(56),
        toronto(53), toronto(49), provinces(47),
        british_columbia(45), quebec(44), provinces(44),
        country(43), ontario(39), ontario(39),
        canadian(35), ontario(31), ontario(30),
        nova_scotia(18), british_columbia(18),
        manitoba(18), ontario(18), province(17),
```

ontario(15), state(5), canada(5), county(5),
ontario(5), ontario(2) classroom(65),
classroom(63), school(62), school(62),
schools(60), school(60), school(58), school(57),
schools(57), school(57), school(55), schools(54),
schools(52), schools(52), kindergarten(50),
school(49), school(49), school(48), schools(47),
high(46), school(45), schools(45), high(44),
school(44), school(44), school(42), school(39),
school(35), schools(29), school(27), schools(21),
public_schools(17), schools(15), school(11),
school(9), public_school(5), elementary_school(4),
private_school(0), public_school(0)

[034] connection(65), connection(63), change(62),
change(58), increases(56), serving(53),
immersion(49), approach(43), immersion(42),
advances(40), joining(39), change(35),
changes(19), lowering(16)

[012] job(62), feel(62), dialogue(62), interests(61),
support(61), funding(60), prevention(57),
resolution(57), conflict(57), music(57),
sports(57), offer(57), protest(56),
campaigns(56), telephone(56), media(56), stop(56),
stop(56), ad(56), face(56), roll(56), freeze(56),
payment(56), opposition(56), letter(56), pay(56),
resistance(55), meet(55), raising(55), raise(54),
disparities(54), raising(53), gap(52),
financing(52), raising(52), pay(52), half(51),
instruction(51), instructions(50), stories(50),
songs(50), guidelines(49), curriculum(48),

legislation(46), act(45), degree(44), involvement(44), participation(43), guidance(42), services(42), organization(39), organization(39), wrong(35), dare(34), second(33), letter(33), reply(32), governing(31), legislation(31), response(31), march(31), letter(31), organization(31), standings(30), recommendations(30), making(30), letters(30), action(29), short(29), pressing(29), dare(28), act(26), role(26), schooling(25), welcome(24), schooling(24), message(23), press(23), achievement(23), organization(23), holds(22), achievement(22), pressure(21), dare(20), dare(18), information(17), dissemination(17), curriculum(17), achievement(16), spelling(16), play(16), organization(16), statement(15), organization(15), organizations(13), support(13), instruction(11), teaching(11), services(9), loss(9), moving(8), mastery(8), aims(8), instruction(8), curriculum(7), activities(6), lecturing(6), teaching(6), teaching(4), curriculum(2)

[015] science(62), philosophy(48), english(39), math(30), science(30), philosophy(6), geography(2), science(2), english(2), mathematics(2), science(2), geography(2)

[002] children(62), children(52), children(42), children(26), children(25), children(16), children(12), child(11), children(11), children(4), children(1), daughters(0)

[043]	avenue(58)
[042]	aids(57)
[041]	special(57), lunches(57)
[040]	child_care(57)
[001]	students(57), parents(54), parents(52), parents(49), parents(48), parents(46), parents(45), parent(45), parent(44), parents(43), parent(42), parent(41), parents(40), parents(39), parents(39), parents(38), parent(35), parents(27), parents(26), parents(25), advocates(24), student(23), parents(23), parents(19), parents(17), parents(13), parents(12), students(12), supporters(11), zealots(11), student(11), parents(11), social_workers(9), psychologists(9), parents(8), students(7), parents(5), kids(4), fed(0)
[006]	salary(56), cost(56), cost(55), cost(1)
[020]	local(56), local(5)
[031]	trustees(56), boards(55), board(53), trustee(53), members(30), members(29), school_boards(19), member(15), members(13), trustees(13), school_board(13)

[039]	fund(55), fund(53), budgets(52), fund(52)
[027]	beginning(55), beginning(10), belt(9)
[008]	trips(52), run(5), exploration(2), trips(1)
[014]	french(51), french(50), french(49), french(49), french(42), french(2), bundles(2)
[005]	years(51), years(12), holiday(1), years(0)
[038]	nights(42)
[007]	means(39), touch(18), touch(2), means(1)
[037]	gains(29)
[036]	resourcefulness(24)
[035]	umbrella(23)
[017]	year(23), year(16), centred(11), billion(10), year(7), year(7), enough(3), autumn(2)
[033]	result(22), outcomes(15)
[032]	president(18), vice_president(15)
[018]	keeps(18), spiral(7), keep(4)
[030]	phonetics(11)

[026]	stresses(11), fear(9)
[029]	funds(10)
[025]	interrelationships(8)
[022]	complexity(7), depth(7), effectiveness(5)
[024]	operative(6)
[023]	rote(6)
[019]	drilling(6), questioning(5), complaints(5)
[021]	familiar(5)
[016]	cornucopia(2)
[010]	steak(1)
[009]	stew(1)

References

Atwell, E. and Elliott, S. (1987). Dealing with ill-formed English text. In *The Computational Analysis of English: A Corpus-Based Approach*, chapter 10. Longman Inc., New York.

Einstein, A. (1939). *Relativity, The Special and the General Theory.* Methuen, London, 12th edition.

Garside, R., Leech, G., and Sampson, G. (1987). *The Computational Analysis of English: A Corpus-Based Approach.* Longman Inc., New York.

Hirst, G. (1988). Semantic interpretation and ambiguity. *Artificial Intelligence*, 34(2):131–177.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Five papers on WordNet. CSL Report 43, Princeton University.

Mitton, R. (1987). Spelling checkers, spelling correctors, and the misspelling of poor spellers. *Information Processing and Management*, 23(5):495–505.

Morris, J. and Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Okumura, M. and Honda, T. (1994). Word sense disambiguation and text segmentation based on lexical cohesion. In *Proceedings of the Fifteen Conference on Computational Linguistics (COLING-94)*, volume 2, pages 755–761.

Roget, P. M. (1992). *Roget's International Thesaurus.* Harper and Row Publishers Inc., fifth edition.

Stairmand, M. (1994). Lexical chains, WordNet and information retrieval. Master's thesis, Centre for Computational Linguistics, UMIST, Manchester.

Wagner, R. A. (1974). Order-n correction for regular languages. *Communications of the ACM*, 17(5):265–268.