An Investigation of Word Sense Disambiguation for Improving Lexical Chaining

by

Matthew John Reinhard Enss

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis investigates how word sense disambiguation affects lexical chains, as well as proposing an improved model for lexical chaining in which word sense disambiguation is performed prior to lexical chaining. A lexical chain is a set of words from a document that are related in meaning. Lexical chains can be used to identify the dominant topics in a document, as well as where changes in topic occur. This makes them useful for applications such as topic segmentation and document summarization.

However, polysemous words are an inherent problem for algorithms that find lexical chains as the intended meaning of a polysemous word must be determined before its semantic relations to other words can be determined. For example, the word *bank* should only be placed in a chain with *money* if in the context of the document *bank* refers to a place that deals with money, rather than a river bank. The process by which the intended senses of polysemous words are determined is word sense disambiguation. To date, lexical chaining algorithms have performed word sense disambiguation as part of the overall process of building lexical chains. Because the intended senses of polysemous words must be determined before words can be properly chained, we propose that word sense disambiguation should be performed before lexical chaining occurs. Furthermore, if word sense disambiguation is performed prior to lexical chaining, then it can be done with any available disambiguation method, without regard to how lexical chains will be built afterwards. Therefore, the most accurate available method for word sense disambiguation should be applied prior to the creation of lexical chains.

We perform an experiment to demonstrate the validity of the proposed model. We compare the lexical chains produced in two cases:

1. Lexical chaining is performed as normal on a corpus of documents that has not been disambiguated.

2. Lexical chaining is performed on the same corpus, but all the words have been correctly disambiguated beforehand.

We show that the lexical chains created in the second case are more correct than the chains created in the first. This result demonstrates that accurate word sense disambiguation performed prior to the creation of lexical chains does lead to better lexical chains being produced, confirming that our model for lexical chaining is an improvement upon previous approaches.

Acknowledgments

First and foremost I would like to thank my supervisor, Chrysanne DiMarco, for all her support. It was Chrysanne who led me to the field of computational linguistics, and supported my interest in classical linguistic approaches despite the prevailing obsession with statistical methods. Without her help this thesis would not have been possible.

I would also like to thank my readers, Robin Cohen and Randy Harris, for their suggestions and corrections, as well as for actually reading my thesis. Randy was also the person who initiated my interest in the field of linguistics when he taught my first course on the subject during my undergrad.

Brenda McBay, Jessica Miranda, and Margaret Towell deserve much thanks for having shepherded me through the administrative trials and tribulations that come with grad school.

I thank my parents Karen and Bo, and my siblings Daniel, Raina, Rhiannon, and Gabriel, for their love and support throughout all of my university studies.

Last, but certainly not least, I would like to thank my friends in the Artificial Intelligence lab, including Laurent Charlin, Greg Hines, Reid Kerr, Fred Kroon, Rado Radoulov, Tyrel Russell, Martin Talbot, and John Whissell. Not only did they lend me their knowledge, advice, and support, they also ensured that grad school was about more than just courses and research.

Dedication

This is dedicated to my parents, Karen and Bodo Enss, for whom I am eternally grateful.

Contents

List of Figures

Chapter 1

Introduction

Coherent text generally has the property of *cohesion*: the elements in the text "stick together" to form a meaningful whole. The cohesion in a text is a product of the syntactic and semantic relations between elements in the text. Among the relations that create cohesion are semantic relations which exist between words related in meaning. These semantic relations create a unity in meaning for the text, such that the text is about the concepts the related words have in common. For example, a passage that is about tools might contain the semantically related words *screwdriver*, *hammer*, and *drill*. These words are semantically related because they are all kinds of tools and are used for assembling and disassembling.

Halliday and Hasan [9] describe in detail how lexical cohesion is achieved in a text. Morris [20] formalized their ideas about cohesion and semantic relations into a concept she called *lexical chains*. Lexical chains are chains of semantically related words in a text. Morris showed that lexical chains could be used to determine which subjects were being discussed where in a text, in effect mapping out the topic structure of the text. She also presented an algorithm for discovering lexical chains in a document. However, this algorithm could not be implemented on a computer because it used Roget's Thesaurus [26], which was not then machine-readable, to determine how words were related in meaning. For example, *hammer* and *drill* are both listed in the tool category in Roget's thesaurus, so Morris's algorithm would place them together in a lexical chain.

Since Morris's work there have been multiple computer-based lexical chaining algorithms developed. Almost all these lexical chaining algorithms use WordNet [1] to determine how words are semantically related. However, WordNet-based lexical chaining algorithms have difficulty with polysemous words, as it is necessary to first determine their intended meaning in the document to properly chain them. For example, if *orange* occurs in a document, it may refer to the colour orange or to the fruit of the same name. It is only related semantically to words such as *apple* and *kiwi* if it is being used as a fruit, and therefore its membership in a chain containing *apple* and *kiwi* depends on its intended meaning in the surrounding context.

Determining the intended meaning of a polysemous word in a given context is known as *word sense disambiguation*, and is a classic unsolved problem in Computational Linguistics. All the lexical chaining algorithms we present perform word sense disambiguation in the process of building lexical chains. In this thesis we argue that word sense disambiguation is a vital part of building lexical chains, that correctness in word sense disambiguation performance is directly related to the correctness of the lexical chains produced, and that for the best lexical chaining performance, word sense disambiguation should be performed prior to chain creation. The centrepiece of our thesis is an experiment in which a lexical chaining algorithm is first run on a document corpus for which the words have already been disambiguated, and subsequently run without the disambiguation information. This experiment will allow us to determine what effect, if any, accurately performing word sense disambiguation before building lexical chains has on the correctness of the lexical chains built. Our results show that prior accurate word sense disambiguation significantly improves the correctness of the lexical chains produced, regardless of how the word sense disambiguation is performed.

The layout of this thesis is as follows:

Chapter 2 provides the background material necessary for understanding computational approaches to lexical chaining. First we present the work of Halliday and Hasan [9] on lexical cohesion and explain how it provides the linguistic basis for lexical chains. We then describe Morris's [20] original lexical chaining algorithm in detail. Together these two works provide the theoretical foundation for all subsequent lexical chaining research. We follow this background material with detailed descriptions of WordNet (a form of machine-readable thesaurus), word sense disambiguation, and semantic relatedness. Knowledge of all three of these subjects is necessary for understanding the lexical chaining algorithms we present in Chapter 3.

Chapter 3 is our literature review, where we survey the main lexical chaining algorithms. The algorithms we present are from Hirst and St. Onge [11], Barzilay and Elhadad [3], Silber and McCoy [27] [28] [29], Galley and McKeown [7], and Stokes [31]. We pay particular attention to how the algorithms perform word sense disambiguation.

Chapter 4 presents the theoretical basis for our claim that word sense disambiguation should be performed prior to lexical chaining. We describe how the approaches used by lexical chaining algorithms to perform word sense disambiguation are constrained. We argue that with an ideal semantic relatedness measure, improved word sense disambiguation will always lead to improved lexical chaining performance. We then outline our experiment for testing the effect of perfect word sense disambiguation on lexical chaining.

Chapter 5 contains the complete description of our experiment and results, including a detailed specification of the lexical chaining algorithm we used. Our lexical chaining algorithm is a slight but significant modification of Silber and McCoy's lexical chaining algorithm [27] [28] [29]. Our results show that performing perfect word sense disambiguation before lexical chaining significantly reduces the number of incorrect chains created. However, our results also indicate that in some cases errors in word sense disambiguation allow for correct lexical chains that are not possible with perfect word sense disambiguation.

Chapter 6 contains our future work and our conclusions. We describe two areas for future work that stem from this thesis. The first is changing existing lexical chaining algorithms to perform word sense disambiguation prior to chain creation. We also briefly cover how other applications that use lexical chains can be used to further evaluate our results. The second direction is improving WordNet to incorporate semantic relations that we identify in our experiment. We conclude with a summarization of our results and contributions, the concluding principles being that accuracy in word sense disambiguation is necessary for good lexical chaining performance and word sense disambiguation should be performed separately from lexical chain creation.

Chapter 2

Background

In this chapter we start by presenting Halliday and Hasan's work on lexical cohesion [9]. It provides the linguistic basis for Morris [20] and Morris and Hirst [21], which we describe next. It was Morris who first proposed and defined "lexical chains" [20]. After describing Morris's work, we present the background material necessary for understanding how computer-based algorithms for lexical chaining work.

2.1 Lexical cohesion

For a text to make sense, it must be more than just grammatically correct. Meaningful text is *about* something, a topic, and as such the words in the text will be related to that topic in meaning. For example, this paragraph and the following one are both about cohesion in text, hence the repeated use of the words *cohesion* and *text*, as well as words related to them such as *topic*, *words*, *meaning*, etc.

Halliday and Hasan [9] give a comprehensive description of how cohesion in text is created in their seminal book on the subject, *Cohesion in English*. As they describe it, cohesive text "hangs together as a whole." Cohesion exists in part as relations between semantically related parts of the text. These relations include lexical cohesion relations, or relations between pairs of words. Lexical cohesion relations are very useful for computational methods because they can be defined formally.

Morris and Hirst [21] describe Halliday and Hasan's relations as being divided into five classes (definitions and examples produced verbatim):

1. Reiteration with identity of reference:

- Mary bit into a *peach*.
- Unfortunately the *peach* wasn't ripe.

2. Reiteration without identity of reference:
   - Mary ate some *peaches*.
   - She likes *peaches* very much.
3. Reiteration by means of a superordinate:
   - Mary ate a *peach*.
   - She likes *fruit*.
4. Systematic semantic relation (systematically classifiable):
   - Mary likes *green* apples.
   - She does not like *red* ones.
5. Nonsystematic semantic relation (not systematically classifiable):
   - Mary spent three hours in the *garden* yesterday.
   - She was *digging* potatoes.

Morris and Hirst make a clear distinction between the first three classes and the last two. As they point out, the first three classes all involve reiteration, what Halliday and Hasan call the "repetition of a lexical item." In the first two cases the repetition is obvious as the same word, *peaches*, has been used twice. Superordinate terms provide repetition because the superordinate includes the subordinate in the entity to which it refers, e.g., *fruit* refers to *peaches* as well as to all other kinds of fruit.

Morris and Hirst describe the fourth class as systemically classifiable because the words are related by a specific semantic relation that is easy to identify. Examples of these easy-to-identify relations that they give include: set membership (e.g., {*red*, *green*, *blue*, ...}), antonymy, and part-to-whole relationships (e.g., *wing-airplane*). In fact, each of the first four classes of relations can be determined systemically. Identifying the repetition of a word (classes one and two) is just text-matching. Identifying superordinate terms is done by comparing one word to the parents of another in a hierarchical categorization of all the words in the language. Determining the hierarchical categorization is not easy, but can be done (see WordNet [1], which we discuss later). The systemic semantic relation can be determined in largely the same way as superordinate terms are recognized, as WordNet includes both part-of and antonym relations. Determining whether two words belong to a semantically related set such as {*red*, *green*, *blue*} just requires an enumeration of all the allowable sets that can be searched.

It is the nonsystematic semantic relation that is troublesome for any automated processing of cohesion, precisely because there is no system for determining it. In the example given, *garden* and *digging* are related because digging is an action that people perform in a garden. Other kinds of relations that fall into this category include entities and actions the they perform (*wheel*, *roll*), entities and the adjectives often applied to them (*car*, *fast*), and objects that are used as part of the same activity (*spoon*, *pot*). However, the list of possible relations covered by this class is so expansive that it is extremely difficult identify all of them with an algorithm. Even listing all the possible relationships that fall into this category may not be possible. Morris and Hirst do not define the relations belonging to this class, saying only that "the exact relationship between these words can be hard to classify, but there does exist a recognizable relationship."

While the above five classes of relations are often used to define semantic relationships in lexical chaining algorithms, there are three different classes of semantic relation that are also often used: *semantic similarity*, *semantic relatedness*, and *semantic distance*.

Semantic similarity is a measurement of the degree of similarity between two concepts. For instance, cars and trucks are similar in that both perform the same function, are very similar in construction (both have wheels, engine, seats, doors, etc.), and are used in generally the same way (driven by people on roads).

Semantic relatedness is much more broad, encompassing any possible relation between two items. As such, it is equivalent to the union of Morris and Hirst's five classes of relations. In this way, semantic relatedness includes semantic similarity; two items can be semantically related by being semantically similar. Beyond similarity, semantic relatedness includes any semantic relation between two items, even if the kind of relation cannot be defined.

Semantic distance is the inverse of semantic relatedness. It measures the degree to which two concepts are unrelated. If two terms are strongly semantically related then the semantic distance between them is small. Conversely, if two items are dissimilar then there is great semantic distance between them.

While semantic relatedness encompasses all five of Morris and Hirst's classes of relations, semantic similarity is roughly equivalent to the first three classes and part of the fourth. When a term is repeated (classes one and two) then obviously the two identical terms are similar in meaning—we can say that they are maximally semantically similar. Superordinate terms have features that are shared by their subordinates, and hence they are similar in their features. For example, robins are semantically similar to birds because robins have wings, feathers, a beak, etc., just like all birds. With regard to the fourth class, systemic semantic relations, there is similarity between two members of the same set (such as the *red*, *green* example). But other systemic semantic relations, such as meronymy, do not imply similarity: a wheel is not very similar to a car, even though a car has wheels; they are very different in their function and appearance.

The above five classes are presented here because they provide the basis for Morris and Hirst's original description of how lexical chains should be created. Furthermore, these classes have been used as the basis for relations in most of the lexical chaining algorithms that have been written since (in particular Hirst and St. Onge [11], Barzilay and Elhadad [3], Silber and McCoy [29], and Stokes [31]). In the section on semantic measurements we explore in detail how semantic relations can be determined computationally.

2.2 Lexical chains in Morris and Morris & Hirst

Morris [20] (and later in Morris and Hirst [21]), first proposed the idea of lexical chains, which they described as "sequences of related words". These lexical chains would indicate the cohesive structure of a text, as defined by Halliday and Hasan [9]. The relations between the words in the chains are those we listed above, adapted from Halliday and Hasan.

Here is a sample lexical chain in a passage of text from Morris and Hirst [21]:

> In front of me lay a *virgin* crescent cut out of *pine bush*. A dozen houses were going up, in various stages of construction, surrounded by hummocks of dry earth and stands of precariously tall *trees* nude halfway up their *trunks*. They were the kind of *trees* you might see in the mountains.

Morris and Hirst investigated the use of using lexical chains to determine the discourse structure of a text as defined by Grosz and Sidner [8]. Morris and Hirst's goal was to automatically discover discourse segments, which are segments of a text that are about a single topic or subject. Automatic discovery of discourse segments is an important step in determining discourse structure, and Morris and Hirst were able to show that the boundaries of the lexical chains (i.e., where the chains start and end in the text) formed by their algorithm corresponded strongly with the divisions between discourse segments.

2.2.1 An algorithm for building lexical chains

Before lexical chaining can be performed on a document, the words to be considered for chaining must be selected. These words are referred to as the *candidate words* for the document. Morris and Hirst did not specify how to choose candidate words, but did rule out pronouns, prepositions, and verbal auxiliaries. They also excluded high-frequency words, listing *good*, *do*, and *taking* as examples.

According to Morris and Hirst's description of a general lexical chaining algorithm, a word is placed into a chain by virtue of the scored relations between it and the other words in that chain. There are two factors used to determine the relations scores between two words: the *semantic relatedness* between the words and the physical *distance* between the words in the document.

The distance between words is important for two reasons. First, if two words are related in meaning but are far apart in the text, then they may not actually refer to each other. For example, if *grow* and *fruit* occur in the same sentence, it is much more likely that *grow* actually refers to the fruit growing or being grown than if *grow* appears in a separate sentence or paragraph. Secondly, even if a word is repeated, if the second occurrence happens much later in the text then the text between the words may not have anything to do with the words. In this case, the subject of discussion to which both words relate ended, the subject changed, but then the discussion returned to the original subject. Morris and Hirst term these occurrences as *chain returns*. Properly identifying chain returns was a very important goal for Morris and Hirst, as these correspond to breaks in the discourse segments. In examining a small set of documents, Morris and Hirst found that two semantically related words could be at most three sentences apart and still be linked together in a chain. Beyond three sentences, a semantically related word can only indicate a chain return.

Some of the lexical chaining algorithms that we present later do not identify breaks in chains or chain returns. As a result, the lexical chains they produce are not suitable for discourse analysis, but are still useful for other applications we mention later.

Morris and Hirst used Roget's Thesaurus [26] to identify most semantic relations between words. The thesaurus groups words into categories, where a category is a list of words that are all related in meaning. The thesaurus does not specify the nature of the relation. Morris and Hirst also used links between the categories and the manner in which the thesaurus groups the categories to determine relations. We do not go into details of the thesaurus's organization or Morris and Hirst's methods for determining relations as neither is relevant to later lexical chaining algorithms, which generally use WordNet [1] to determine semantic relations.

Morris and Hirst did not implement their lexical chaining algorithm on a computer because Roget's thesaurus was not available to them in a machine-readable form. However, that is not the only problem that they would have to overcome to automate their algorithm, as Morris and Hirst were also determining the meaning of polysemous words by hand. If a word is polysemous, then it appears in multiple categories in Roget's Thesaurus. When polysemous words were encountered by their algorithm, Morris and Hirst used their own judgement to decide which category in the thesaurus applied to the particular instance of the polysemous word. For a lexical chaining algorithm to be completely automated, it must determine the intended meaning of any polysemous words itself. This thesis focuses on the problems inherent in handling polysemous words, and the effect these problems have on the lexical chains produced.

2.3 Word sense disambiguation

We now turn our attention to the problems inherent in modeling the meanings of words, and the relations between those meanings, in computational Natural Language systems.

Many words in the English language have multiple meanings or senses. For example, *bat* can refer to both a baseball bat and a flying mammal, among other meanings. However, when a word is used in a document usually only one of its possible meanings is intended, and it is the context surrounding each word's use that makes the intended meaning unambiguous. In the phrase "bats often sleep in caves", *bats* refers to flying mammals and not baseball bats, as flying mammals sleep and baseball bats do not. In this context, *bats* should not be put in a lexical chain with baseball-related terms such as *glove*, *home run*, and *outfield*. Likewise, the baseball meaning of *bat* should not be chained with words such as *wings* and *echolocation*.

Because Morris and Hirst's lexical chaining algorithm was not automated, they manually *disambiguated* (i.e., determined the intended meaning) of every candidate word themselves. All lexical chaining algorithms proposed since have disambiguated the candidate words automatically in the course of building the lexical chains. Automatic word disambiguation requires a machine-readable lexicon that lists all the possible senses for all the possible candidate words. A single sense is then selected for each word before it is placed in a chain. Morris and Hirst used Roget's Thesaurus [26] as their lexicon. The thesaurus contains an index of all its words, and under each word all of its different meanings are listed. Each different meaning is described with a single word or short phrase (Figure 2.1 lists the different noun senses of *bat* in Roget's Thesaurus) and an index to the category for that meaning. The category entry is a list of words related to the noun sense from the index.

- blind 441.4
- hit 283.4
- mammal 414.58; 415.8
- plaything 878.16
- revel 878.6
- spree 996.5

Figure 2.1: Noun senses of *bat* from Roget's Thesaurus [26]

Morris and Hirst did not attempt to implement their lexical chaining algorithm because no machine-readable version of Roget's thesaurus was available to them. Since then, the 1911 version of Roget's thesaurus has been made available online, but Hirst and St. Onge [11] claim it is unusable for lexical chaining for two reasons: it lacks an index of the words and its information about the English language is outdated. Instead the vast majority of lexical chaining algorithms developed since Morris and Hirst have used the electronic lexical database WordNet [1] as their lexicon of word senses.

2.4 WordNet

WordNet [1] is a linguistic ontology of English words. By design, it is easily accessed in computational form, making it an ideal resource for Natural Language applications, especially lexical chaining algorithms. WordNet is based on psycholinguistic principles, so it is possible to search for words conceptually, rather than just lexi-

Synset	Index	Gloss
{bat, chiropteran}	2055000	"nocturnal mouselike mammal with forelimbs modified to form membranous wings and anatomical adaptations for echolocation by which they navigate"

Figure 2.2: WordNet sense entry for the flying mammal meaning of *bat* [1]

cally as in a dictionary. In this way, WordNet is very similar to a thesaurus such as Roget's [5] [1].

Figure 2.2 shows a sample sense entry from WordNet. Each WordNet sense has three attributes defined as follows:

- *Synset*: Synset is short for "synonym set". A synset is a set of all the words that are synonyms for the sense. In the sample sense, *bat* and *chiropteran* are synonyms for "flying mammal", where *chiropteran* is the scientific term for this order of mammals.
- *Index*: The index of a sense is an integer that is a unique identifier for the sense.
- *Gloss*: The gloss is a short description of the sense. The purpose of the gloss is to make the meaning of the sense clear to the average English reader. It is often similar to a definition but not necessarily so. Some glosses contain examples of one of the synonyms being used.

WordNet contains only nouns, verbs, adjectives, and adverbs. Commonly used lexical chaining algorithms to date have used only nouns as candidate words, so we look only at the noun portion of WordNet.

WordNet contains a number of formal semantic relations between senses. Among nouns, the most prominent relations are *hypernymy* and *hyponymy*. Sense a is a hypernym of sense b if a entirely encompasses b. For example, mammal is a hypernym of elephant because all elephants are mammals. The hyponym relations goes in the other direction: if a is a hypernym of b, then b is a hyponym of a. This relationship is also referred to as the *is-a* or *kind-of* relationship, as we can say that an elephant

[1] All the information presented here pertains to version 2.0 of WordNet, though most of it also applies to previous versions as well. Throughout this thesis, any reference to WordNet refers to version 2.0.

is-a mammal or that an elephant is a kind-of mammal. We also refer to the hyponyms of a sense as its *children* and a sense's hypernyms as its *parents*. As well, two senses are *siblings* if they have a common parent. We recursively define the *descendants* of a sense as its children, and all children of its descendants. Likewise, a sense's *ancestors* are its parents, and all parents of its ancestors. For example, mammal is an ancestor of all senses that refer to specific kinds of mammals, and all the specific kinds of mammals are descendants of mammal. The hypernymy and hyponymy relations among the nouns form a *hierarchy* of nouns. This hierarchy is a directed acyclic graph, and our use of parents, children, siblings, ancestors, and descendants is consistent with their use with regards to directed acyclic graphs [5] [18].

Most noun senses in WordNet have only a single parent, so large portions of the noun hierarchy have a tree structure. However, the noun hierarchy does not have a single root sense from which all other senses descend. Instead there are nine noun senses with no parents (listed in Figure 2.3) which are referred to as the *unique beginners*.

Synset	Gloss
{entity}	that which is perceived or known or inferred to have its own physical existence (living or nonliving)
{psychological_feature}	a feature of the mental life of a living organism
{abstraction}	a general concept formed by extracting common features from specific examples
{state}	the way something is with respect to its main attributes; "the current state of knowledge"; "his state of health"; "in a weak financial state"
{event}	something that happens at a given place and time
{act, human_action, human_activity}	something that people do or cause to happen
{group, grouping}	any number of entities (members) considered as a unit
{possession}	anything owned or possessed
{phenomenon}	any state or process known through the senses rather than by intuition or reasoning

Figure 2.3: Unique beginner noun senses in WordNet [1]

In addition to hypernymy and hyponymy, the other semantic relations between nouns in WordNet are *meronymy*, *holonomy*, and *antonymy*. Sense a is a meronym

Synset	Index	Gloss
{bat, chiropteran}	2055000	nocturnal mouselike mammal with forelimbs modified to form membranous wings and anatomical adaptations for echolocation by which they navigate
{bat, at-bat}	434369	(baseball) a turn batting; "he was at bat when it happened"; "he got 4 hits in 4 at-bats"
{squash racket, squash racquet, bat}	4127511	a small racket with a long handle used for playing squash
{cricket bat, bat}	3018500	a bat used in playing cricket
{bat}	2708091	a club used for hitting a ball in various games

Figure 2.4: All noun senses of *bat* in WordNet

of sense b if a is a part of b. For example, chain is a meronym of bicycle because a chain is part of a bicycle. If a is a meronym of b, then b is a holonym of a. Meronymy is often referred to as the *part-of* relation, and holonymy is often referred to as the *has-a* or *contains* relation. Antonymy exists between senses that are the opposite of each other. For example, back and front are antonyms. Meronymy, holonymy, and antonymy are all much less prevalent in WordNet than hypernymy and hyponymy, and many semantic relatedness measurements and lexical chaining algorithms ignore them.

If a word has multiple meanings, then it appears in multiple synsets. The set of possible meanings of a word w in WordNet is the set of all senses such that the synsets of those senses contain w. For example, Figure 2.4 lists all the noun senses of *bat* in WordNet.

Lexical chaining algorithms developed since Morris and Hirst [21] use WordNet to determine the semantic relations between words. When a word w belonging to multiple synsets is encountered, the lexical chaining algorithm disambiguates w by choosing one sense from WordNet for it. The WordNet relations for the chosen sense of w are then used to determine which other words in the text should be chained with w.

2.5 Semantic relatedness

WordNet-based lexical chaining algorithms require a means of determining how strongly pairs of senses are related semantically to each other. The hypernymy and hyponymy relations in the noun hierarchy provide one method for measuring semantic relatedness. By definition, any two senses linked by hypernymy or hyponymy should be semantically related. Intuition suggests that senses connected by a short path in WordNet should also be related, due to semantic relations being transitive to some degree. As well, senses with short paths between them in the hierarchy often have a close common ancestor. Senses with a common ancestor are often similar to one another because they all share the properties of the ancestor. For example, it can be deduced that car and truck are similar because both are children of the WordNet sense {motor_vehicle, automotive_vehicle}: "a self-propelled wheeled vehicle that does not run on rails".

It should be noted that under this reasoning, the length of the shortest path between two senses is inversely related to their semantic relatedness. Therefore the length of the path is actually a measurement of the semantic distance between the two senses. This measurement can be converted into one of semantic relatedness by either inverting it (i.e., $relatedness = \frac{1}{distance}$), or by subtracting the path length from a constant value: $relatedness = C - distance$.

One problem with using hypernym and hyponym paths to determine relatedness is deciding the maximum length of a path to allow. Deeper parts of the noun hierarchy tend to be very detailed, and as a result there can be many intermediate senses on the path between two related senses. For example, Dobermans and Labrador retrievers are semantically related because they are dogs, but the shortest path between them is eight senses long. From Doberman to dog the path is Doberman *is-a* pinscher *is-a* watchdog *is-a* working dog *is-a* dog, and from Labrador retriever to dog the path is Labrador retriever *is-a* retriever *is-a* sporting dog *is-a* dog (the path from Doberman to Labrador retriever is shown in detail in Figure 2.5). Despite Dobermans and Labradors both being dogs, the great number of distinctions between different types of dogs causes the two breeds to be far apart in WordNet.

Conversely, some unrelated senses have fairly short paths between them. These short paths usually pass through the senses close to the top of the noun hierarchy, which tend to be very general or broad in their meaning. Consequently, their descendants are often not very closely related to each other. For example, the path between water and human is only five senses long and goes through the shared ancestor, entity. Figure 2.6 shows this path in detail.

To alleviate both these problems, various methods have been proposed for weighting the links between senses such that the sum of the weighted shortest path between any two senses more accurately reflects the semantic distance between them (Budanitsky and Hirst [4] provide a good overview of such methods).

{*Doberman, Doberman_pinscher*}	medium large breed of dog of German origin with a glossy black and tan coat; used as a watchdog
{*pinscher*}	any of three breeds of dogs whose ears and tail are usually cropped
{*watchdog, guard_dog*}	a dog trained to guard property
{*working_dog*}	any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs
{*dog, domestic_dog, Canis_familiaris*}	a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds; "the dog barked all night"
{*sporting_dog, gun_dog*}	a dog trained to work with sportsmen when they hunt with guns
{*retriever*}	a dog with heavy water-resistant coat that can be trained to retrieve game
{*Labrador_retriever*}	breed originally from Labrador having a short black or golden-brown coat

Figure 2.5: Shortest path between Doberman and Labrador retriever in WordNet [1]

Given that senses deeper in the WordNet noun hierarchy are more closely related to their neighbouring senses than senses located higher up, it makes sense to weight links between senses based on their depth. For example, the animal sense of *dog* has a depth of 11, where as entity, the closest common ancestor of water and human, has a depth of zero as a top-level sense in the noun hierarchy. Jiang and Conrath [12] argue that this effect of depth occurs because as senses become more specific deeper in the hierarchy, the distinctions between their children become more and more fine. As a result, the semantic distance between the children shrinks.

Richardson and Smeaton [25] argue that the density of the hierarchy also affects semantic distance in that the children of a sense with many children will be more closely related than the children of a sense with few children. Each sense can be thought of as occupying a semantic space, and its children divide this space up into subspaces. Therefore if there are many children, they are only accounting for a small part of what their parent sense represents or means, and so will be closer in meaning than if their parent had only a few children.

Budanitsky and Hirst [4] describe how these and other methods can be used

{body_of_water, water}	the part of the earth's surface covered with water (such as a river or lake or ocean); "they invaded our territorial waters"; "they were sitting by the water's edge"
{thing}	a separate and self-contained entity
{entity}	that which is perceived or known or inferred to have its own distinct existence (living or nonliving)
{causal_agent, cause, causal_agency}	any entity that causes events to happen
{person, individual, someone, somebody, mortal, human, soul}	a human being; "there was too much for one person to do"

Figure 2.6: Shortest path between water and human in WordNet [1]

to measure semantic similarity. They also provide a comprehensive evaluation of the various methods. We present the previous two methods to highlight some of the problems inherent in using paths in the WordNet noun hierarchy to determine semantic relatedness. These problems are important as they lead to errors in lexical chaining that we explore later.

Not only are the paths in WordNet not necessarily reflective of the semantic distance between the senses they link, but often there is no path between two related senses, or the path between the related senses is overly long and passes through completely unrelated senses. For example, tennis racket and tennis have no common ancestor or descendant. The shortest path between them is 14 senses long and passes through senses such as weaponry and human activity. The relations in WordNet do not provide a way to determine that racket, ball, and net are all related to the game of tennis. In general, many semantically related concepts are not linked in a way that indicates their relation. This problem arises because the number of possible semantic relations is qute large and are often not easy to quantify. As Morris and Hirst [21] mentioned, some words just "feel" related. These are the relations that they described as nonsystematic semantic relations which are not systematically classifiable. Handling such relations is exceedingly difficult for any electronic thesaurus and has become known as the 'Tennis Problem' [5]. The lack of these paths in WordNet causes the same kind of errors as caused by the overlong paths between related senses we mentioned above.

Chapter 3

Literature review

In our background chapter we discussed Morris's [20] seminal work on lexical chaining. Morris proposed the concept of lexical chains and provided an algorithm for finding lexical chains in a document. Morris did not implement her algorithm because Roget's Thesaurus [26], which was used to determine which words were semantically related, was not available then in machine-readable form. As we pointed out, lack of a machine-readable dictionary is not the only hurdle Morris faced in implementing her algorithm—she also lacked a means for performing automated word sense disambiguation.

In this section we review five key lexical chaining algorithms: Hirst and St. Onge [11], Barzilay and Elhadad [3], Silber and McCoy [27] [28] [29], Galley and McKeown [7], and Stokes [31]. We chose these papers because they contain detailed descriptions of their respective lexical chaining algorithms and because all are well-cited in the field. While there are many more published works about lexical chaining, most of these focus on applications, rather than the details of the algorithms. As we are concerned with word sense disambiguation, and its effects on lexical results, applications of lexical chains are not germane to this thesis. For descriptions of the papers that deal primarily with the applications of lexical chaining, we recommend the second chapter of Stokes's thesis on lexical chaining [31].

3.1 Hirst and St. Onge

Hirst and St. Onge [11] [30] use lexical chains to detect *malapropisms*, or real-world spelling mistakes. These are errors where a person makes a mistake in spelling a word, but the resulting word is still a correct word—e.g., writing *purse* when *nurse* is intended. These errors are difficult for automatic spell checkers to detect because a dictionary lookup would indicate that *purse* is a correct spelling. Hirst and St. Onge reason that words in a document are usually semantically related to each other, and therefore words that are not related to surrounding words around them, i.e., words that are not part of a lexical chain, are possibly incorrect. If there exists another word w that is close in spelling to an unchained word and w would fit into a lexical chain already present in the document, then it is quite possible that w is the word the author intended to write, and the unchained word is a spelling mistake.

Hirst and St. Onge use all words that appear in the noun section of WordNet as candidate words. This means that all the nouns in a document are considered candidates, as well as all words that have a noun form. For instance, *walk* is considered a candidate word even if it is being used as a verb in the document (adjectives and adverbs are treated the same way). These words with noun forms are treated as nouns for the purpose of chaining because even if they are not being used as nouns, usually they are still semantically related to the noun form.

Hirst and St. Onge define three levels of semantic relation with regard to WordNet: *extra-strong*, *strong*, and *medium-strong*.

- *Extra-strong* relations exist only between repetitions of the same word. Hirst and St. Onge require that all instances of the same word in a document have the same sense, so all words that have an extra-strong relationship will have the same meaning. For example, if *bat* appears multiple times in a document, then the lexical chainer will not consider one instance as referring to a baseball bat and another instance as referring to a mammal with wings.

- There are three types of *strong* relations:

    1. Members of the same synset. For example, there is a strong relation between *exam* and *test* because both are synonyms for the sense "a set of questions or exercises evaluating skill or knowledge; *when the test was stolen the professor had to make a new set of questions*" [1].
    2. Senses that are antonyms.
    3. Two senses that are linked in WordNet where one of the synonyms of one sense is a compound word that contains one of the synonyms of the other sense. An example of this is *private_school* and *school*, as the only sense of *private_school* is a child of one of the senses of *school*.

- *Medium-strong* relations exist between senses that have an *allowed* path in WordNet between them. Allowed paths are defined below. All allowed paths are assigned a score, where a higher score indicates that the senses are more closely related.

3.1.1 Allowed paths

Hirst and St. Onge classify noun relations in WordNet in terms of direction. They define hypernym and meronym links in WordNet as being *up* relations and hyponym and holonym relations as being *down* relations. Antonymy is defined as a *horizontal* relation because it indicates neither generalization (up) nor specialization (down).

The only allowed paths without horizontal links consist of just down links, just up links, up links followed by down links, or down links followed by up links. In the first three cases the connected senses have a shared ancestor, and in the last three cases they have a shared descendant.

If an allowed path contains horizontal links, then the path can have at most one change in direction, except for one case we list below. So the horizontal links may all come before a series of down links or after a series of down links, but not before and after. The same is the case for up links. The only case where more than one change in direction is allowed is when a series of up links is followed by a series of horizontal links which are then followed by a series of down links.

All of these allowed paths have a weight defined by the function:

$$weight = C - path\ length - k \times number\ of\ changes\ in\ direction$$

where C and k are constants.

3.1.2 Chain creation

Hirst and St. Onge's lexical chaining algorithm makes a single pass through the document, during which all the words are processed from left to right. For each word w, the existing chains are searched for a word that shares an extra-strong, strong, or medium-strong relation with w. If a relation is found, then w is added to the chain that contains the word with which w is most strongly related. If there is a tie among chains, then the chain that was most recently updated is chosen. If no relation is found for w among the already existing chains, a new chain is created and w is added to it.

3.1.3 Word sense disambiguation

For each word in a chain, Hirst and St. Onge's lexical chaining algorithm stores a set of possible senses. When a chain is first created it contains a single word. No disambiguation has been performed yet, so every sense of that word in WordNet is in its set of possible senses. If a word w is added to an already existing chain, then there exists some word wc already in the chain that has at least one sense related to a sense of w. When w is added, the possible senses for wc and w are reduced to senses that are most strongly related between them. For example, if w has possible senses $\{w_1, w_2, w_3, w_4\}$ and $wc\{wc_1, wc_2\}$, and w is added to wc's chain because w_1 and w_3 are strongly related to wc_2, then w's possible senses are set to $\{w_1, w_3\}$ and wc's are set to $\{wc_2\}$. When future words are processed, only senses w_1, w_3, and wc_2 will be checked for relations with the new word. This constraining of senses disambiguates words as they are added to chains.

If adding w changes the set of possible senses for wc, then those changes are propagated through the chain. For example, consider a chain that contains only the single term *bat*. *Bat* has five noun senses in WordNet, one of which refers to the flying mammal, and all of these will be stored in the chain as possible senses for *bat*. If another instance of *bat* is added to this chain then it will be connected to the first instance via an extra-strong relation, and its set of possible senses will be set to the same five senses as the first instance of *bat*. The word *wing* can be added to this chain because it has a medium-strong relation with *bat*: one sense of *wing* has a *part-of* relation with the mammal sense of *bat*. *Wing* would be connected with the most recent instance of *bat*, and their sets of senses would be constrained to only contain the single sense that shares the medium-strong relation (for *bat* this would be "nocturnal mouselike mammal" and for *wing* it would be "a movable organ for flying"). Then the change in the possible senses for the second instance of *bat* would be propagated to the word in the chain to which it was connected, the first instance of *bat*, and its possible senses would be set to the same single sense as the second *bat*.

Rarely does a word have more than one sense that is related to another word. Therefore whenever the algorithm adds a word to a chain using a relation other than extra-strong, a single sense is usually chosen for both the new word and the word in the chain to which it is connected. As a result, Hirst and St. Onge's algorithm disambiguates words rather quickly, and only looks at previously processed words when deciding which chain a word should be placed in. Other lexical chaining algorithms that we review consider all the possible relations for a word before choosing a sense for it.

Analysis Various evaluations of semantic relatedness measures have found fault with Hirst and St. Onge's path-based measure for medium-strong relations [4] [10] [23]. In particular, it is noted that Hirst and St. Onge's allowed paths can connect many unrelated senses in WordNet, particularly senses located near the top of the noun hierarchy (we describe this problem at the end of chapter 2). As a result, their lexical chaining algorithm often chains words together even though none of those words' senses are semantically related.

As we noted above, Hirst and St. Onge's algorithm processes words in the order they appear in the document, and, when processing a word w, examines previously processed words first for relations with w. Therefore words that appear prior to w are more likely to be used for disambiguating w than words that appear after it. As well, multiple occurrences of words that are related to w are not given more weight than a single related word: if a single word related to a sense w_1 of w occurs before w and five instances of a word related to a sense w_2 of w occur after w, then sense w_1 is used to connect w to the previous related word before words after w are even considered (assuming the strengths of all the relations are the same). Barzilay and Elhadad [3] argue that this approach leads to poor word sense disambiguation performance, and that all related words in the document should be taken into consideration when selecting a sense for a word. Galley and McKeown [7] also argue for considering all related words when disambiguating. Pedersen et al. [23] propose a general framework for performing word sense disambiguation by using various semantic relatedness measures. They find that when disambiguating a word w, accuracy increases as the window of surrounding words considered for relations increases in size, which suggests that localized approaches to disambiguation such as Hirst and St. Onge's should not be used.

3.2 Barzilay and Elhadad

Barzilay and Elhadad [3] use lexical chains for document summarization. They argue that the strongest lexical chains in a document reflect the central or most important topics of the document, where the strength of a lexical chain is defined as its length minus the number of unique words it contains. This equation recognizes that the longer a chain is the more important it is, but also takes into account the importance of word repetition.

Barzilay and Elhadad use only nouns and noun compounds as candidate words, stating that nouns are the most important words for determining the topic of a document. To identify nouns they use a part-of-speech tagger, and to identify compound words they use a shallow parser.

They use a small number of simple semantic relations to build lexical chains: reiteration, synonymy, antonymy, hypernymy/hyponymy, and holonymy/meronymy. Each relation is assigned a weight, where a greater weight indicates a stronger semantic relation: reiteration and synonymy ten, antonymy seven, and four for hypernymy/hyponymy and holonymy/meronymy.

In describing their lexical chaining algorithm, Barzilay and Elhadad focus on word sense disambiguation. They find fault with Hirst and St. Onge for considering only the previous words when disambiguating a word, and point out that in some cases it is necessary to look at following words to disambiguate a word properly. To address this problem, they consider every possible interpretation of the candidate words, where an interpretation is an assignment of a single sense for every candidate. Each interpretation has a score that is the sum of the scores of all the relations, using the relation scores listed above. Consider the set of candidate words $< grip, racket, noise >$. *Grip* has seven noun senses, *racket* has four noun senses, and *noise* has six, which gives a total of $7 \times 4 \times 6 = 168$ possible interpretations. One possible interpretation is "the appendage to an object that is designed to be held" for *grip*, "a sports implement" for *racket*, and "sound that lacks musical quality" for *noise*. This interpretation has a score of four, as the only scored relation is meronymy between *racket* and *grip*, as a grip is part of a racket that is used for sports. However, the highest scoring interpretation is found by setting the sense for *racket* to "sound that lacks musical quality", which is synonymous with the sense chosen for *noise*. As Barzilay and Elhadad's relation score for synonymy is 10, this new interpretation has a score of 10.

Once the highest scoring interpretation of the candidate words is found, the scored relations are used to form lexical chains such that if any two words have a non-zero relation score under the chosen interpretation, then those words are connected together in a chain. In the above example, *racket* and *noise* would be chained together, and *grip* would remain unchained.

Analysis Barzilay and Elhadad's lexical chaining algorithm has one significant drawback: its time and space complexity is exponential in the number of polysemous candidate words in a document. If a document contains n polysemous candidate words, then the number of possible interpretations is bounded below by 2^n. Barzilay and Elhadad do prune interpretations with low scores while generating all possible interpretations, but this only saves space, not time, as all possible interpretations still need to be generated. Generally speaking, algorithms with exponential runtimes are considered infeasible. Silber and McCoy [29] describe Barzilay and

Elhadad's algorithm as "impossible" to run on "documents of any reasonable size". They propose a lexical chaining algorithm with linear complexity as an alternative to Barzilay and Elhadad. Galley and McKeown's [7] proposed lexical chaining algorithm does not have an exponential complexity either, and it considers all candidate words in the document when disambiguating each candidate word in a manner similar to Barzilay and Elhadad's algorithm. Furthermore, Galley and McKeown demonstrate that their algorithm performs word sense disambiguation significantly better than Barzilay and Elhadad's lexical chainer.

3.3 Silber and McCoy

Silber and McCoy [27] [28] [29] use lexical chains for document summarization in the same manner as Barzilay and Elhadad [3]. Where the two approaches differ is how the lexical chains are formed. Silber and McCoy point out that, in most cases, the exponential runtime of Barzilay and Elhadad's lexical chaining algorithm makes it infeasible. To address this problem Silber and McCoy designed a lexical chaining algorithm with a linear runtime.

The lexical chaining algorithm we use for our experiment is based on Silber and McCoy's algorithm and is described in Chapter 5. Consequently, we only describe Silber and McCoy's algorithm briefly here, leaving the details of their algorithm as well as how our algorithm differs to the later chapter.

Silber and McCoy use only nouns as candidate words. They determine which words are nouns using a part-of-speech tagger. Lexical chains are defined in terms of their topics. Each chain has an overriding sense or topic, which is a noun sense in WordNet that is semantically related to all the words in the chain.

The lexical chaining algorithm makes two passes through the document. On the first pass, each possible chain in the document is created. These possible chains are called *metachains*. To create the metachains, each candidate word is placed into every metachain whose overriding sense is possibly related to the candidate. For example, *bat* would be placed into every metachain whose overriding sense is related to the baseball-bat sense of *bat*, as well as every metachain whose overriding sense is related to the flying-mammal sense.

The second pass consists of selecting a single metachain for each candidate term. For each candidate term, the metachain it contributes to most is selected, where the contribution is based on the semantic relation between the candidate term and the other term in the metachain to which the candidate is most strongly related.

The candidate term is then removed from all metachains other than the one it contributes to most. Selecting a single chain for a term in effect disambiguates the term by selecting the sense of the term that is related to the overriding sense of the selected chain.

Analysis When disambiguating each candidate word, Silber and McCoy's algorithm selects the metachain to which the candidate contributes the most. The only semantic relation used in calculating the contribution is the strongest relation between the candidate and the other words in the metachain. Therefore, other related words in the metachains have no effect on which metachain (and sense) is chosen for the candidate (we explain this problem in more detail in Chapter 5). In only considering the most closely related word when choosing a chain for a candidate word, Silber and McCoy's algorithm is similar to Hirst and St. Onge's. We modify Silber and McCoy's algorithm so that contributions are computed as sums of all the semantic relations between the candidate and the other words in a metachain, and demonstrate that this approach performs word sense disambiguation significantly more accurately than Silber and McCoy's approach. However, Silber and McCoy's method for computing the contribution is necessary for a linear runtime. Our method has a polynomial runtime, which is still completely feasible even for large documents.

3.4 Galley and McKeown

Unlike the other authors we review, Galley and McKeown [7] do not use their lexical chaining algorithm for any application. Instead they focus on the relation between word sense disambiguation and lexical chaining performance. They argue as we do, that improved word sense disambiguation leads to better lexical chains and that word sense disambiguation should be performed before lexical chains are formed. However, they are committed to using only semantic relations to perform disambiguation. They develop a lexical chaining algorithm with a focus on performing word sense disambiguation accurately, and evaluate its disambiguation accuracy in comparison to Barzilay and Elhadad's and Silber and McCoy's algorithms.

Galley and McKeown use only nouns as candidate words. They run their algorithm only on documents that are already tagged with parts-of-speech, so they can determine which words are candidates without using a part-of-speech tagger. The algorithm builds representations of all possible interpretations of the candidate words in a manner very similar to Silber and McCoy's algorithm. For each possible sense s of each candidate word, an array containing all the other candidate words that have senses related to s is created. Senses are only considered related if they are identical, hypernyms/hyponyms, or siblings. The relations between senses are assigned scores based on the type of relation and distance between the words in the document.

The algorithm then disambiguates each different word in the document, rather than each word instance. That is, if the word *bank* appears multiple times in the document, then all instances are disambiguated collectively to the same sense, rather than disambiguating each instance of *bank* on its own. Disambiguation of a word w is performed by looking at the array entries for each possible sense of w and summing the relation scores of all the words in the document that are related to that sense. Even after a single sense is chosen for a word, the other possible senses of that word are still considered when disambiguating later words. For example, if *orange* and *blue* both occur in a document, the colour sense of *orange* is always considered when disambiguating *blue*, even if *orange* has already been disambiguated as referring to the fruit, rather than the colour. This helps prevent disambiguation mistakes from propagating and causing more mistakes. In the above case, if *orange* has been incorrectly disambiguated as referring to the fruit when it actually refers to the colour, that error will not affect how *blue* is disambiguated.

Once disambiguation has been completed, the lexical chains are formed in the same way as Barzilay and Elhadad by connecting all semantically related words together.

Galley and McKeown conclude their paper by comparing the word sense disambiguation accuracy of their lexical chaining algorithm with Barzilay and Elhadad [3] and Silber and McCoy [29]. Galley and McKeown show that the senses selected for candidate words by their algorithm are significantly more accurate than the senses chosen by both Barzilay and Elhadad's aglorithm and Silber and McCoy's algorithm.

Analysis As mentioned above, Galley and McKeown make the same argument we do: word sense disambiguation should be performed prior to building lexical chains. However, they provide no experimental evidence to support their claim. Their evaluation shows that the word sense disambiguation accuracy of lexical chaining algorithms can be improved, but Galley and McKeown's approach remains committed to using simple semantic relations (i.e., only hypernyms, hyponyms, and siblings) for performing disambiguation. As we demonstrate in the next chapter, if word sense disambiguation is separated from the lexical chaining process entirely, then any method for disambiguation can be used, and disambiguation performance can be improved independent of how the lexical chains are built..

Galley and McKeown's algorithm has a linear runtime, despite considering all related candidate words when disambiguating a word. This very efficient runtime is achieved because the algorithm disambiguates each different word, rather than each word instance. The number of distinct candidate nouns that can appear in a document is bounded above by the number of distinct noun words in WordNet, which is 114648 for version 2.0 [1]. It should be noted that most documents contain far fewer distinct nouns than 114648, and that the runtime of the algorithm is $O(m \times n)$, where m is the number of distinct candidate nouns in the document.

3.5 Stokes

Stokes [31] uses lexical chains to perform headline generation (a specialized form of document summarization), and topic detection and tracking. Topic detection and tracking determines where topics and stories begin and end in broadcast news transcripts, and is very similar to the detection of linguistic segments performed by Morris and Hirst's original lexical chaining algorithm [21].

Stokes's lexical chaining algorithm is a small extension of Hirst and St. Onge's algorithm [11]. Stokes uses the statistical word associations (explained below) as a possible semantic relation between words, in addition to the extra-strong, strong, and medium-strong relations used by Hirst and St. Onge. This statistical relation is given the lowest priority, and is only used to place a word in a chain if that word is not already related to a chain by Hirst and St. Onge's original relations.

Statistical word associations are based on patterns of word usage in a large corpus. These associations are determined by the degree of similarity between the contexts in which words appear. Words that appear in similar contexts are said to be *distributionally related*. Mohammad and Hirst [19] describe the notion of distributionally related as follows: "Distributionally related words tend to be semantically related, where two words (w_1 and w_2, say) are said to be distributionally related if they have many common co-occurring words and this set of co-occurring words is not restricted to only those that are related to w_1 and w_2 by the same syntactic relation." There are a number of different ways of measuring statistical word associations. We omit the details of how Stokes computed the word associations for their lexical chaining algorithm as they are fairly complicated and unimportant for our work.

Analysis Statistical word associations are useful because they can be used to measure the nonsystematic semantic relations that Morris and Hirst found troublesome. As well, they provide a way of solving the tennis problem in WordNet that we described in Chapter 2. It may appear that statistical word associations are superior to WordNet-based measurements of semantic relatedness, but they have one significant weakness: they only score relations between words, not word meanings. As a result, the semantic score between *bat* and *wing* does not differ based on which sense of *bat* is used. This lack of distinction between senses occurs because the corpora available for statistical analysis are not tagged with WordNet senses. As a result, the statistical word associations can only show that the word *bat* often appears with the word *wing* as well as with the word *ball*, with no difference noted between the two instances of *bat*. To calculate accurate statistical word sense associations would require a large corpus in which the words have been tagged with their correct sense from WordNet. Such corpora are exceedingly difficult to create ([14] describes the trouble involved in such an endeavour). Stokes does not provide a formal evaluation of how using statistical word associations affected the lexical chains created.

We find it odd that Stokes chose to adapt Hirst and St. Onge's lexical chaining algorithm. As we pointed out, multiple evaluations have found fault with Hirst and St. Onge's medium-strong semantic relatedness measure ([4] [10] [23]). Hirst and Budanitsky found Hirst and St. Onge's lexical chaining algorithm as a whole unsuited to the application for which they originally created it, malapropism detection [10]. Stokes argues that the criticism of Hirst and St. Onge's algorithm, particularly with regards to word sense disambiguation, is unfounded. Nonetheless, we remain skeptical of the lexical chainer until an experiment comparing its performance on a specific task (such as headline generation or document summarization) with other lexical chaining algorithms is performed.

Chapter 4

An improved model for lexical chaining

In this chapter we propose a new model for performing lexical chaining. Under our model, word sense disambiguation is performed prior to lexical chaining. The central claim of our thesis is that performing word sense disambiguation prior to lexical chaining improves the correctness of the lexical chains produced. This chapter also describes our experiment that tests the validity of our proposed model.

All the lexical chaining algorithms discussed in the previous chapter use semantic relations to perform word sense disambiguation by choosing the sense for each candidate word that is most strongly semantically related to the candidate words near it in the text. This word sense disambiguation is performed as part of the lexical chaining process. Galley and McKeown [7] note the importance of accurate word sense disambiguation to producing correct lexical chains, and it is this relationship between word sense disambiguation and lexical chaining that is the focus of our research.

Galley and McKeown's lexical chaining algorithm is particularly interesting because it completely disambiguates all candidate words before creating any chains. The chains it builds are then determined by the semantic relations between the senses selected for the candidate words. However, other possible senses of the polysemous candidate words have no affect on the lexical chains formed after the disambiguation step is complete. Therefore, the disambiguation of the candidate words could be performed in a manner completely unrelated to the rest of the lexical chaining algorithm.

Correctly disambiguating a word does not guarantee that it will be chained correctly. An imperfect measurement of semantic relatedness may judge two unrelated senses as being related or two related senses as not being related (below we show how WordNet-based semantic measurements can exhibit both these problems). Nonetheless, with an imperfect semantic relatedness measurement, increased accuracy in word sense disambiguation will still lead to better lexical chains if the following property holds:

Correct Sense Property: There exists no word w such that choosing the correct sense for w would lead to it being chained incorrectly and choosing an incorrect sense for w would lead to it being chained correctly.

Intuition suggests that this property should hold. Even if a semantic relatedness measurement does not recognize that the flying-mammal sense of *bat* is related to wings and echolocation, that same measurement should not recognize a relationship between the baseball sense of *bat* and wings or echolocation. That is, choosing the wrong sense for a word should not cause the word to be chained with words to which it is related if choosing the correct sense for that word would not cause it to be chained with those related words. **It should never be the case that choosing the wrong sense for a word leads to better lexical chains.**

Based on the above reasoning, we make the following two claims:

Claim 1: The correctness of the lexical chains produced by a lexical chaining algorithm is directly related to the accuracy of the word sense disambiguation performed as part of or prior to the lexical chaining algorithm. That is, improved word sense disambiguation necessarily leads to improved lexical chains.

Claim 2: Word sense disambiguation should be performed separately from building lexical chains, using the most accurate method available.

These claims lead to the following model for lexical chaining:

1. Select the candidate words in the document.

2. Perform word sense disambiguation on the candidate words with the most accurate disambiguation algorithm available.

3. Connect the disambiguated words into chains based on the semantic relations between their disambiguated senses.

To investigate the validity of our approach, we propose an experiment evaluating the differences in the chains produced by a lexical chaining algorithm when perfectly accurate word sense disambiguation is performed beforehand. That is, the lexical chainer will first be run on a set of documents where the candidate words have not been disambiguated, and then the lexical chainer will be run on the same set of documents, but with candidate words correctly disambiguated by humans, such that no senses other than the correct senses can be used to form lexical chains.

Such an experiment requires four things:

1. A lexical chaining algorithm.
2. A corpus of documents.
3. The correct senses for all the candidate words in the corpus documents.
4. A method for evaluating the lexical chains produced.

For a lexical chaining algorithm we developed our own as a slight variation of Silber and McCoy's lexical chaining algorithm [27] [28] [29] (discussed in detail in the next chapter). We chose Silber and McCoy's algorithm because it is fairly easy to implement and the overriding senses it uses in building chains are useful for our evaluation. Unfortunately, the difficulty of evaluating lexical chains (explained below) prevents us from testing other lexical chaining algorithms as well. However, our results should apply to other lexical chaining algorithms as well, as we are only concerned with the lexical chains produced by the lexical chaining algorithm, not with the details of how those lexical chains are produced. Since all lexical chaining algorithms must deal with polysemous words, all of them should benefit from prior, accurate word sense disambiguation.

The documents we use in our experiment are from the SemCor corpus [14]. We use this corpus because all its documents have been manually disambiguated. Every noun, verb, adjective, and adverb in the corpus has been tagged with its correct sense in WordNet, giving us perfectly disambiguated documents to test the lexical chainer.

Evaluation is in many ways the trickiest part of our experiment. Most lexical chaining algorithms are evaluated by testing their performance on a specific application that can be machine-evaluated, such as document summarization [3] [29] [31] or malapropism detection [11]. However, Galley and McKeown [7] observe that this is not an accurate evaluation of the lexical chains produced as performance on the task is dependent not only on the quality of the lexical chains produced, but also the general appropriateness of lexical chains to the application. Galley and McKeown use word sense disambiguation accuracy as an application-independent evaluation of lexical chaining performance, but this metric provides no assessment lexical chains produced. Unfortunately, at this point there is no automated method for evaluating the quality or correctness of lexical chains. The problem is that semantic relatedness measurements to date are far from perfect, and thus there is no computational method for correctly deciding whether two words are related such that they belong together in a chain. Only humans can make this decision, and therefore we will manually evaluate the lexical chains produced in our experiment.

Human evaluation of lexical chains is problematic for two reasons: it is very time-consuming and it is subjective. We reduce the time required by only looking at a small subset of the lexical chains produced under the two conditions (undisambiguated input and disambiguated input). Specifically we choose the longest chains in a document to look at because we believe they are the most representative of the most dominant/important topics in the document. Smaller chains occur much more frequently than long chains in documents, making the short chains less useful for determining the topic structure of a document. As well, their commonality lessens the impact of making an error in a small chain. Errors in the longest chains of a document can lead to subsequent errors in interpreting the document's meaning, particularly in document summarization applications. Deviations in the resulting topic structure can be caused by making a chain longer than it should be or by missing related words that are relevant to a chain: this omission is especially significant if the chain would have otherwise been a longest chain. Focusing on this specific subset of chains should reduce the subjectivity of our results because the judgements we make will largely be comparisons between chains, rather than general judgements about the chains in isolation. In the following chapter we explain how we judge chains in more detail and list more specific was by which we reduce subjectivity in our observations.

In evaluating the correctness of lexical chains, there are two types of possible errors that can occur:

1. A semantically related word is incorrectly omitted from a chain.
2. An unrelated word is incorrectly placed in a chain.

An example of the first error is the word *subtraction* being left out of a chain containing related words such as *addition*, *multiplication*, and *arithmetic*. If an unrelated word such as *giraffe* were added to the chain with *addition*, *multiplication*, and *arithmetic*, then that would be an occurrence of the second error.

Both of these errors can be caused in two different ways: a mistake in word sense disambiguation or a mistake in measuring semantic relatedness. A lexical chaining error due to word sense disambiguation occurs when:

(a) A word is omitted from a chain because an incorrect sense was chosen for it. In this case it would have been connected to the chain if the correct sense had been chosen.

(b) A word is connected to a chain because an incorrect sense was chosen for it. Here, the word would not have been connected if its correct sense had been chosen.

Consider the word *ball*. If, in the context of a document, *ball* refers to a formal dance, then it is semantically related to the word *dance*. Therefore, if the word *dance* also occurs in the document, it should be chained with *ball*. However, if *ball* is incorrectly disambiguated as referring to a round object (such as a tennis ball), then it will probably not be chained with *dance*, which is an error of type (a). Furthermore, if the same instance of *ball* is incorrectly placed in a lexical chain with the word *sphere* (they should not be related because *ball* refers to a formal dance in this case), then this is an instance of error (b).

Errors due to imperfect semantic relatedness measures can cause a word to be incorrectly omitted from a chain if the measurement does not recognize the word's relation to the chain. Such measures can also cause a word to be incorrectly added to a chain if the semantic relation between the word and chain is scored overly high, effectively denoting a semantic relation where one does not exist.

In the next chapter we describe in more detail how these chaining errors are recognized in our experiment and used in determining our results.

Chapter 5

Experiment

To test the effect of improved word sense disambiguation accuracy on lexical chaining performance, we will compare the chains created by a lexical chaining algorithm in two different cases. In the first case, the lexical chainer will be run on a corpus of documents with no information about the senses of the terms in the documents other than the knowledge that all are nouns and the list of all the possible senses in WordNet [1] for these terms. These are the circumstances under which lexical chaining algorithms are intended to be run, where the intended meaning of each term in its specific context must be determined by the algorithm. In the second case, we pass only the correct sense of each term to the lexical chainer. In this case, terms can only be placed into chains that relate to their correct sense. This is equivalent to running a word sense disambiguation algorithm with 100% accuracy on the documents before running the lexical chaining algorithm.

To perform our experiment we require a lexical chaining algorithm and a corpus in which all the nouns have been tagged with their correct sense from WordNet [1]. For the lexical chaining algorithm we implement a variation of Silber and McCoy's lexical chaining algorithm ([27], [28], [29]) and for our corpus we use the SemCor corpus [14]. Both the algorithm and the corpus are described below.

5.1 Our lexical chaining algorithm

Our lexical chaining algorithm is an implementation of Silber and McCoy's lexical chaining algorithm ([27], [28], [29]). However, we made some changes to their algorithm, which are noted below.

The Silber and McCoy algorithm makes two passes through the candidate terms in a document. On the first pass, each candidate term is placed into every chain to which it could possibly belong (these possible chains are called *metachains*). On the second pass, each term is removed from every metachain except the one to which it is most strongly related. Chains left with only a single term are discarded, and the remaining chains become the lexical chains for the document.

5.1.1 The first pass

With the Silber and McCoy lexical chaining algorithm, every metachain has an overriding sense, which is a sense from WordNet that is related to all the terms placed in the chain. For two senses to be considered related, they must be either the same sense, a parent–child pair, or siblings (children of the same parent). To keep track of all the metachains, a hash table is constructed where the keys are the index values of noun senses in WordNet (every noun sense has a unique index) and the values are lists of the terms in that metachain. A sample metachain table is shown in Figure 5.2. WordNet has a total of 114648 unique noun senses, which is an upper bound on the number of possible metachains in any document.

```
for all candidate words w in the document do
  for all senses s of w do
    for all children c of s do
      Insert w into metachain hashtable at index c
    end for
    for all parents p of s do
      Insert w into metachain hashtable at index p
      for all children c of p do
        Insert w into metachain hashtable at index c
      end for
    end for
  end for
end for
```

Figure 5.1: Algorithm for metachain creation

In the first pass, each term is added to every metachain to which it is related. This is done by looking at each possible sense s for a term and adding that term to the metachains with the same index of s, a parent or child of s, or a sibling of s. The algorithm for this first pass is shown in Figure 5.1.

Metachain index	Overriding sense	Chain terms	
4703162	chromatic color: “a color that has hue”	orange	blue
4707837	orange: “range of colors between red and yellow”	orange	blue
4711153	blue: “the color of the clear sky in the daytime”	orange	blue
7234431	edible fruit:	apple	
7267116	apple: “fruit with red or yellow or green skin”	apple	
7275039	citrus fruit	orange	apple
7275573	orange: “round yellow to orange fruit”	orange	

Figure 5.2: Sample metachain table

Figure 5.2 shows entries excerpted from a sample metachain table created for a document containing the terms *orange*, *apple*, and *blue*, in that order. The descriptions of the various senses are excerpted from their WordNet glosses [1]. First the term *orange* is processed. There are five different noun senses for *orange* in WordNet. Two of these senses are used in creating the table, one referring to the colour orange (WordNet index 4707837) and the other referring to the fruit (index 7275573). *Orange* is added to the metachains at 4707837 and 7275573 because *orange* is one of the synonyms for both those senses. *Orange* is also added to the metachain at 4703162 because that sense of *chromatic color* is a parent of sense 4707837. *Orange* is then added to the metachain at 4711153 because that sense of *blue* is a sibling of sense 4707837 via their shared parent 4703162 (i.e., both *orange* and *blue* are colours). Finally, *orange* is added to the metachain at 7275039 because its sense 7275039 is a child of sense 7275039 (“any of numerous fruits of the genus Citrus having thick rind and juicy pulp”). *Blue* is added to the metachains at 4711153, 4703162, and 7275573 because these senses share an identity, parent, and sibling relation respectively with the “the color of the clear sky in the daytime” sense of *blue*. *Apple* is added via identity, child, and sibling relations to the metachains at 7267116, 7234431, and 7275039 respectively. *Apple* and *orange* are not added to each other’s senses (7267116 and 7275573) as they do not share a parent-child or sibling relation, but they still end up in the same metachain at 7275039 because they are both related to citrus fruit.

To summarize, *orange* has been added to the metachains at 4703162, 4707837, and 4711153 based on one of its senses (the colour) and metachains at 7275039 and 7275573 because of another sense (the fruit). When a single metachain is chosen for *orange* in the second pass, it will determine a single final sense for the term.

5.1.2 The second pass

Once all the metachains are created, the lexical chaining algorithm makes a second pass through the candidate terms in the document. For each metachain to which a given term t belongs, t's contribution to that chain is computed based on the semantic and distance relations between t and all other terms in the chain, where distance refers to how far apart the terms are in the document. The term t is then removed from every metachain except for the metachain to which t contributes the most. In the event of a tie among metachains for the maximum contribution, the metachain with the lowest sense index is chosen. The algorithm for the second pass is shown in Figure 5.3.

In Silber and McCoy's first two papers describing their lexical chaining algorithm ([27] and [28]), they did not specify which metachain to choose for a term if there was a tie among metachains. In a later specification of their algorithm [29], Silber and McCoy choose the metachain with the highest overriding sense index (as opposed to the lowest index which we use). The only justification given for this choice is that "WordNet is organized with more specific concepts indexed with higher numbers." The more specific concepts to which they refer are senses with greater depths in the noun hierarchy. That is, for any two senses s_1 and s_2, if s_2's depth (distance from its closest ancestor) is greater than s_1's depth, then s_2 will have a larger index. While favouring metachains with more specific overriding senses may reduce the number of overly broad chains formed, it may also prevent some legitimate chains. Our algorithm picks metachains with more general overriding senses to allow for larger chains, which we believe are more representative of the overall subject of a document.

The score function we use to measure the strength of the relation between any two terms in a document is given in Figure 5.4. This is the same score function used by Silber and McCoy [29]. Silber and McCoy state that these values were arrived at by "empirical testing", but do not provide any details about the experiments used to derive the values. This score function differs from a semantic relatedness measure in that it also takes into account the location of the terms in the document, whereas a semantic relatedness measure is only between word senses.

We compute a term t's contribution to a metachain by summing the scores between t and every other term in the metachain. When t was added to the metachain, it was because one of its senses, s, is related to the overriding sense for that metachain. This sense s is the sense used to compute the score between t and the other terms in the metachain. In the previous example (Figure 5.2), sense 4707837 of *orange* (the colour) would be used for computing the contribution of

```
for all candidate words w in the document do
  //First collect the indexes of chains that w is in
  Set neighbours = {}
  for all senses s of w do
    for all children c of s do
      Insert c into neighbours
    end for
    for all parents p of s do
      Insert p into neighbours
      for all children c of p do
        Insert c into neighbours
      end for
    end for
  end for
  //Now determine which chain w contributes to most
  Set max_contribution = 0
  Set max_index = −1
  for all senses s in neighbours do
    Set chain to the metachain at index s
    if Contribution(w, chain) > max_contribution then
      Set max_contribution = Contribution(w, chain)
      Set max_index = s.index
      //In the event of a tie, we pick the chain with the lowest overriding sense
    else if Contribution(w, chain) == max_contribution ∧ s.index < max_index then
      Set max_contribution = Contribution(w, chain)
      Set max_index = s.index
    end if
  end for
  // Now remove w from all other metachains
  for all senses s in neighbours do
    if s.index ≠ max_index then
      Remove w from the metachain at s
    end if
  end for
end for
```

Figure 5.3: Algorithm for chain selection

	Same sentence	Within three sentences	Same paragraph	Default
Same synset	1	1	1	1
Parent or child	1	0.5	0.5	0.5
Sibling	1	0.3	0.2	0

Figure 5.4: Silber and McCoy term–based score function [29]

orange to the metachains at 4703162, 4707837, and 4711153 because 4707837 is the sense related to these metachain senses in WordNet. Sense 7275573 *orange* (the fruit) would be used for the other metachains in which *orange* appears because it is related to their overriding senses (citrus fruit and "round yellow to orange fruit").

This method for computing the contribution of a term to a metachain differs from the method used by Silber and McCoy. In their first two descriptions of their algorithm ([27], [28]), functions for determining scores between terms are given which are earlier versions of the function in Figure 5.4. However, these functions are not accompanied by any description of how they are used to calculate the contribution of a term to an entire chain. Silber and McCoy's later paper [29] specifies that whenever a term t is first added to a metachain, t's contribution is the relation score between it and the term in the metachain closest to t in the document. We consider this a poor method for computing a term's contribution as it renders a term insensitive to all terms in a metachain, other than the closest one. For example, consider a document containing multiple instances of the terms *orange*, *blue*, and *grapefruit*. If there are two instances of *orange* very close together, then both will have a contribution of 1 for every metachain in which they are placed, regardless of the number of instances of *blue* or *grapefruit* in the document. None of the non-*orange* terms will be used to disambiguate *orange* if only the closest term is considered. To verify our choice of contribution measure, we compared the word sense disambiguation performance of the lexical chainer using the two different contribution measures. We used the same corpus as in our experiment. We found that the accuracy of word sense disambiguation was 42.9% using the closest term measure of Silber and McCoy, compared to 52.1% accuracy for our measure.

As an example of how a term's contribution to a metachain is used, consider a document in which *orange* and *blue* appear in the same sentence, while the next sentence contains *apple* and two occurrences of *grapefruit* (as in "large yellow fruit", sense index 7277914). Figure 5.5 shows the metachain table for this example, the same table as in Figure 5.2, but with *grapefruit* added. To select a single metachain for *orange*, all the metachains to which it belongs are scored (i.e., the metachains at 4703162, 4707837, 4711153, 7275039, 7275573, and 7277914). The contribution

Metachain index	Chain terms			
4703162	orange	blue		
4707837	orange	blue		
4711153	orange	blue		
7234431	apple			
7267116	apple			
7275039	orange	apple	grapefruit	grapefruit
7275573	orange	grapefruit	grapefruit	
7277914	orange	grapefruit	grapefruit	

Figure 5.5: Chain selection example

of *orange* to the metachains at 4703162, 4707837 and 4711153 is 1, as the only other term in those metachains, *blue*, is a sibling of the sense of *orange* at index 4707837 and both terms are in the same sentence. The score between *orange* and each instance of *grapefruit* is 0.3, as *grapefruit* is a sibling of the sense of *orange* at index 7275573 via the shared parent *citrus fruit* at index 7275039 and *grapefruit* and *orange* are one sentence apart (i.e., within three sentences). So the contribution score of *orange* to the metachains at indexes 7275039, 7275573 and 7277914 is 0.6 (0.3 for each instance of *grapefruit*). The *apple* term does not affect the contribution of *orange* to the 7275039 metachain because sense 7275573 is not identical to sense 7267116, nor is it a parent, child, or sibling of 7267116. Therefore the highest scoring metachains for *orange* are those related to colour at 4703162, 4707837 and 4711153. To break the tie between those chains, the metachain at 4703162 is chosen because it has the lowest index. *Orange* is removed from all other metachains and the pass continues to select a single chain for the rest of the senses. *Orange* has now been disambiguated to refer to the colour orange, as that is the sense of orange that is related to the overriding sense for its chain, 4703162 "chromatic color".

5.1.3 Runtime

We do not provide a detailed proof of our algorithm's runtime here, but we can state that the runtime of our lexical chaining algorithm is $O(n^2)$, where n is the number of candidate terms in the document. Our algorithm must process all n words, and when calculating a word w's contribution, at most $n-1$ other words must be considered for possible semantic relations with w.

An n^2 runtime is perfectly reasonable for most realistic lexical chaining applications. We implemented our lexical chaining algorithm in Java and ran it on a computer with an 800 MHz Pentium III processor and 1 GB of RAM. Using this setup, our algorithm took less than 10 minutes to build all the lexical chains for all of the 102 documents in our corpus.

5.2 Corpus

The SemCor corpus [14] contains 103 documents from the Standard Corpus of Present-Day Edited American English (commonly known as the Brown Corpus), plus Stephen Crane's novella *The Red Badge Of Courage* in its entirety. The Brown Corpus is described in detail in Francis and Kucera [6]. All words in the SemCor corpus that are also present in WordNet are tagged with their correct sense in WordNet. Multiple versions of the corpus exist, each corresponding to the version of WordNet that was used for tagging. We used the WordNet 2.0 version of the SemCor corpus, available online at http://lit.csci.unt.edu/ rada/downloads/semcor/semcor2.0.tar.gz. The words were disambiguated by linguists, and are widely considered to be a gold standard for word sense disambiguation [13].

We use only 102 documents of the Brown Corpus portion for our experiment (one document from that portion, br-j56, was omitted due to errors in parsing it). Figure 5.6 is a breakdown of the documents we used in terms of content. The average number of words per document is 1932, with the shortest document having 1764 words and the longest 2066 words. Appendix A lists all the documents we used by name, along with their lengths in words and their categories.

The documents have been marked up in SGML. A detailed description of the document format can be found in Landes et al. [14]. Each document consists of a number of ordered terms, and each term is surrounded by a tag which contains multiple attributes describing the term. We use the following attributes:

- Sentence: A positive integer indicating in which sentence the term occurs.
- Paragraph: A positive integer indicating in which paragraph the term occurs.
- Lemma: A string indicating the base form of the term. For example, the lemma for *running* and *ran* is *run*. The lemma is used to find the possible senses of the word in WordNet.

  In the SemCor corpus, the lemma given for a proper noun is the noun itself if it appears as an entry in WordNet; for example, *atlanta*, referring to the city of Atlanta, is in WordNet. For a proper noun that does not appear in WordNet,

- 7 News articles from newspapers and periodicals.
- 2 Editorial articles from newspapers and periodicals.
- 3 Reviews from newspapers and periodicals.
- 4 Essays on religion from newspapers and periodicals.
- 6 Articles on hobbies, recreation and skills from newspapers and periodicals.
- 4 Articles on history and folklore from newspapers and periodicals.
- 3 Social commentary articles from newspapers and periodicals.
- 1 Government report.
- 32 Academic journal papers.
- 35 Pieces of fiction, both short stories and novel excerpts.
- 5 Humorous essays.

Figure 5.6: The brown1 section of the SemCor corpus

the noun in WordNet that best describes it is used. For example, the lemma for *Fulton_County_Grand_Jury* is *group* (SemCor document br-a01).

- Part of speech (POS): A string indicating to which part of speech the term belongs. We are only concerned with nouns, which are indicated by the POS strings NN, NNP, NNPS, NNS, NP, and NPS. In WordNet, no distinction is made between the various types of nouns; all types occur together in the noun hierarchies.
- WordNet sense index: A positive integer indicating the correct sense of the term in WordNet. This number, along with the lemma and POS for the term, uniquely identifies a synset in WordNet. As this sense is provided for all nouns in our corpus, all the nouns are perfectly disambiguated.

When a document is read in, it is converted into an array of Term objects which we refer to as $Terms[]$. $Terms[i]$ refers to the i^{th} term of the document. Each Term object has the following parameters: sentence, paragraph, lemma, POS, sense, and index. Sentence, paragraph and lemma are the same as in the tagged document file. POS, sense and index are as follows:

- POS: The part of speech for the Term. Possible values are: *noun*, *verb*, *adjective*, and *adverb*, which are the same parts of speech used by WordNet.
- Sense: The sense in WordNet of the Term.
- Index: A positive integer denoting the position of the Term in the document. $Terms[i]$ has index i.

The candidate terms for chaining in a document are the Terms with POS noun.

A Term can be made ambiguous by ignoring the WordNet sense-index attribute from the document. Instead, the lemma and POS for the term are used to retrieve all possible senses for the word from WordNet. The sense for the Term then becomes a set containing all the possible senses. When our lexical chaining algorithm is performed on ambiguous Terms, it selects a single sense for every Term that is placed in a chain. In effect, the lexical chaining algorithm disambiguates every document term that is put into a chain.

5.3 Method

```
1: for all documents d in the corpus do
2:    Set candidates := findCandidates(d)
3:    Set perfectChains := findChains(candidates)
4:    Set ambiguous := copy(candidates)
5:    for all Terms t in ambiguous do
6:       Set t.sense := WordNet.getSenses(t.lemma, t.POS)
7:    end for
8:    Set classicChains := findChains(ambiguous)
9: end for
```

Figure 5.7: Algorithm for computing perfect and classic chains

Figure 5.7 presents our algorithm for finding the perfect and classic chains for all the documents in the corpus. Line 2 selects the candidate terms, the nouns, from the document. Line 3 runs the lexical chaining algorithm on the perfectly disambiguated terms, resulting in a set of chains. Lines 4 through 6 copy the candidate terms and make them ambiguous by setting each term's sense to the set of all possible noun senses for that term in WordNet. Therefore, in line 8 when the chainer is run on the ambiguous terms, it must select a single sense for each candidate term. When the experiment is finished, if $Candidates[i].sense \neq Ambiguous[i].sense$ for any i, then the algorithm selected the wrong sense for that term.

5.3.1 The output of our experiment

At the completion of chain creation, $Perfect[]$ is the set of the chains created when the lexical chaining algorithm is run on perfectly disambiguated terms, whereas $Classic[]$ is the set of chains created when the chainer is run on ambiguous or undisambiguated text. Each lexical chain is a set of Term objects. When comparing Terms inside $Classic[]$ and $Perfect[]$, we say that two Terms are the same if they have the same index. For example, if $Classic[i].index = Perfect[j].index$, then $Classic[i]$ and $Perfect[j]$ refer to the same term in the original document, represented by $Terms[Classic[i].index]$. These Terms may still differ with regards to their sense, i.e., if the lexical chainer did not pick the correct sense for $Classic[i]$.

The *size* or *length* of a chain is the number of Terms it contains. The order of the Terms in a chain does not matter. The first Term of a chain refers to the Term in the chain that occurs first in the document. Correspondingly, chains have a last Term that occurs last in the document. The *span* of a chain is the distance from its first Term to its last, which can be measured in sentences or paragraphs.

We refer to a chain that belongs to $Perfect[]$ as a *perfect chain*, and a chain that belongs to $Classic[]$ as a *classic chain*.

5.4 Comparing chains

For each document we will generate two sets of lexical chains, the perfect chains and the classic chains. From each set of chains we will pick the three longest chains as the most 'important' chains for the document and compare them. We refer to these as the *longest classic chains* and *longest perfect chains*. In the event of ties for the longest three chains, we will pick all tying chains. Formally, the set of longest chains is defined as:

Let $C = \{c_1, c_2, ..., c_n\}$ be a set of chains in a document, where every chain is a set of terms from that document. The set of longest chains for the document is the smallest set of chains $L = \{c_{l_1}, c_{l_2}, ..., c_{l_k}\}$ such that:

1. $k \geq 3$

2. For all i, j, if $c_i \in L$, $c_j \in C$ and $c_j \notin L$, then $|c_i| > |c_j|$.

We hypothesize that the average number of longest chains selected by this method will be close to three. If many more chains are selected for a document, then this suggests either that the document has many equally important or dominant topics, or that despite there being only a few dominant topics in the document, the longest lexical chains do not necessarily correspond with these dominant topics.

As decisions about semantic relatedness and lexical cohesion are, to some degree, subjective, we take care to minimize the amount of subjectivity in our analysis. We compare only the longest chains for each document, focusing on the differences between the classic and perfect chains. When a classic and perfect chain are the same, we count both chains as correct without examining them further. When a classic chain and a perfect chain have terms in common, but are not the same, we judge whether the correct meaning of the common terms is being used.

For example, if one of the longest perfect chains in a document contains the terms *bank* and *teller*, then *bank* refers to a financial institution rather than a river bank. Consequently, the chain—by virtue of its overriding sense that is related to both *bank* and *teller*—indicates that the document is about banks as financial institutions. If a longest classic chain for that document contains *bank* and *slope*, then the lexical chaining algorithm has misinterpreted *bank*. We count this kind of longest classic chain as an error.

If a longest classic chain mistakenly adds terms to a longest perfect chain, but still uses the correct meanings for the terms the chains have in common, then we do not count the classic chain as incorrect. For example, consider a longest perfect chain that contains the terms *slope* and *ridge* and a longest classic chain from the same document that contains *slope*, *ridge*, and *bank*. Even if *bank* has been added incorrectly (i.e., misinterpreted as the side of a river when it is actually intended as a financial institution), the meaning of the common terms, *slope* and *ridge*, has been maintained. Therefore the overriding sense for the classic chain is the same or closely linked to the overriding sense for the perfect chain. In both cases, the longest chain indicates that the document is about geological features such as slopes and ridges. As the subject for the document indicated by the longest chain has not changed, we do not count this as an error, despite the incorrectly added terms in the longest classic chain.

Lastly, there is the case of a longest classic chain that does not have any terms in common with any of the longest perfect chains, or vice versa for a longest perfect chain. We count such a chain as incorrect if its longest correct portion would not be included in the set of longest chains for the document if the incorrect terms were removed.

In none of these cases do we consider whether a longest chain is missing terms, i.e., if there are terms in the document that should be part of a longest chain but are not included. Missing terms are unimportant as the chains in question will have already been selected as the longest chains for the document. Ignoring terms that have been omitted from chains is very useful, as checking every document for terms omitted from chains would be extremely time-consuming.

5.5 Hypotheses of this experiment

Hypothesis 1: Given that WordNet links some senses that are only loosely related, it is possible for perfect chains to be incorrect. Despite this, we hypothesize that there will be very few incorrect longest perfect chains.

Hypothesis 2: We hypothesize that accurate word sense disambiguation is necessary for building lexical chains that are correct. As a result, many more longest classic chains than longest perfect chains will be incorrect.

5.6 Results

In this section we present the results of our experiment in detail.

Our lexical chainer, when run on the entire corpus, generates 6663 chains when perfect word disambiguation is used and 6757 chains when there is no prior disambiguation. Figure 5.8 lists the number of classic and perfect chains by chain length. Most of the chains are very small; 4253 of the perfect chains and 3941 of the classic chains contain only two or three terms, and these short chains account for 64% and 58% of the total perfect and classic chains respectively. The average chain length is 4.37 terms for the perfect chains and 4.93 terms for the classic chains.

A combined total of 344 perfect chains and 334 classic chains was selected by our method as the set of longest chains in the corpus documents. This works out to an average of 3.37 long perfect chains and 3.27 long classic chains per document. The longest classic chains were longer than the longest perfect chains for the most part, with an average length of 21.3 terms compared to 17.2 terms for the longest perfect chains. The shortest of the longest chains selected was of length seven for the classic chains and length five for the perfect chains.

Chain Length	Perfect	Classic
2	3004	2652
3	1249	1289
4	699	730
5	428	476
6	286	341
7	203	240
8	161	167
9	112	139
10	74	98
11	66	99
12	48	82
13	42	66
14	33	48
15	40	35
16	20	37
17	21	22
18	20	23
19	19	26
20	21	20
21	14	21
22	11	14
23	10	15
24	12	15
25	10	9
26	5	8
27	5	5

Chain Length	Perfect	Classic
28	4	5
29	2	5
30	1	5
31	7	6
32	2	4
33	3	9
34	2	4
35	1	1
36	3	4
37	3	4
38	3	1
39	0	1
40	3	2
41	0	4
42	4	2
43	1	1
44	1	3
45	2	1
46	1	2
47	1	1
48	1	2
49	1	2
50	0	1
51	0	1
52	1	2
53	0	0

Chain Length	Perfect	Classic
54	0	0
55	0	0
56	0	0
57	0	0
58	0	1
59	0	0
60	0	0
61	0	0
62	0	0
63	0	0
64	0	1
65	0	1
66	1	0
67	0	0
68	0	0
69	0	0
70	0	0
71	0	2
72	1	0
73	0	0
74	0	1
75	0	0
76	0	0
77	1	0
78	0	0
79	0	1

Figure 5.8: Number of lexical chains by length

Two hundred and forty two of the longest perfect chains share at least one term with one of the longest classic chains, whereas 235 of the longest classic chains share at least one term with a longest perfect chain. The reason for the difference between the two numbers is that thirteen longest classic chains contain terms from more than one longest perfect chain, while only six longest perfect chains contained terms from more than one longest classic chain. Chains with common terms account for a majority of the longest chains, forming 70.3% of the longest perfect chains and 70.4% of the longest classic chains. While some of these chains have few terms in common, most are very similar. These intersecting chains fall into four categories:

- Chains that are the same or almost the same, i.e., neither chain contains more than two terms not present in the other.
- Perfect chains that contain most or all of the terms in a longest classic chain plus many terms not in any longest classic chain.
- Classic chains that contain most or all of the terms in a longest perfect chain plus many terms not in any longest perfect chain.
- Chains that overlap such that they have some terms in common, but both chains also contain terms not found in the other.

Forty-four of the longest chains were exactly the same, i.e., both the longest classic chain and longest perfect chain contained exactly the same terms. A further 50 pairs of longest classic and perfect chains we consider to be virtually the same (differing by at most two terms). These chains represent the cases in which prior word sense disambiguation had little or no effect on the longest chains produced by the lexical chaining algorithm.

Only 12 longest perfect chains were longer than the longest classic chains with which they shared terms. Three of these longest perfect chains completely contained a longest classic chain.

In contrast, there were 112 longest classic chains that expanded upon longest perfect chains. In many cases the longest classic chain added many terms to a longest perfect chain. In document a-11, one of the longest perfect chains is {*center, left, right, short, right, center, right, short, center, right, center, short, center*}, which are all terms in the document that refer to baseball positions. One of the longest classic chains expands this chain with more baseball terms: {*second, Catcher, catcher, Shortstop, center, plate, pitchers, mound, short, second, center, third, plate, short, third, second, center, third, center, short, center, shortstop,* *mound, plate, thirds, shortstop*}. The extra terms are added because the lexical chainer chooses senses for some terms that are not correct but are related to each other. For example, the WordNet gloss for the correct sense for *short* in the perfect chain is "the location on a baseball field where the shortstop is stationed" [1] but in the classic chain the sense used is "the fielding position of the player on a baseball team who is stationed between 2nd and 3rd base". This sense allows it to be connected with other position names, such as *shortstop* and *catcher*. The Silber and McCoy relatedness function does not recognize these two senses of *short* as being related, as the shortest path between the two senses in WordNet is too long at twelve senses.

When a classic chain extends a perfect chain, terms may be added correctly or incorrectly. In the document br-a01, the classic longest chain {*election, election, election, election, election, election, election, election, election, election, election, election, primary, reelection, vote, general_election, primary, vote, vote, vote, votes*} correctly extends the perfect longest chain {*reelection, primary_election, election, election, primary, election, election, election, primary, general_election, primary, election, election, election, election, election, election, election*} by adding the terms *vote, vote, vote, votes*, as voting is closely related to elections. An incorrect extension occurs in document br-f10, where the classic longest chain adds *light*, *light* to the longest perfect chain {*machine, applicator, machine, machine, machine, machine, machine, machines, machine, machine, machine, machines, devices, machine, devices, gadgets, machines, gadget, machines, device, device, machines, gadgets, device, machines, devices, devices, device, device, gadgets, devices, machine, machine, gadget, device, machine, device, device, machines, device*}. The correct sense of *light* in the document is "(physics) electromagnetic radiation that can produce a visual sensation; *the light was filtered through a soft glass window*". However, in the classic case, *light* is misinterpreted as "a device for lighting or igniting fuel or charges or fires; *do you have a light?*" and therefore incorrectly added. As both incorrect and correct additions can occur in the same classic chain, and the portion of terms that are incorrect can vary greatly, it is difficult to measure the goodness or correctness of an expanded classic chain. For this reason, we do not distinguish between different types of chain expansion, except in the case described below.

In the previous example, the classic chain maintains the correct meanings of all the terms it shares with the perfect chain it expands (*machine, gadget, device, applicator*). Only the mistake in interpreting *light* causes it to be added incorrectly. The occurrence of *machine* can be explained by studying the expanded chain more closely. It is apparent that *machine* still refers to "any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of human tasks", rather than another sense such as "an efficient person; *the boxer was a magnificent fighting machine*". In some cases, terms from the perfect chain are misinterpreted, leading to a classic chain with a very different overriding meaning than the perfect chain that it contains. For example, the document br-b20 contains {*West, West, West, West, West, West, West*} as a longest perfect chain, where *West* refers to "the countries of (originally) Europe and (now including) North and South America". The terms of that chain are contained within a longest classic chain: {*West, West, London, West, West, commentators, West, writer, London, France, West, West, page*}. In this chain, *West* is incorrectly interpreted as referring to an author ("British writer (born in Ireland) (1892-1983)"). Likewise, *London*, *France*, and *page* have also been incorrectly interpreted as the last names of authors, which are all related to *commentators* and *writer* because all of them are writers. We judged these sorts of chains as incorrect classic chains; they account for 10 of the total 112 longest classic chains that expand upon longest perfect chains.

There are 23 instances of a longest classic chain overlapping a longest perfect chain such that each chain has at least three terms not present in the other. When characterizing these chains, we paid particular attention to how the terms they had in common were interpreted. In 10 cases, senses were chosen for the terms in common that are unrelated to the correct senses. This resulted in a classic chain that is unrelated in meaning to the perfect chain, in the same manner as the *West* chain described above. An example of an incorrect overlap occurs in document br-k16. One longest perfect chain is {*bay, bay, bay, bay, water, New_York_Bay, inlets, sea*}, where the correct sense of *bay* is "an indentation of a shoreline larger than a cove but smaller than a gulf." This chain overlaps with the longest classic chain {*horse, horse, horses, horse, horse, bay, bay, bay, bay*}, where *bay* is incorrectly taken to mean "a horse of a moderate reddish-brown color". These nine longest classic chains were counted as incorrect.

In the case of the other thirteen pairs of overlapping longest chains, the classic chains used either the correct sense or a closely related sense for all the terms the chains have in common. The terms that the classic chain adds may or not be added correctly. When the classic chain does contain extra correct terms, then neither chain contains all the terms that should be chained together. For example, in document br-k03, the longest classic chain {*man, boy, boy, man, boy, boy, white_men, man, man, man, old_man*} overlaps the longest perfect chain {*boy, boy, fellow, boy, boy, fellow, fellow*}. Between the two chains there are a total of 14 unique terms, all of which we consider to be related, forming the chain {*man, boy, boy, man, boy, boy, white_man, man, man, man, old_man, fellow, fellow, fellow*}. In some cases, such as this example, it is possible to pick senses for all the terms that allow them to be placed in a single chain—this requires picking senses for all the terms that are all related to a single overriding sense for that chain. For this example, senses can be selected for all of these terms that are related via WordNet to the sense of *man*, "an adult male person (as opposed to a woman); *there were two women and six men on the bus*".

In some cases, neither overlapping chain contains all the terms that should be chained together, but it is not possible to pick senses for these words such that they are all related to a single overriding sense. In four of the overlapping pairs, despite all the terms being related, it was not possible to pick senses that were all related to a single overriding sense for the terms. In document br-j02, the longest classic chain {*temperature, temperature, temperature, temperature, temperature, temperature, Temperatures, temperature, temperature, temperature, temperature, temperature, temperature, temperature, temperature, temperatures, temperature, temperatures, temperature, temperatures, temperature, temperature, temperature, temperature, temperature, temperature, temperature, temperatures, heat, heat, high_temperature, heat, heat, Heat, heat, heat, heat, heat, heat, heat, heat, heat, enthalpy, enthalpy, reflection, heat, heat, heat*} overlaps the longest perfect chain {*energy, energy, energy, energy, energy, energy, energy, energy, energy, energy, heat, heat, heat, physical_phenomena, heat, work, Heat, conduction, radiation, heat, pressure, heat, heat, heat, conduction, heat, radiation, heat, pressure, heat, heat, radiation, reflection, radiation, heat, heat, heat, radiation*}, with the numerous instances of *heat* being common to both chains. There are 68 total unique terms between the two chains, but whichever senses may be picked for the terms, the longest possible chain is always the classic chain with 48 terms. The classic chain makes the obvious connection between *heat* and *temperature*, but the perfect chain links *heat* with *energy*, *radiation*, and *conduction* because the sense it uses for *heat* is the scientific/theoretical definition, "a form of energy that is transferred by a difference in temperature". In contrast, the sense of *heat* used in the classic chain is the more simplistic "the presence of heat", which allows the connection to *temperature* in WordNet. Despite the two senses of *heat* being related, they are not closely linked in WordNet, so it is impossible to create a chain that contains all the related terms as long as only a single sense is allowed for the *heat* terms.

One hundred and two of the longest perfect chains had no terms in common with any of the longest classic chains, while 99 of the longest classic chains had no terms in common with any of the longest perfect chains: we refer to these chains as *untouched* chains. Most of the untouched longest perfect chains shared some or all of their terms with a classic chain, but the classic chain was not long enough to rank in the top three longest classic chains for the document. Most of the untouched longest classic chains combined or overlapped multiple perfect chains that were not as long as the longest perfect chains in the document.

While it is not possible for any of the perfect chains to have any mistakes due to an incorrect meaning of a term being used, it is still possible for terms to be incorrectly connected in a perfect chain because of incorrect or tenuous links between senses in WordNet. We classify 10 of the untouched longest perfect chains as being incorrect. An example of an incorrect perfect chain formed via tenuous links is {*occurrences, experience, experience, instances, experience, instance, experience, incident, cases, things, cases, break, experiences, instances*} from document br-f03. An overriding sense for this chain is *thing:* "an event; *a funny thing happened on the way to the...*".

We classified 72 of the untouched longest classic chains as incorrect. Mistakes in a classic chain can be caused by either a mistake in word sense disambiguation or in semantic relatedness, and often a combination of both happened in incorrect classic chains. An example of an untouched longest classic chain containing both kinds of mistakes is {*end, end, death, offices, state, state, state, state, state, action, action, action, state, order*} from document br-a01. In this case, the *state* terms have been interpreted as meaning "the way something is with respect to its main attributes; *the current state of knowledge*; *his state of health*; *in a weak financial state*" when the correct sense is "the territory occupied by one of the constituent administrative districts of a nation; *his state is in the deep south.*" All the terms in the chain are then linked because they are all states that something can be in (i.e., state of death, state of order, etc.).

5.7 Discussion

5.7.1 Longest chains

When selecting the three longest chains for a document, more than three chains could be chosen if some long chains were tied in length. If many more than three longest chains were chosen then that would indicate that the longest chains in the document were not distinct. This did not prove to be the case: in general, our method was fairly good at selecting only three longest chains of each type per document. The average number of longest chains selected per document, and the large difference between the average chain lengths of the longest chains compared to all chains (shown in Figure 5.7.1), indicates the distinct nature of the longest chains in a document.

	Perfect	Classic
Average number of longest chains per document	3.37	3.27
Average length of longest chains	17.2	21.3
Average length of all chains	4.37	4.931

5.7.2 Improvement in chains with perfect disambiguation

The results show a significant improvement in the performance of the lexical chaining algorithm when the correct senses for all terms are used. A total of 92 longest classic chains were judged to be incorrect, compared to only 10 of the longest perfect chains being incorrect. Incorrect chains account for 27.5% of all longest classic chains, but only 2.9% of all longest perfect chains are incorrect. The untouched longest classic chains represent all the instances where the algorithm with no prior word sense disambiguation selected a completely different chain compared to when perfect disambiguation was used. 72.7% of these longest classic chains were incorrect, whereas only 9.8% of the untouched longest perfect chains were incorrect. Therefore, perfectly disambiguating candidate terms before performing lexical chaining does greatly improve the correctness of the resulting chains, which is what we hypothesized.

5.7.3 Errors in perfect chains

Even with perfect word sense disambiguation, at least 10 incorrect longest chains were created (these are the incorrect untouched longest perfect chains). The 94 matching and almost matching longest perfect chains may also include some incorrect chains, as we did not check those chains for errors. These errors occur because Silber and McCoy's semantic relatedness measure incorrectly considers some unrelated senses to be related. Usually the cause of such errors is the links in WordNet between unrelated senses located high in the noun hierarchy. The incorrect longest perfect chain referred to earlier, {*occurrences, experience, experience, instances, experience, instance, experience, incident, cases, things, cases, break, experiences, instances*}, is a result of this problem. WordNet lists *occurrence* as the parent of *thing* ("an event; *a funny thing happened on the way to the. . . .*") and *experience, instance, incident, case*, and *break* as children of *thing* [1], so all the terms are related to *thing* according to the Silber and McCoy relatedness measure.

5.7.4 Incorrect disambiguation leads to better chains

One unexpected result to note is the formation of longest classic chains that are untouched and correct, of which there were 27. These chains combine terms from multiple perfect chains that are shorter than the longest perfect chains selected for the document. For example, document br-d03 contains the longest classic chain {*churches, Church, churches, churches, Churches, churches, churches, churches, churches, churches, churches, church, church, church, churches, churches, church, Church, Protestant, Protestants, Protestants, Protestants, Protestants, faith, faith, Faith, faith, religious_orders, Catholic_Church, faith, Catholic_church*}. This chain combines terms from the following perfect chains:

1. {*churches, churches, church, Church, Protestant, faith, Faith, religious_orders*}
2. {*faith, faith, faith*}
3. {*services, Mass, church, Mass, Mass, Mass*}
4. {*Protestants, Protestants, Protestants, Protestants, Anglicans, Nonconformists*}
5. {*Church, churches, churches, Churches, churches, churches, churches, churches, churches, churches, church, church, churches, cathedrals, chapel, chapel, chapels*}
6. {*Catholic_Church, Catholic_church, Roman, Roman*}

The longest classic chains selects the same sense for all instances of *church* in the document, "one of the groups of Christians who have their own beliefs and forms of worship", but this sense is only correct for the instances of *church* in the first perfect chain. In the third perfect chain, *church* refers to "a service conducted in a church; *don't be late for church*" and in the fifth perfect chain *church* refers to "a place for public (especially Christian) worship; *the church was empty*". To us, it seems obvious that these different senses of *church* are related, and so should be placed together in a chain. Yet there is no short path between them in WordNet. They have no common ancestor or descendant, and the shortest path between any of the senses is 14 terms long (as shown in Figure 5.9). Such a path connects all manner of buildings to religion and faith via senses such as *flower_arrangement*, making it impossible for any WordNet-path-based semantic relatedness measure to recognize that the two senses of *church* are related without incorrectly relating other unrelated senses.

For the lexical chainer to place all the *church* terms in the document in the same chain, then the same sense must be used for all the terms, whether or not it is the correct sense for them. In these cases, choosing an incorrect sense for some terms actually leads to better chains compared to using the correct senses, which contradicts our hypothesis that improved word sense disambiguation performance necessarily leads to better lexical chaining performance.

Formally speaking, the situation is as follows:

A and B are words in a document whose correct senses when disambiguated are ${s_A}^*$ and ${s_B}^*$ respectively. ${s_A}^*$ and ${s_B}^*$ are semantically related such that A and B should be chained together, but the relation between ${s_A}^*$ and ${s_B}^*$ is not recognized in WordNet (i.e., there does not exist a short path between ${s_A}^*$ and ${s_B}^*$). However, A has another sense, ${s_A}'$, which is semantically related to both ${s_A}^*$ and ${s_B}^*$. Furthermore, there exists a short path from ${s_A}'$ to ${s_B}^*$ in WordNet. Therefore by selecting an incorrect sense for A, ${s_A}^*$, A and B can be chained together, but if the correct sense for A is selected they cannot be chained together.

In the above case, "a place for public (especially Christian) worship" is the correct sense for some of the instances of *church*. This sense is semantically related to the correct sense for *Mass*, "(Roman Catholic Church and Protestant Churches) the celebration of the Eucharist", but the shortest WordNet path between these senses is 14 senses long. If the incorrect sense for *church*, "a service conducted in a church", is used, then *church* and *Mass* can be chained together, as they are siblings. In this case, the correct sense of *church* (as a building) is semantically related to the incorrect sense (church service), but there is no short path between them.

5.7.5 Correct chains that cannot be created

Selecting incorrect senses allows for all the *church* terms to be placed in the same chain. However, other terms can only be placed in this chain if they are related to the single sense selected for *church*. Terms such as *mass* and *chapel* are omitted because they are only linked to senses of *church* not used in the classic chain, even though both terms are related to faith, religion and churches. Given the 44 terms *churches, Church, churches, churches, Churches, churches, churches, churches, churches, churches, churches, church, church, church, churches, churches, church, Church, Protestant, Protestants, Protestants, Protestants, Protestants, faith, faith, Faith, faith, religious_orders, Catholic_Church, faith, Catholic_church, services, Mass, Mass, Mass, Mass, Anglicans, Nonconformists, cathedrals, chapel, chapel, chapels, Roman*, and *Roman*, there is no combination of senses for the terms, incorrect or not, that are all linked to a single overriding sense. Therefore, no selection of senses allows for all these terms to be placed in the same lexical chain by our chaining algorithm. The largest chain that can possibly be formed from these terms is the longest classic chains with 31 terms. This is the same problem noted previously in the *temperature*, *heat*, and *energy* overlapping chains, and is described formally as follows:

Given a longest perfect chain $p = \{t_1, t_2, ..., t_k, tp_1, tp_2, ..., tp_n\}$ and longest classic chain $c = \{t_1, t_2, ..., t_k, tc_1, tc_2, ..., tc_m\}$, where $t_1, ..., t_k$ are the terms they have in common, there does not exist senses $s_1, s_2, ..., s_k, sp_1, sp_2, ..., sp_n, sc_1, sc_2, ..., sc_m$, and overriding sense s^* such that:

1. s_i is a valid sense for term t_i for all $1 \leq i \leq k$
2. sp_i is a valid sense for term tp_i for all $1 \leq i \leq n$
3. sc_i is a valid sense for term tc_i for all $1 \leq i \leq m$
4. s_i is the same sense as s^* or it shares a parent, child or sibling relation with s^* for all $1 \leq i \leq k$
5. sp_i is the same sense as s^* or it shares a parent, child or sibling relation with s^* for all $1 \leq i \leq n$
6. sc_i is the same sense as s^* or it shares a parent, child or sibling relation with s^* for all $1 \leq i \leq m$

For example, there is no way to select senses for the terms *faith*, *church*, and *cathedral* such that all of them can be placed in the same chain. If the sense "one of the groups of Christians who have their own beliefs and forms of worship" is selected for *church*, then it can be chained with *faith*. Selecting the sense "a place for public (especially Christian) worship" for *church* allows it to be chained with *cathedral*. However, there is no sense that can be chosen for *church* that allows it to be chained with both *faith* and *cathedral*, regardless of which senses are selected for *faith* and *cathedral*.

While we did not count all the times that the above situation occurred, we did note more than a few occurrences of it. Such instances demonstrate that even allowing for incorrect senses to be used when they are related to the correct sense for a word would not allow for the largest possible correct chains to be found. This problem of incorrect disambiguation leading to better lexical chains is due to how related senses of the same word are represented in WordNet. We examine this problem in detail later in this chapter.

5.7.6 Validity of our model

In the previous chapter we defined the Correct Sense Property: There exists no word w such that choosing the correct sense for w would lead to it being chained incorrectly and choosing an incorrect sense for w would lead to it being chained correctly. Our results show that this property does not hold, as in some cases the incorrect sense chosen for a word by the lexical chainer allowed it to be placed in a correct chain that it could not be placed in when perfect disambiguation was used. The *church* chain we described above is a perfect example of this.

In the previous chapter, claim 1 states that "improved word sense disambiguation necessarily leads to improved lexical chains". This claim has been shown to be false, because the Correct Sense Property does not hold. Our results show that correct word sense disambiguation does not always lead to better lexical chains. However, because greater accuracy in word sense disambiguation does lead to more correct lexical chains on average, our second claim has been proven true: Word sense disambiguation should be performed separately from building lexical chains, using the most accurate method available.

In general, our model has proven to be a better method for performing lexical chaining, despite the instances where incorrect disambiguation was useful.

5.7.7 Why incorrect disambiguation is sometimes useful

WordNet is described as being "fine-grained" because often words in WordNet have multiple senses that are related in meaning. Sometimes these senses are so similar in meaning that even humans have difficulty distinguishing between them [16]. However, even words with closely related senses can also have completely unrelated senses. Such a word is *bank*, which has the following senses in WordNet:

1. A financial institution that accepts deposits and channels the money into lending activities: "he cashed a check at the bank"; "that bank holds the mortgage on my home".

2. Sloping land (especially the slope beside a body of water): "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents".

3. A supply or stock held in reserve for future use (especially in emergencies).

4. A building in which commercial banking is transacted: “the bank is on the corner of Nassau and Witherspoon”.

5. An arrangement of similar objects in a row or in tiers: “he operated a bank of switches”.

6. A container (usually with a slot in the top) for keeping money at home: “the coin bank was empty”.

7. A long ridge or pile: “a huge bank of earth”.

8. The funds held by a gambling house or the dealer in some gambling games: “he tried to break the bank at Monte Carlo”.

9. A slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force.

10. A flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning): “the plane went into a steep bank”.

Senses 1, 3, 4, 6, and 8 are all related: they largely have to do with storing money. Senses 2, 7, 9, and 10 are all related to a sloped shape: some have to do with sloped earth and one has to do with tilting a plane such that its wings form a slope. Sense 5 is distinct from all the other senses.

Despite senses 1 and 4 being related, they are not closely linked in WordNet. Sense 1, *bank* as an institution (e.g., the Bank of Canada) is a descendant of the sense {*group, grouping*}: “any number of entities (members) considered as a unit”, whereas sense 4, the actual bank building, is a descendant of {*entity*}: “that which is perceived or known or inferred to have its own distinct existence (living or nonliving))”. Some of the synsets to which sense 1 is linked are merchant bank, credit union, savings and loan, and trust company, all of which share the concept of dealing with finance. On the other hand, sense 4 is related to other buildings that store things, such as library and museum, and the financial aspect is completely ignored. As a result, if a document uses sense 4 of bank, a lexical chaining algorithm which correctly identifies its sense will be unable to connect to other terms related to finance via links in WordNet.

Our lexical chaining algorithm will usually chain together *bank* and *credit union* in a document because those terms have senses that are directly linked in WordNet. This will occur even if the intended sense of *bank* is number 4 (a physical bank building). This happens because our algorithm always seeks to select senses for words that allow them to be chained together: in the absence of words that are linked to senses of *bank* other than sense number 4, the chainer will select the only sense for *bank* that will result in it being placed in a chain (number 4). The following sentence is an example of this: "I use the bank on King Street, but if you want the credit union, their closest branch is in Kitchener." Here, *bank* refers to a physical bank building (located on King Street), but the chainer will select sense number 1.

The problem here is that the relations in WordNet do not indicate which senses of the same word are related. If WordNet did indicate which senses of the same word are related, then possibly the problem of having to choose incorrect senses to create the correct lexical chains in a document would be alleviated. Consider the case we described earlier where three different but related senses of *church* lead to three different chains under perfect disambiguation:

1. {*churches, churches, church, Church, Protestant, faith, Faith, religious_orders*}
2. {*services, Mass, church, Mass, Mass, Mass*}
3. {*Church, churches, churches, Churches, churches, churches, churches, churches, churches, churches, church, church, churches, cathedrals, chapel, chapel, chapels*}

These are the senses of *church* used in the respective chains above:

1. "one of the groups of Christians who have their own beliefs and forms of worship"
2. "a service conducted in a church; *don't be late for church*"
3. "a place for public (especially Christian) worship; *the church was empty*"

If these three different senses were linked in WordNet, then all of the terms in the three perfect chains could be placed together in a single chain. Correct word sense disambiguation would no longer lead to worse lexical chains than when incorrect senses are used. The new chain would be even better than the classic *church* chain from our results. The classic chain only contains words related to the single sense chosen for all the instances of *church*, whereas the new chain would contain all the words related to any of the three senses of *church*. In the future work section of the next chapter we suggest possible methods for automatically identifying these senses of the same word that are semantically related.

1. church, church_building: "a place for public (especially Christian) worship; *the church was empty*"

2. place_of_worship, house_of_prayer, house_of_God, house_of_worship: "any building where congregations gather for prayer"

3. building, edifice: "a structure that has a roof and walls and stands more or less permanently in one place; *there was a three-story building on the corner*; *it was an imposing edifice*

4. structure, construction: "a thing constructed; a complex construction or entity; *the structure consisted of a series of arches*; *she wore her hair in an amazing construction of whirls and ribbons*"

5. artifact, artefact: "a man-made object taken as a whole"

6. decoration, ornament, ornamentation: "something used to beautify"

7. flower_arrangement: a decorative arrangement of flowers

8. arrangement: "an orderly grouping (of things or persons) considered as a unit; the result of arranging; *a flower arrangement*"

9. group, grouping: "any number of entities (members) considered as a unit"

10. social_group: "people sharing some social relation"

11. organization, organisation: "a group of people who work together"

12. institution, establishment: "an organization founded and united for a specific purpose"

13. religion, faith: "institution to express belief in a divine power; *he was raised in the Baptist religion*; *a member of his own faith contradicted him*"

14. church, Christian_church: "one of the groups of Christians who have their own beliefs and forms of worship"

Figure 5.9: Shortest path from *church_building* to *Christian_church* in WordNet [1]

Chapter 6

Conclusion

6.1 Future work

For future work, we see our research continuing in two directions. The first is obvious: our model for separating word sense disambiguation from lexical chaining would be useful in any application that uses lexical chaining. The second direction would follow up on developing a method for identifying and linking semantically related senses of the same word in WordNet.

6.2 Applications using lexical chaining

The lexical chaining algorithms we presented in our literature review, apart from Galley and McKeown, were all developed as part of a larger application. These applications involved malapropism detection and correction [11], document summarization [3] [29], and topic segmentation [31]. Lexical chaining has been applied to other problems as well (Stokes [31] lists over 30 papers on applications of lexical chains). We believe that all applications that use lexical chains would benefit from separating word sense disambiguation from the chaining process.

Hirst and St. Onge [11] [30], Barzilay and Elhadad [3], Silber and McCoy [27] [28] [29], and Stokes [31] all perform automatic evaluations of the performance of their algorithms on their respective tasks. It would be interesting to evaluate the changes, if any, in the performance of these algorithms, when word sense disambiguation is performed separately from lexical chaining. Changing these algorithms to use our model of lexical chaining would be straightforward, but choosing a test corpus for the evaluation is more problematic. The papers listed above all use specialized corpora for evaluating their algorithms, none of which have been disambiguated in the manner of the SemCor corpus we used for our experiment.

One option is to adapt the SemCor corpus to work with their evaluation methods. Adapting SemCor would then allow testing the effect of perfect disambiguation with the applications listed above, in a manner similar to our experiment. For Barzilay and Elhadad, Silber and McCoy, and Stokes, this adaptation would involve generating gold standard summaries of the SemCor documents by hand. Hirst and St. Onge require malapropisms to be manually added to the documents, and they explain how to do this in their work.

The SemCor corpus is useful because it allows for perfect disambiguation to be tested. However, our model can be evaluated without using perfect disambiguation. Instead, the corpora already used by the above papers for evaluation could be disambiguated with an available word sense disambiguation algorithm. The disambiguation would not be perfect, possibly far from perfect, but it would still be useful to evaluate whether performing word sense disambiguation separately with an imperfect disambiguation algorithm gives better results than when disambiguation is performed by the lexical chaining algorithm. Choosing a word sense disambiguation algorithm is easy in that a great many such algorithms exist, but difficult in that evaluation of these algorithms often varies. We direct readers to Mihalcea and Pedersen's [17] tutorial on word sense disambiguation for an excellent overview of the various approaches to the problem. As well, the Senseval Workshop [15] on the evaluation of word sense disambiguation algorithms is a good resource. Pedersen et al. [23] propose an approach that is particularly pertinent to lexical chaining in which senses are chosen so as to maximize semantic relation scores, in the same manner as current lexical chaining algorithms. Theirs is a generalized framework that allows for a number of parameters to be set to maximize disambiguation performance. The framework allows for a variety of WordNet-based semantic relatedness measurements to be used, most of which are more sophisticated than those currently used in lexical chaining algorithms.

6.3 Identifying related senses of the same word

We believe it is feasible to identify and link related senses of the same word in WordNet (but not the unrelated senses), as there has already been some work on this problem. This problem is usually defined as generating a "coarse-grained" WordNet, where very similar senses are combined. This work is usually done with the intention of improving the evaluation of word sense disambiguation methods which to date have largely involved rewarding only exact matches, i.e., only considering an algorithm to be correct if it selects the single correct sense for a polysemous word. All other answers are considered incorrect, regardless of how close in meaning they are to the correct answer. For example, if the word to be disambiguated is *bank* and the correct sense is the first sense (a financial institution), then all the other senses are considered equally wrong, whether they are close to the right answer (e.g., a bank building) or completely unrelated (sloping land). Resnik and Yarowsky [24] outline how semantic relatedness between senses of the same word can be used in disambiguation evaluations, and methods for combining similar senses have been presented by Mihalcea and Moldovan [16] and Palmer et al. [22].

One approach which we believe has merit is using dictionaries to identify similar senses of the same word. Dictionaries often arrange definitions of a word into hierarchies, first dividing a word into its unrelated meanings, and then listing all the related senses for the word under their shared meanings. For example, the Oxford English Dictionary [2] entry for *bank* lists three main noun senses: one pertaining to the finance-related senses of *bank*, one pertaining to slopes or hills, and one to a bench, shelf, or row of items. A WordNet sense's gloss could be compared to the definitions in a dictionary to decide which main dictionary sense the WordNet sense would fall under. WordNet senses belonging to the same dictionary main sense could then be linked.

While the above research ideas are more speculative than our earlier ideas for adapting existing lexical chaining algorithms, we consider them more interesting. Modeling word meanings computationally is a fascinating but difficult problem, and we believe the problems we encountered with the lexical chains produced during our experiment indicate key problems with current approaches.

6.4 Thesis contributions

This thesis has made six main contributions to the areas of lexical chaining:

New lexical chaining model: In Chapter 4 we presented a new model for lexical chaining. Our model separates word sense disambiguation from lexical chain creation. We separate word sense disambiguation because polysemous words must be disambiguated before the semantic relations between them can be determined; the semantic relations, if any, between two words cannot be determined before the meanings of the two words are known. Once word sense disambiguation is separate from lexical chain creation, any method for word sense disambiguation can be used, with more accurate methods being desirable.

Improved lexical chaining algorithm: In Chapter 5 we proposed a modified version of Silber and McCoy's lexical chaining algorithm for use in our experiment. We showed that by modifying how the contribution of a word to a metachain was calculated, the word sense disambiguation accuracy of the algorithm was significantly improved. While this change does increase the runtime of the algorithm from $O(n)$ to $O(n^2)$, the new runtime is still reasonable.

Experimental evaluation of our model: Chapter 5 contains our experiment which evaluates the effectiveness of performing accurate word sense disambiguation separately from building the lexical chains. The experiment compared the correctness of the lexical chains created under two conditions:

1. No prior disambiguation of candidate terms. This is how lexical chaining algorithms are normally run.
2. Perfect disambiguation performed prior to the creation of lexical chains.

Our results showed that the longest lexical chains created under the second condition were significantly more likely to be correct than those created under the first. These results prove that accurate word sense disambiguation is important for building correct lexical chains, and therefore disambiguation should be performed separately from lexical chaining. Our results also demonstrated that correct word sense disambiguation does not always lead to better chains than incorrect disambiguation.

How incorrect disambiguation can improve lexical chaining: In our results, we found that sometimes incorrect word sense disambiguation resulted in more correct lexical chains than those produced under correct disambiguation. At the end of Chapter 5 we showed that WordNet contains senses of the same word that are related in meaning, yet no short path between these senses exists. As a result, selecting an incorrect but related sense for a candidate word can allow it to be placed in a correct chain, whereas selecting the correct sense for that candidate would leave it unchained.

Applying our model to other lexical chaining algorithms: In the future work section of this chapter we explained how other lexical chaining algorithms could be adapted to fit our model. Adapting these algorithms would allow for our model to be evaluated with regard to specific applications of lexical chaining.

Identifying related senses of the same word: In the future work section we also proposed possible methods for identifying senses of the same word that are related, such as the following senses of *church*: "a service conducted in a church; *don't be late for church*" and "a place for public (especially Christian) worship; *the church was empty*" [1]. Linking related senses such as these could eliminate the cases where incorrect word sense disambiguation allows for more correct chains than correct disambiguation.

Our original goal was to demonstrate the importance of word sense disambiguation to lexical chaining, and prove the benefits of performing disambiguation before building lexical chains. We accomplished both these goals, but also found an unexpected result: it is sometimes the case that better lexical chains can be created by selecting an incorrect sense for a word.

Appendix A

Brown corpus portion of the SemCor corpus

Document	Content
br-a01	Newspaper/periodical article
br-a02	Newspaper/periodical article
br-a11	Newspaper/periodical article
br-a12	Newspaper/periodical article
br-a13	Newspaper/periodical article
br-a14	Newspaper/periodical article
br-a15	Newspaper/periodical article
br-b13	Newspaper/periodical editorial
br-b20	Newspaper/periodical editorial
br-c01	Newspaper/periodical review
br-c02	Newspaper/periodical review
br-c04	Newspaper/periodical review
br-d01	Essay on religion
br-d02	Essay on religion
br-d03	Essay on religion
br-d04	Essay on religion
br-e01	Article on hobbies/recreation
br-e02	Article on hobbies/recreation
br-e04	Article on hobbies/recreation
br-e21	Article on hobbies/recreation
br-e24	Article on hobbies/recreation
br-e29	Article on hobbies/recreation

Document	Content
br-f03	Article on history/folklore
br-f10	Article on history/folklore
br-f19	Article on history/folklore
br-f43	Article on history/folklore
br-g01	Social commentary article
br-g11	Social commentary article
br-g15	Social commentary article
br-h01	Government report
br-j01	Academic journal article
br-j02	Academic journal article
br-j03	Academic journal article
br-j04	Academic journal article
br-j05	Academic journal article
br-j06	Academic journal article
br-j07	Academic journal article
br-j08	Academic journal article
br-j09	Academic journal article
br-j10	Academic journal article
br-j11	Academic journal article
br-j12	Academic journal article
br-j13	Academic journal article
br-j14	Academic journal article
br-j15	Academic journal article
br-j16	Academic journal article
br-j17	Academic journal article
br-j18	Academic journal article
br-j19	Academic journal article
br-j20	Academic journal article
br-j22	Academic journal article
br-j23	Academic journal article
br-j37	Academic journal article
br-j52	Academic journal article
br-j53	Academic journal article
br-j54	Academic journal article
br-j55	Academic journal article
br-j56	Academic journal article
br-j57	Academic journal article
br-j58	Academic journal article

Document	Content
br-j59	Academic journal article
br-j60	Academic journal article
br-j70	Academic journal article
br-k01	Fiction
br-k02	Fiction
br-k03	Fiction
br-k04	Fiction
br-k05	Fiction
br-k06	Fiction
br-k07	Fiction
br-k08	Fiction
br-k09	Fiction
br-k10	Fiction
br-k11	Fiction
br-k12	Fiction
br-k13	Fiction
br-k14	Fiction
br-k15	Fiction
br-k16	Fiction
br-k17	Fiction
br-k18	Fiction
br-k19	Fiction
br-k20	Fiction
br-k21	Fiction
br-k22	Fiction
br-k23	Fiction
br-k24	Fiction
br-k25	Fiction
br-k26	Fiction
br-k27	Fiction
br-k28	Fiction
br-k29	Fiction
br-l11	Fiction
br-l12	Fiction
br-m01	Fiction
br-m02	Fiction
br-n05	Fiction
br-p01	Fiction

Document	Content
br-r05	Humour
br-r06	Humour
br-r07	Humour
br-r08	Humour
br-r09	Humour

Bibliography

[1] WordNet 2.0. *http://wordnet.princeton.edu/2.0/WordNet-2.0.exe*.

[2] Oxford English Dictionary. *dictionary.oed.com*, 2006.

[3] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization (ISTS)*, Madrid, Spain, 1997.

[4] Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32:13–47, June 2006.

[5] Christiane Fellbaum. Introduction. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database*, pages 1–20. MIT Press, Cambridge, Massachusetts, 1998.

[6] W. N. Francis and H. Kucera. Brown corpus manual. *http://helmer.aksis.uib.no/icame/brown/bcm.html*, 1964.

[7] Michel Galley and Kathleen McKeown. Improving word sense disambiguation in lexical chaining. In *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[8] Barbara Grosz and Candace Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204, July 1986.

[9] M. A. K. Halliday and Ruqaiya Hasan. *Cohesion in English*. Longman Group Ltd., 1976.

[10] Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111, March 2005.

[11] Graeme Hirst and David St. Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database*, chapter 13. MIT Press, Cambridge, Massachusetts, 1998.

[12] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*, Taiwan, 1997.

[13] Adam Kilgarriff. Gold standard datasets for evaluating word sense disambiguation programs. *Computer Speech and Language*, 12:453–472, 1998.

[14] Shari Landes, Claudia Leacock, and Randee Tengi. Building semantic concordances. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database*, chapter 8. MIT Press, Cambridge, Massachusetts, 1998.

[15] Rada Mihalcea and Phil Edmonds, editors. *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July 2004. Association for Computational Linguistics Conference.

[16] Rada Mihalcea and Dan I. Moldovan. Automatic generation of a coarse grained wordnet. In *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA, June 2001.

[17] Rada Mihalcea and Ted Pedersen, editors. *Advances in word sense disambiguation*. Association for Computational Linguistics Conference, ACL 2005, June 2005.

[18] George A. Miller. Nouns in WordNet. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database*, chapter 1. MIT Press, Cambridge, Massachusetts, 1998.

[19] Saif Mohammad and Graeme Hirst. Distributional measures as proxies for semantic relatedness. Submitted, 2005.

[20] Jane Morris. Lexical cohesion, the thesaurus, and the structure of text. Master's thesis, University of Toronto, December 1988.

[21] Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–43, 1991.

[22] Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, To appear.

[23] Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation. Technical Report 2005/25, University of Minnesota Supercomputing Institute, March 2005.

[24] Philip Resnik and David Yarowsky. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5:113–133, June 1999.

[25] R. Richardson and A. F. Smeaton. Using WordNet in a knowledge-based approach to information retrieval. Working paper, Dublin City University, Dublin, Ireland, 1995.

[26] Peter Roget. *Roget's International Thesaurus*. Harper and Row, Publishers Inc., fourth edition, 1977.

[27] Gregory Silber and Kathleen McCoy. Efficient text summarization using lexical chains. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI 2000)*, pages 252–255, New Orleans, January 2000.

[28] Gregory Silber and Kathleen McCoy. An efficient text summarizer using lexical chains. In *Proceedings of the First International Conference on Natural Language Generation (INLG 2000)*, pages 268–271, Israel, June 2000.

[29] Gregory Silber and Kathleen McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496, December 2002.

[30] David St-Onge. Detecting and correcting malapropisms with lexical chaining. Master's thesis, University of Toronto, Toronto, Canada, 1995.

[31] Nicola Stokes. *Applications of lexical cohesion analysis in the topic detection and tracking domain*. PhD thesis, National University of Ireland, Dublin, Ireland, April 2004.