

Ben-Gurion University of the Negev,
Department of Mathematics & Computer Science

Lexical Chains for Summarization

Thesis submitted as part of the requirements for the
M.Sc. degree of Ben-Gurion University of the Negev

by

Regina Barzilay:

(regina@cs.bgu.ac.il)

The research work for this thesis has been carried out at
Ben-Gurion University of the Negev,
under the direction of Dr. Michael Elhadad

November 30 1997

1 Kislev hatashnaz

Subject: **Lexical Chains for Summarization**

This thesis is submitted as part of the requirements for the M.Sc. degree

Written by: **Regina Barzilay**

Advisor: **Dr. Michael Elhadad**

Department: **Math & Computer Science**

Faculty: **Exact Sciences**

Ben-Gurion University of the Negev

Author signature: _____ Date: _____

Advisor signature: _____ Date: _____

Dept. Committee Chairman sig.: _____ Date: _____

At this point I would like to thank¹ the following:

- Michael Elhadad, for advising and, for professional and personal support, and for getting me interested in the subject of computational linguistics.
- Eli — this work would never happen without him.
- Samir Genaim.
- Yael Dahan-Netzer.
- Cohavit Taboch, Mohamad Abo-Zaed and Carmel Domshlak — for being a part of the “projects room”.
- Skyblue.
- Matti Rubin, Avraham Melkman and Michael Codish.
- Parents and Grandparents.

Regina Barzilay

¹This is an alpha version.

Abstract

This thesis investigates one technique to produce a summary of an original text without requiring its full semantic interpretation, but instead relying on a model of the topic progression in the text derived from lexical chains. We present a new algorithm to compute lexical chains in a text, merging several robust knowledge sources: the WordNet thesaurus, a part-of-speech tagger and shallow parser for the identification of nominal groups, and a segmentation algorithm derived from [8].

Summarization proceeds in three steps: the original text is first segmented, lexical chains are then constructed, *strong* chains are identified and *significant* sentences are extracted from the text.

An extensive empirical evaluation of the work is presented. First an intrinsic evaluation determines the quality of the lexical chainer by identifying how successful the lexical chainer is in disambiguating nouns in context. High precision disambiguation is achieved for the nouns belonging to the strongest lexical chains.

Second, an extrinsic evaluation is performed that determines to what extent the sentences extracted by the summarizer match those that human judges would extract. We compare our summarizer with two other available summarization systems recently made available (Summer 97) and find our approach to give results significantly closer to human judges (using majority rule) than the other systems.

The work presented is a robust sentence extractor, relying on a widely available lexical knowledge base (WordNet). The lexical chooser improves on existing systems. It can also, in the future, be integrated with other techniques relying on different knowledge sources (referential chains, conceptual maps) to strengthen the overall sentence extraction process. These current research directions are outlined at the end of the thesis.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Lexical Chains | 7 |
| 2.1 | Cohesion | 7 |
| 2.1.1 | Texture and Cohesion | 7 |
| 2.1.2 | Coherence and Cohesion | 8 |
| 2.1.3 | Types of Cohesion | 9 |
| 2.2 | Lexical Cohesion | 10 |
| 2.2.1 | Types of Lexical Cohesion | 10 |
| 2.2.2 | The Dominance of Lexical Cohesion | 11 |
| 2.2.3 | What is a Lexical Chain? | 11 |
| 3 | Computing Lexical Chains | 15 |
| 3.1 | Generic Algorithm | 15 |
| 3.2 | Semantic Distance Among Words | 16 |
| 3.2.1 | Knowledge Source | 16 |
| 3.2.2 | Differences Among Knowledge Sources | 18 |
| 3.2.3 | Knowledge Sources Appropriateness | 19 |
| 3.2.4 | Using Thesaurus for Computing | 20 |
| 3.3 | Candidate Words | 22 |
| 3.4 | Choice of a Receiving Chain | 24 |
| 3.5 | The Dynamic Chaining Algorithm | 26 |
| 3.6 | Dynamic Chaining Algorithm Description | 28 |
| 3.6.1 | General Explanation | 28 |
| 3.6.2 | Data Structures | 29 |
| 3.6.3 | Functions | 30 |
| 3.6.4 | Running example | 33 |

| | | |
|----------|---|-----------|
| 3.6.5 | Scoring Chains | 34 |
| 3.7 | Summary: Building Lexical Chains | 36 |
| 4 | Evaluating the Lexical Chainer | 37 |
| 4.1 | Evaluating Methods | 37 |
| 4.1.1 | Evaluation Framework | 37 |
| 4.1.2 | Topic Identification Measure | 39 |
| 4.1.3 | Word Sense Disambiguation Ratio | 39 |
| 4.2 | Influence of Performance Factors | 41 |
| 4.2.1 | Influence of Algorithm Parameters | 41 |
| 4.2.2 | Influence of Environment Properties | 49 |
| 4.3 | Lessons Learned from the Evaluation | 53 |
| 5 | Lexical Chains for Summarization | 55 |
| 5.1 | Motivation | 55 |
| 5.2 | Building Summaries Using Lexical Chains | 58 |
| 5.2.1 | Extracting Significant Sentences | 59 |
| 5.3 | Sentence Extraction Using Lexical Chains | 61 |
| 6 | Summary Evaluation | 63 |
| 6.1 | Description of the Experiment | 63 |
| 6.2 | Results and Analysis | 64 |
| 6.2.1 | Agreement Among Human Subjects | 64 |
| 6.2.2 | Statistical Significance | 66 |
| 6.2.3 | Systems Comparison | 66 |
| 7 | Conceptual Maps | 67 |
| 7.1 | Chain Interaction and “Cohesive Harmony” | 67 |
| 7.2 | Building Conceptual Maps | 70 |
| 7.2.1 | Definition of a Conceptual Map | 70 |
| 7.2.2 | Algorithm for Conceptual Map Construction | 72 |
| 8 | Contributions and Future Work | 75 |
| 8.1 | Contributions | 75 |
| 8.1.1 | The Chaining Algorithm | 75 |
| 8.1.2 | Summarization | 75 |
| 8.2 | Limitations & Future Work | 76 |

| | |
|--|-----------|
| <i>CONTENTS</i> | iii |
| 8.2.1 The Summarization System | 76 |
| 8.2.2 More Future Work | 77 |
| A Bayesian Networks Text | 79 |
| A.1 The Raw Text | 79 |
| A.2 Bayesian Network Text: the Strongest Chain | 81 |
| Bibliography | 86 |

Chapter 1

Introduction

The last decade has witnessed a revolution in the way the public can access information: wide Internet availability has made the notion of full-text search and information retrieval familiar to million of casual users. At the same time, the amount of textual material on-line (and, therefore, susceptible to be of interest to an information-seeker user) has grown at an enormous rate.

This has led to the creation of a problem known as *information overload*: a simple query to an information retrieval service such as Digital's Alta Vista can routinely return millions of documents as "potentially relevant." Most of these documents are mainly textual. Summarization is one of the key technologies that will address the problem of information overload.

Summarization is the process of condensing a source text into a shorter version while preserving its information content. It can serve several goals — from survey analysis of a scientific field to quick indicative notes on the general topic of a text. Producing a quality summary of an arbitrary text remains a challenge which requires full understanding of the text. Indicative summaries, which can be used to quickly decide whether a text is worth reading in the context of an information retrieval task are naturally easier to produce. We investigate in this work a method for the production of such indicative summaries from arbitrary text.

Sparck Jones [15] describes summarization as a two-step process:

1. building from the source text a source representation.

2. summary generation — forming a summary representation from the source representation built in step 1 and synthesizing the output summary text.

Within this framework the relevant question is what information has to be included in the source representation in order to create a summary. There are three types of source text information: linguistic, domain and communicative. Each of these text aspects can be chosen as a basis for source representation.

Summaries can be built on a deep semantic analysis of the source text. For example, in [21], McKeown and Radev investigate how to produce a coherent summary of several texts describing the same event, when a full semantic representation of the source texts is available (in their case, they use MUC-style systems to interpret the source texts). In contrast, we deal in this work with the issue of producing a summary from an arbitrary text without requiring its full understanding, and using only widely available knowledge sources.

Alternatively, early summarization systems [18] used only linguistic source information. The intuition was that the most frequent words represent the important concepts of the text. In this approach the source representation was the frequency table of text words. This representation abstracts the text into the union of its words without considering any connection among them.

It is clear that this simplification can harm the quality of the source representation. As a trivial illustration, consider the following two sequences:

1. “*Yael reads a book. She is at the university.*”
2. “*Yael reads a book. It is about history.*”

“*Yael*” appears once in 1 and 2, and so does “*book*”. But 1 is about *Yael*, and 2 is about the *book*. This example indicates that if the source representation does not supply information about semantically related terms, one cannot capture the “aboutness” of the text, and therefore the summary will not capture the main point of the original text.

The notion of cohesion introduced in Halliday and Hasan [6] captures part of the intuition. Cohesion is a device for “sticking together”

different parts of the text. Cohesion is achieved through the use in the text of semantically related terms, reference, ellipsis and conjunctions.

Among these different means, the most easily identifiable and the most frequent type is lexical cohesion (as discussed in [10]). Lexical cohesion is created by using semantically related words. Halliday and Hasan classified lexical cohesion into reiteration category and collocation category. Reiteration can be achieved by repetition, synonyms and hyponyms. For example:

1. Repetition:

“*The **lecture** begins at 18:00.*”

“*The topic of the **lecture** was published yesterday.*”

2. Synonyms:

“*The **lecture** begins at 18:00.*”

“*The topic of the **talk** was published yesterday.*”

3. Hyponyms:

“*Dr Kenny has invented an anaesthetic **machine**.*”

“*This **device** controls the rate at which an anaesthetic is pumped into the blood.*”

Collocation relations specify the relation between words that tend to co-occur in the same lexical contexts. For example:

“*She works as a **teacher** in the **School**.*”

Collocation relations are more problematic for identification than reiteration, but both of these categories are identifiable on the surface of the text. Lexical cohesion occurs not only between two terms, but among sequences of related words — called *lexical chains* [6]. Lexical chains provide a representation of the lexical cohesive structure of the text. Lexical chains have also been used for information retrieval [31] and for correction of malapropisms [9]. In this work, we investigate how lexical chains can be used as a source representation for summarization.

Another important dimension of the linguistic structure of a source text is captured under the related notion of *coherence*. Coherence defines the macro-level semantic structure of a connected discourse, while cohesion creates connectedness in a non-structural manner. Coherence is represented in terms of coherence relations between text segments,

such as *elaboration*, *cause* and *explanation*. A number of researchers [20] [22] use discourse structure (encoded using RST [19] as a source representation for summarization. Clearly, this representation is expressive enough, the question is if it is computable. In contrast with lexical cohesion, coherence is difficult to identify without a complete understanding of the text and complex inference.

There is, however, a close connection between discourse structure and cohesion. Related words tend to co-occur within discourse unit of the text. So cohesion is one of the surface signs of discourse structure and lexical chains can be used for identifying it. Other signs can be used to identify discourse structure as well (connectives, paragraph markers, tense shifts). In recent work, Marcu [20] proposes a summarization system based on the determination of coherence relations. We will discuss this approach in the following chapters as well.

In this work, we investigate the use of lexical chains as a model of the source text for the purpose of producing a summary. Clearly, other aspects of the source text need to be integrated in the text representation to produce quality summarization. But we want to empirically investigate how far one can go exploiting mainly lexical chains. A main focus of this work has been to determine techniques to evaluate the quality of the results produced by the lexical chainer and the sentence extraction system we have developed. We develop two distinct evaluation techniques: one intrinsic and one extrinsic.

The intrinsic evaluation method determines a necessary condition for an ideal lexical chainer: that it correctly disambiguate the nouns it identifies as central topics in the text. It then measures to what extent our implementation of a lexical chaining algorithm satisfies this constraint. We compare using this measure different lexical chaining algorithms and show that our algorithm significantly improves on existing systems.

The extrinsic evaluation method compares how our sentence extraction system compares with the judgment of human subjects asked to extract sentences manually from a set of texts. Again, we compare our sentence extraction system with 2 available sentence extraction systems, and find our system to perform significantly closer to the human judgment (using majority rule).

The sentence extraction system is fully implemented and many ex-

ample summaries are available on the Web¹. It is a robust system, working on arbitrary texts that can highlight the most important sentences in context, or extract them to produce a condensed version of the source text. Appendix A.2 shows an example of output produced by the system for the text in A.1, a summary of this text can be found in page 59.

The thesis is organized as follows: we first present background material on lexical chains and the linguistic theories underlying them (cohesion, texture and coherence). We then present our algorithm for the construction of lexical chains, comparing it with existing lexical chaining algorithms presented in the literature. Chapter 4 presents the intrinsic evaluation of the lexical chainer. Chapter 5 explains how the computation of the lexical chains can be exploited as a basis for a sentence extraction technique. Chapter 6 presents evaluation of our summarization algorithm. The final chapters present an alternative method that builds what we call “conceptual maps” of a source text using the links among lexical chains that can be identified in the text. We conclude with a discussion of the current limitations of the implementations and plans to address them in the future.

¹At <http://www.cs.bgu.ac.il/summarization-test>

Chapter 2

Lexical Chains

We introduce in this chapter a set of concepts and definitions that serve as the basis for the lexical chain techniques described in this work. These notions evolve around the definition of what constitutes *text* and follow the analytical framework introduced by Halliday & Hasan.

2.1 Cohesion

2.1.1 Texture and Cohesion

Every naive reader can distinguish a text from a random set of sentences. Halliday [7] used the notion *texture* for the text property, that of “being a text.” A text contains certain linguistic features that contribute to its “texture” which encourage a reader to interpret sentences as belonging together in some way.

Cohesion can be defined as the way certain words or grammatical features of a sentence can connect it to its predecessors (and successors) in a text. Halliday & Hasan [6] defined cohesion as “the set of possibilities that exist in the language for making text hang together.”

Cohesion occurs where the interpretation of some element in the discourse is dependent on that of another. For example, an understanding of the reference “he” requires to look back to something that has been said before. Through this cohesion relation, two text clauses are linked together.

2.1.2 Coherence and Cohesion

Another important dimension of the linguistic structure of a source text is captured under the related notion of *coherence*. Coherence defines the macro-level semantic structure of a connected discourse, while cohesion creates connectedness in a non-structural manner. Coherence is represented in terms of relations between text segments, such as *elaboration*, *cause* and *explanation*.

Hasan [28] claims that coherence is a “relative, not an absolute property” of a text, and human judgments can distinguish more coherent text from less coherent one. Hoey [10] refers to this phenomena as “subjectiveness” of coherence — as produced by a reader’s evaluation of some text. In contrast with this “relativeness” of coherence, cohesion is an objective text property, which is created by lexical and grammatical devices.

In contrast to cohesion, coherence is difficult to identify without a complete understanding of the text and complex inferences. In addition, there is no precise criteria for classification of different relations. Consider the following example from Hobbs [13]: “*John can open the safe. He knows the combination.*”

Morris and Hirst [24] show that the relation between these two sentences can be interpreted as an *elaboration* or as an *explanation*, depending on “context, knowledge and beliefs.” In contrast with coherence, there is widespread agreement on the classification of cohesion relations — the classification of Halliday & Hasan [6] provides a very detailed account of cohesion devices.

There is, however, a close connection between discourse structure and cohesion. A non-coherent sequence of sentences can exhibit cohesion and similarly a set of sentences can be coherent without evidence of cohesion (Morris and Hirst, [24]). But generally cohesion is evident when sentences relate coherently and this trait can be exploited by using cohesion relations as means of identifying coherent parts of the text.

Hasan states that the foundations of coherence are laid through cohesion relations, but that the source of coherence is the interaction of the cohesive relations (we continue this discussion in the “conceptual map” chapter below).

2.1.3 Types of Cohesion

Halliday & Hasan [6] describe in detail the devices that create the semantic bounds between parts of text clauses. The relationships amongst items in a text are divided into four broad categories: *reference*, *ellipsis*, *conjunction* and *lexical cohesion*. We illustrate all these relations from the following text¹:

“Error analysis” results in a percentage of words which have been deemed incorrect in an entire text. The method involves a post-editor counting every word addition or deletion, every word substitution, and every word transposition that he or she makes to the “raw” output from the system. This measure is meant to be an objective measure of successful, accurate or faithful translation. However, such error counting is not in fact a valid method for objective fidelity, unless other factors of the larger setup are kept constant.

The use of form “*counting every word addition or deletion*” is an example of ellipsis — it must be interpreted as “... *or counting every word deletion*”. The usage of “*this measure*” to the “*error analysis*” exemplifies reference. The word “*however*” expresses a conjunctive relation between “*this measure is meant to be an objective measure...*” and “*error counting is not in fact a valid method...*”. The usage of the words “*counting*”, “*measure*”, “*measure*”, and “*counting*” is an instance of lexical cohesion.

We begin by describing briefly the first three categories, and then focus on lexical cohesion.

Reference A reference is linked by a semantic relation to some element in the preceding text; and this connection enables the referenced item to be interpreted. For example, in the sentences “*I am waiting for Cohavit. She will arrive later*”, the word “*she*” is interpreted as identical to the word “*Cohavit*”.

Ellipsis Ellipsis occurs when a clause or some of its parts “may be presupposed at a subsequent place in the text by the device of positive omission — that is, by saying nothing, where something

¹Taken from Karen Sparck Jones [16]

is required to make up some sense.” For example, in the sentence “*Samir is studying in this semester Complexity and Pattern Recognition*”, the verb “*study*” relates to the word “*Complexity*” explicitly and to the word “*Pattern Recognition*” implicitly.

Conjunction Conjunctions expresses logic-semantic relations between clauses explicitly. The following sentences exemplify a causal relation, expressed by the word “*because*”: “*Yael finished her thesis in time because she worked very hard*”.

2.2 Lexical Cohesion

Cohesion may be created in a text by a choice of words. Lexical cohesion occurs through the selection of words that are related in some way to words that have been used before. Lexical cohesion can be illustrated on the following text:

*But Mr. Kenny’s move speeded up work on a **machine** which uses **micro-computers** to control the rate at which an **anaesthetic** is pumped into the blood of **patients** undergoing surgery. Such **machines** are nothing new. But Mr. Kenny’s **device** uses two **personal-computers** to achieve much closer monitoring of the **pump** feeding the **anaesthetic** into the **patient**. Extensive testing of the **equipment** has sufficiently impressed the authorities which regulate medical **equipment** in Britain and, so far, four other countries, to make this the first such **machine** to be licensed for commercial sale to hospitals.*

This text contains many words that are semantically related one to another, for example: machine/device/equipment, patients/patient. These relations link sentences together and by this create a cohesive text unit from them. Word relations can be one of the many forms which are the basis for classification of lexical cohesion.

2.2.1 Types of Lexical Cohesion

Halliday and Hasan classified lexical cohesion into two main categories — *reiteration* and *collocation*.

Reiteration occurs when one lexical item brings to mind the meaning of an earlier item in the discourse. Reiteration is created by using repetition, synonyms and hyponyms. For example, the word “machine” is repeated three times. The synonymy relation can be exemplified by the pair “*personal-computers*” and “*micro-computer*”. Similarly, the word “*equipment*” is more general than the word “*machine*” (hyponymy relation).

Collocation refer to words that tend to co-occur in the text. These can be divided into two sub-categories: systematic, and non-systematic. The systematic semantic relation is created by meronymy and antonymy relations. Non-systematic relations are hard to define. Such collocations describe things that tend to co-occur in similar situations or contexts in the real world. In our text, an example of such collocations are the words “*patient*” and “*hospital*”, or “*hospital*” and “*anaesthetic*”.

2.2.2 The Dominance of Lexical Cohesion

Halliday & Hasan checked the frequency of different cohesion types with a variety of text styles. According to their results, lexical cohesion is the most dominant category — it makes more than 40% of all cohesive devices. These conclusions were supported by Hoey [10], who made similar experiments on seven different types of text. Hoey claims that lexical cohesion is the crucial factor in creating texture and that “the study of the greater part of cohesion is the study of lexis.”

Our claim that cohesion analysis using only lexical cohesion can yield partial but significant results is based on these data of Halliday & Hasan and Hoey.

2.2.3 What is a Lexical Chain?

Lexical cohesion occurs not only between two terms, but among sequences of related words as well — these are called lexical chains. Lexical chains can hold over sentences and different text parts. This term was introduced by Halliday & Hasan in 1976 [6], and was expanded in the later work of Hasan [28].

“Identity Chains” vs. “Similarity Chains”

Hasan recognizes two types of chains in [28] — “identity chains” (IC) and “similarity chains” (SC). Identity chains contain terms that refer to the same object. It is created by pronominal cohesion, lexical repetition or instancial equivalents. For example, “*The name of the cat was **Gershon***”. Hasan noted that identity chains are “always text-bound” because the relation of co-reference can be determined only in the context of a text.

The elements of similarity chains are held together by “semantic bonds which are supra-textual, with a language-wide validity.” It is realized only through lexical cohesion categories. In contrast with IC, SC are not text-bound — its formation and content “are facts of the system of language”.

The notion of a *lexical chain*, introduced for the first time in a computational framework by Morris & Hirst [24], is equivalent to the notion of *similarity chains*. According to Morris & Hirst, lexical chains are defined as “sequences of related words” extending over a topical unit of the text. Items can be included in a lexical chain based on different types of lexical cohesion — reiteration or collocation categories, the precise nature is not significant.

Why Can Lexical Chains Be Computed

Two types of chains are important for cohesion analysis, however, the advantage of similarity chains (lexical chains) over identity chains is that they can be computed without requiring ‘deep’ text understanding.

Consider the cohesive device of using synonyms, for example: “*PC*” and “*microcomputer*”. The semantic relation between these terms is created by the “identity of their experimental meaning.” Therefore, the semantic bounds between synonyms are independent of a particular context of some text; the relations between them are part of a language system. This makes it possible to calculate lexical chains irrespective to the context in which related words actually occur.

This property suggests that one can develop a computational process capable of identifying lexical chains. We discuss this point in the next chapter and continue by discussing how the lexical chain informa-

tion can be exploited to build summaries.

Chapter 3

Computing Lexical Chains

3.1 Generic Algorithm

Lexical cohesion is a surface property of the text, and it is a frequent one. This property makes it possible to compute lexical chains. We define lexical chains as groups of semantically related words.

A general algorithm for computing a chain (in terms of collecting word clusters) can be presented in the following way:

1. Select a set of *candidate words* from the text;
2. For each of the candidate words, find an appropriate chain to receive a the candidate word, relying on a *relatedness criterion* among members of the chains and the candidate words;
3. If such a receiving chain is found, insert the candidate word in this chain and update it accordingly; otherwise, create a new chain.

In the following sections we discuss possible values of the algorithm parameters — the definitions of *candidate words*, *relatedness criterion* and *choice of appropriate chain for the new candidate word*.

3.2 Semantic Distance Among Words

3.2.1 Knowledge Source

In order to determine if two words are semantically related we need additional knowledge about connections between words. In this section we consider the current machine-readable sources of knowledge which may be suitable for lexical cohesion analysis.

Roget International Thesaurus

The Roget's Thesaurus [29] is composed of 1042 basic categories, organized in a hierarchical structure. The topmost hierarchical level consists of eight classes: abstract relations, space, matter, sensations, physics, intellect, volition, and affections. Each class is divided into subclasses, and subclasses are further divided into subclasses. Subclasses contain categories. For example:

Class 1: Abstract Relations

Section I. EXISTENCE

A BEING, IN THE ABSTRACT

1 Existence

1 N. existence, being, entity, ens[Lat], esse[Lat], subsistence.

2 reality, actuality; positiveness. adj.; fact, matter of fact, sober reality; truth § 494; actual existence.

...

2 Inexistence

1 ...

B CONTINUOUS RELATION

1 ...

Section II. RELATION

...

Class 2: Space

• ...

Each category contains numbered paragraphs of related words. A category may have pointers to other related categories or paragraphs. A category contains different part of speech words, but a paragraph contains words of only one syntactic category. For each entry, a list of words represents its various distinct subsenses. For example, the “*city*” entry is represented in the Table 3.1.

| | |
|------------|-----|
| Abode | 189 |
| Prosperity | 734 |
| Barter | 794 |
| Amusement | 840 |

Table 3.1: The Roget’s thesaurus entry for “*city*”.

WordNet

WordNet [23] is a lexical knowledge base developed at Princeton University. It is divided into four data files containing data for adjectives, adverbs, nouns and verbs. In WordNet, a word form is represented by a string of ASCII characters, and a sense is represented by a *synset* — a set of synonyms, which refer to a common semantic concept. Words may be present in more than one synset. For example, “*gold*” is present in the synset “*Gold, Au, atomic number 79*” and also in the different synset “*Amber, Gold, Brownish Yellow*” (this means that the word “gold” is ambiguous – it can refer to a material or to a color).

WordNet contains more than 118,000 different word forms and more than 90,000 different word senses. A list of pointers is attached to each synset, these pointers express relations between synsets. Relations are summarized in the Table 3.2.

For example, the synset “*War, State of war, Hot war, Hostilities*” has an antonym “*peace*”. A hyperonym of the “*war*” synset is “*conflict*” and one of its hyponyms is “*Terrorism*”. More than 116,000 pointers represent semantic relations between words and word senses.

Nouns make the biggest word group in WordNet — more than 60% of the total database.

| | |
|------------|--|
| Antonym | Word with opposite meaning |
| Hyperonym | Generalization of a word |
| Hyponym | Specification of a word |
| Meronym | Corresponds to the whole in a part-whole relation |
| Holonym | Corresponds to the part in a part-whole relation |
| Attribute | Relation of implication between a noun and an adjective |
| Pertain | Relation between a noun and an adjective, or an adjective and adverb based on morphological relation |
| Cause | Cause of another action |
| Entailment | Implication of another action |

Table 3.2: Types of WordNet relations.

Automatically Created Thesaurus

An automatically created thesaurus is a domain-dependent clustering of words.

Similarity between words is measured numerically, based on co-occurrence frequency or mutual information. Words in the same cluster have high tendency to occur in a similar lexical environment, in other words, the relation between terms in the same cluster are non-systematic (see page 11).

3.2.2 Differences Among Knowledge Sources

Organization WordNet and Roget's thesaurus are organized around word senses, whereas in an automatically created thesaurus, word senses are not represented explicitly.

Representation of Relations As a result of the organization difference, relations in different knowledge sources are represented in different levels of explicitness. The precision of relation classification increases in the following order: Roget's thesaurus — zero-one metrics, WordNet — 11 relation types, automatic thesaurus — distance function.

Systematic vs. Non Systematic Relations The WordNet databa-

se covers only systematic relations, whereas Roget Thesaurus and an Automatically Created Thesaurus cover also non-systematic relations.

Relations between Different Syntactic Categories At the moment, WordNet is the only source where syntactic categories are not completely connected together, there are some limited relations between nouns and adjectives and adjectives:

- Nouns refer to adjectives with attribute relations;
- Adjectives refer to nouns with attribute and pertain relations;
- Adverbs refer to adjectives with pertain relations.

Domain Dependence An automatic thesaurus is domain-dependent whereas Roget's Thesaurus and WordNet claim to be domain-independent.

Coverage A great advantage of WordNet over Roget's Thesaurus is the large number of words that it includes, its updated vocabulary of modern English, and the presence of a large number of scientific terms. However, WordNet, the Roget's thesaurus and a general dictionary do not include domain-dependent information and proper nouns; and cannot be updated automatically.

3.2.3 Appropriateness of Knowledge Sources for Computing

What are the factors that influence the accuracy of computing relatedness between two items?

1. Precondition: the items are in the lexicon.
2. The lexicon is covering different kinds of relations: systematic and non-systematic.
3. The lexicon contains information about the level of connectedness.

Roget's thesaurus fulfills the second request, but because the vocabulary is not updated and connectedness metrics are zero-one, the calculation precision decreases. In spite of the fact that WordNet does not support non-systematic relations, its rich vocabulary and accurate systematic relations among categories makes it an appropriate tool for computing, but with the following constraint: relatedness can be computed only within a single syntactic category. If relatedness must be computed on a specific domain text, the most appropriate knowledge source is an automatic domain thesaurus (AT), which contains systematic and non-systematic relations between domain terms. But then, the following issues must be addressed:

1. The construction of an automatically created thesaurus requires huge corpora;
2. How to treat the words that have low frequency in corpus, and therefore are not covered by AT.

A possible solution for the second problem would be to combine an AT with WordNet, but this is delicate to work out.

3.2.4 Using Thesaurus for Computing

In this section we present different methods for computing relatedness using Roget's thesaurus and WordNet. (We are not aware of any published work using automatic thesaurus for chain computing).

Roget's Thesaurus Using Roget's Thesaurus [24] — two words are related to each other if their stems satisfy any one of the following five conditions:

1. They both have an index entry that refers to the same category;
2. They both have an index entry that refers to a different category but one of these two categories has a pointer to the other one;
3. One has an index entry that refers to a category containing the other stem;

4. They are both contained in the same sub-category;
5. They both have an index entry that refers to a different category but these two categories have a common pointer to another category.

WordNet WordNet was used as a knowledge source by several algorithms: St-Onge & Hirst [9], Stairmand [31] and Elhadad & Barzilay [1].

1. St-Onge & Hirst [9] define three relation kinds: extra-strong (between a word and its repetition), strong (between two words connected by a WordNet relation) and medium-strong when the link between the synsets of the words is longer than one (only paths satisfying certain restrictions are accepted as valid connections).
2. Stairmand [31] considers two words connected if there is some path in WordNet between them, without limitation on the length (except for paths which include meronyms). Term linking is regulated with reference to a list of “conceptually overloaded synonym sets”, that is, nodes judged too generic to function as the justification for a link between terms. For example, “object” is connected to “lipstick” and “machine”. Stairmand manually identified approximately 500 nodes of this type.
3. Our relatedness criteria put constraints on the path length according to the type of edges: the length of a path between a node and its offspring in the hyperonym graph is not limited, meronym relations and siblings in the hyperonym graph however, are limited in length.

In short, different algorithms based on WordNet define relatedness in terms of path topology, its length and the type of links composing the path.

Using Word Distribution in Text for Computing Relatedness

What other factors influence relatedness? The distance between two words in the text is one. Consider two words “*machine*” and “*device*”,

which have a hyperonym relation, and are connected by all the criteria presented above. However, the probability that they are actually connected is much higher if “*machine*” and “*device*” belong to adjacent sentences than to two different parts of the text. Hirst & Morris considered the influence of distance on chain formation — “when distances between relate-able words are not tightly bound . . . the chances of incorrect chain linkages increase.” The question is how to formulate what can be considered as an “allowed distance”, or, in other words, in what units can we measure this distance?

Different definitions of the distance were presented in the literature.

Fixed word window Stairmand defined maximal distance between two related items as 80 words.

Fixed sentence window Hirst & Morris defined maximum distance according to the type of relations between items: for extra-strong relations, there is no distance limit, for strong relations, there is a limited window of seven sentences; and for medium-strong relations, it is three.

Segmentation We propose that the distance between related words is related to the topic distribution in the text. If two words are used in the description of the same subtopic, then the chance that they are related increases. Therefore, our definition of maximal distance is a topic segment. We first rely on Hearst’s algorithm [8] for approximating segmentation and discuss in the chapter on evaluation other possible ways to approximate topic distribution.

3.3 Candidate Words

The way candidate words are selected is related to properties of the knowledge source used to identify the relations between different syntactic categories. Since Roget’s thesaurus contains connections between all parts-of-speech, all words in a text can be included in chains. Hirst & Morris choose as candidate words all words except pronouns, prepositions, verbal auxiliaries, and other high-frequency words.

Since WordNet does not support connections between different syntactic categories, except nouns and adjectives, candidate words must be of the same part-of-speech. All existing algorithms rely basically on nouns as candidate words, because nouns are the main contributors to the “aboutness” of a text, and noun synsets dominate in WordNet.

St-Onge & Hirst limit the chaining process to nouns. They do not use a part-of-speech tagger “in order to avoid the slowdown and the error that would have resulted” (p. 13). Instead, the algorithm selects tokens that happen to occur as nouns in WordNet, under the assumption that “most words that exist as nouns, but that are used in different grammatical categories, are semantically close to their noun form (e.g. “to walk” and “walk”).” St-Onge claimed morphological transformation resulted in chaining inaccuracy.

Stairmand takes a different view on the tagging and morphological transformation. He considers primarily nouns and adjectives, but derivational relations between verbs and nouns are also accounted for. A tagging program (the *style* utility from UNIX BSD4.2) processes each text and assigns a part of speech to each text token. This allows selection of only those terms labeled as nouns and adjectives, which constitute the candidate terms. In addition, the Edinburgh Morphological Analyzer is used to identify repetition based on morphological variants, for example “*smoking*”, “*smoke*” and “*smoker*”.

We use the results of Brill’s part-of-speech tagging algorithm to identify nouns. In addition, we extend the set of candidate words to include noun compounds. We first evaluate empirically the importance of noun compounds by taking into account the ones explicitly present in WordNet (around 50,000 entries in WordNet are noun compounds such as “*sea-level*” or collocations such as “*digital-computers*”). However, the English language includes a productive system for noun compounds, and in each domain, new noun compounds and collocations can be created, these will not be present in WordNet, but they can play a major role.

We addressed this issue by using a shallow parser (provided by Ido Dagan) to identify noun compounds using a simple characterization of noun sequences (as regular expressions over part-of-speech tags). When a noun compound is selected, the relatedness criterion in WordNet is used by considering its head noun only. This has two major benefits:

1. It identifies important concepts in the domain (for example, in a text on “*quantum computing*”, the main token was the noun compound “*quantum computing*” which was not present in WordNet);
2. It eliminates words that occur as modifiers from the set of possible candidates for chain membership. For example, when “*quantum computing*” is selected as a single unit, the word “*quantum*” is not selected. This is beneficial because in this example, the text was not about “*quantum*”, but more about computers.

To summarize, there is an agreement that in WordNet nouns are basic part-of-speech, but there is disagreement about the usage of grammatical and morphological tools for preprocessing.

3.4 Choice of a Receiving Chain

Selecting an appropriate chain to receive a candidate word is equivalent to disambiguating the given word in the current context. Consider the word “*computer*”, it has two senses in WordNet: “*person that computes*” and “*information processing system*”. If the chain {*pc, data processor, machine*} already exists, then “*computer*” will be inserted in it and through this decision “*computer*” will be disambiguated to its first sense. If only the chain {*person, estimator*} exists, then “*computer*” will relate to it in the second sense. The problem arises when these two chains are active in the current context, and the algorithm must decide which chain will receive the word “*computer*”?

Hirst & Morris did not require the same word to appear with the same sense in its different occurrences for it to belong to a chain. In the above example the “*person*” chain and the “*PC*” chain would be merged into a single chain, connected by the two senses of “*computer*”. Mixing senses of semantically ambiguous words however, can lead to confusions.

Hirst & St-Onge choose the appropriate chain according to the type of the relation between a candidate word and the possible chains. To find a chain in which to insert a given candidate word, extra-strong relations are preferred to strong-relations and both of them are preferred to medium-strong relations. If a chain is found, then the candidate

word is inserted with the appropriate sense, and the senses of the other words in the receiving chain are updated, so that every word connected to the new word in the chain relates to its selected senses only. If no chain is found, then a new chain is created and the candidate word is inserted with all its possible senses in WordNet.

The greedy disambiguation strategy implemented in this algorithm has some limitations illustrated by the following example:

*Mr. Kenny is the **person** that invented an anaesthetic **machine** which uses **micro-computers** to control the rate at which an anaesthetic is pumped into the blood. Such **machines** are nothing new. But his **device** uses two **micro-computers** to achieve much closer monitoring of the **pump** feeding the anaesthetic into the patient.*

According to Hirst & St-onge’s algorithm, the chain for the word “*Mr.*” is first created, holding one sense:

[lex “Mr.”, sense {mister, Mr.}].

“*Mr.*” belongs only to one synset, so it is disambiguated from the beginning. The word “*person*” is related to this chain in the sense “*a human being*” by a medium-strong relation, so the chain now contains two entries:

[lex “Mr.”, sense {mister, Mr.}]

[lex “person”, sense {person, individual, someone, man, mortal, human, soul}].

When the algorithm processes the word “*machine*”, it relates it to this chain, because “*machine*” in the first WordNet sense (“*an efficient person*”) is a holonym of “*person*” in the chosen sense. In other words, “*machine*” and “*person*” are related by a strong relation. In this case, “*machine*” is disambiguated in the wrong way, even though after this first occurrence of “*machine*”, there is strong evidence supporting the selection of its more common sense: “*micro-computer*”, “*device*” and “*pump*” all point to its correct sense in this context — “*any mechanical or electrical device that performs or assists in the performance*”.

This example indicates that disambiguation cannot be a greedy decision. In order to choose the right sense of the word we must consider the ‘whole picture’ of chain distribution in the text. We propose to develop a chaining model according to all possible alternatives of word senses and then choose the best one among them.

3.5 The Dynamic Chaining Algorithm

Let us illustrate our method on the above example. First, a node for the word “*Mr.*” is created [lex “*Mr.*”, sense {*mister*, *Mr.*}]. The next candidate word is “*person*”. It has two senses: “*human being*” (*person*₁) and “*grammatical category of pronouns and verb forms*” (*person*₂). The choice of sense for “*person*” splits the chain world to two different interpretations as demonstrated in Figure 3.1.

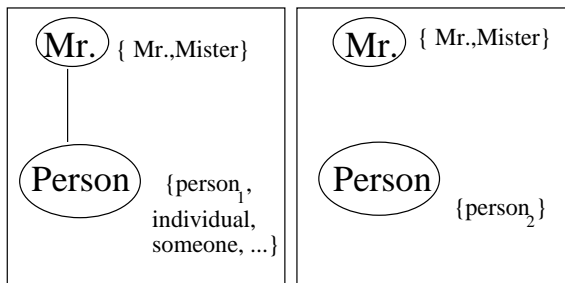


Figure 3.1: Step 1: Interpretations 1 and 2.

Define a *component* as a list of interpretations that are mutually exclusive of each other. Component words influence each other in the selection of their respective senses.

The next candidate word “*anaesthetic*” is not related to any word in the first component, so we create a new component for it with a single interpretation.

The word “*machine*” has 5 senses *machine*₁ to *machine*₅. In its first sense, “*an efficient person*”, it is related to the senses “*person*” and “*Mr.*”. Therefore, it influences the selection of their senses, thus “*machine*” has to be inserted in the first component. After its insertion, the picture of the first component becomes the one shown in Figure 3.2.

If we continue the process and insert the words “*micro-computer*”, “*device*” and “*pump*”, the number of alternatives greatly increases. The strongest interpretations are given in Figure 3.3.

Under the assumption that the text is *cohesive*, we define the best interpretation as the one with the most connections (edges in the graph).

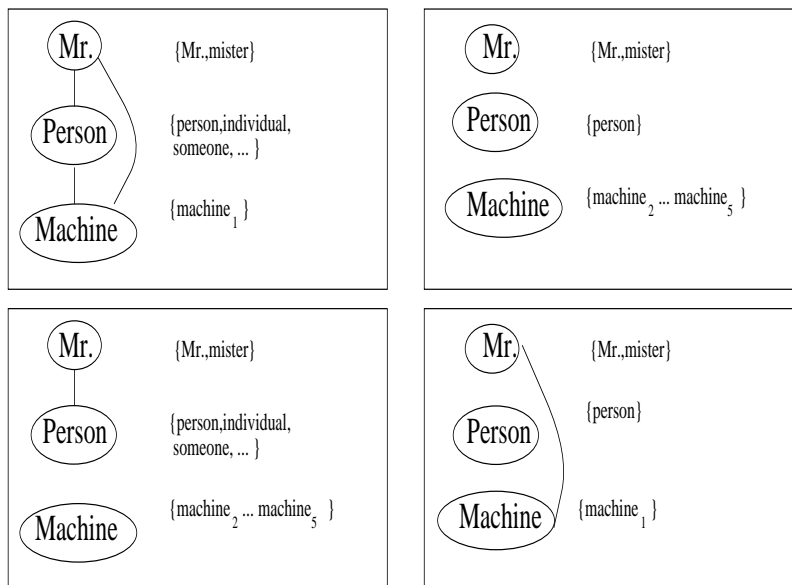


Figure 3.2: Step 2: Interpretations 1-4.

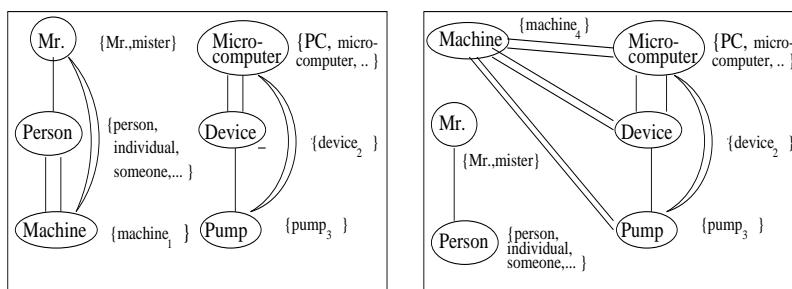


Figure 3.3: Step 3: Interpretations 1, 2.

In this case, the second interpretation at the end of Step 3 is selected, which predicts the right sense for “*machine*”. We define the *score* of an interpretation as the sum of its *chain scores*. The score of a chain is determined by the number and weight of the relations between chain members. Experimentally, we fixed the weight of reiteration and synonym to 10, of antonym to 7, and of hyperonym and holonym to 4. Our algorithm develops all possible interpretations, maintaining each one without self contradiction. When the number of possible interpretations is larger than a certain threshold, we prune the weak interpretations according to this criteria. In the end, we select from each component the strongest interpretation.

To conclude, the main difference between our algorithm and the Hirst & St-Onge algorithm is that it introduces a non-greedy disambiguation heuristic to select the appropriate senses of chain members, in addition to the relatedness criterion for membership to a chain.

3.6 Dynamic Chaining Algorithm Description

Having explained the intuitive motivation behind our approach, we present a more detailed description of the Dynamic Chaining Algorithm (DCA) in this section.

3.6.1 General Explanation

The DCA algorithm is dynamic: the decision about the right sense of a word is delayed, therefore, several exclusive chains coexist at the same time. For example, when processing the sequence of nouns “*water*”, “*ocean*”, the algorithm creates two alternative combinations:

1. A chain that contains the word “*water*” in the sense “body of water” and “*ocean*” with the same sense;
2. two separate chains of “*water*” and “*ocean*”, each with all possible senses of these words, excluding connected senses (senses that are chosen for the first alternative).

We call each such alternative an *interpretation*.

Suppose that the next two words in the text are “*computer*” and “*person*”. These can be mutually connected by a sense of “*computer*” (as a computing person) and the second sense of “*person*” (a human being). Sense selection in the pair “*computer*”–“*person*” is independent from the sense selection in the pair “*water*”–“*ocean*”.

In order to reduce the total number of interpretations, we keep independent words in different interpretations. In our example, there are 4 interpretations — two interpretations, contain the words “*computer*” and “*person*” and no other words, and two interpretations, contain the words “*ocean*” and “*water*” and no other words.

Sets of interpretations, which contain the same set of words, are mutually exclusive: if two interpretations contain one word, then they both contain this word in different senses, and no other interpretation, that do not hold this word, is concerned about it. In the above example, the “*computer*”–“*person*” interpretation will never hold any information about the two other words, unless they are united by an additional word that is related to both. We called such a set of interpretations a *component*.

Words that can be related to each other are contained in the same component, various subsets of these words build interpretations of the component.

3.6.2 Data Structures

entry In the computation of lexical chains, the following information is kept for each word in a chain — an entry:

lex-head The head of a noun compound or the single noun itself;

lex Noun compounds with the lex-head; (in the case of single noun this field is equal to the previous one)

sense Initially, holds all possible synsets of the word extracted from WordNet, reduced during computation;

sentences A list of sentences that contain the word;

relations A list of chain members that the word is related to (the relation types is kept as well).

lexical-chain A set of chain entries.

interpretation a set of chains that can all co-occur without contradictions;

component a set of interpretations that are mutually exclusive.

3.6.3 Functions

We describe our algorithm function by function, from the top-level one (**build-chains**) down.

build-chains(text)

1. Segment the text using Hearst's algorithm.
2. For each segment call **build-segment-chains(segment)** to get a list of chains for this segment (**segments-chains** holds a list of the results)¹.
3. Call **merge-segment-chains(segments-chains)** to merge all segment chain lists.

build-segment-chains(segment)

1. Initialize **components** to be an empty list of components.
2. Find collocations in the text (*e.g.* "personal-computer"). We identify collocation by checking pairs of sequential nouns as a single token in WordNet.
3. Tokenization — separate words to tokens.
4. Tagging — tag each word with its corresponding part-of-speech, and its normalized form.
5. Select only simple nouns and noun compound heads using a shallow parser.
- ~~6. Build Chains — for each word W:~~

¹Note that this list of chains is not considered an interpretation, interpretations are used only inside the **build-segment-chains** function.

if W is a repetition of a previous occurrence then
 Add W with the same sense to the same chain
else begin
 related-components :=
 components-related-to-word(components,W) *1*
 if **related-components** is empty then begin
 create a new component with only W
 add it to **components**
 else begin
 remove **related-components** from **components**
 $U :=$ **merge-components(related-components)** *2*
 foreach interpretation l in U do
 replace l with **split-interpretation(W,l)** in U *3*
 from all new interpretations in U :
 choose **max-active-interpretations**
 with highest **interpretation-score**
 and build a new component from them
 add this component to **components**
 end
end

7. From each component in **components** choose the maximal interpretation (using **interpretation-score**).
8. Return the list of chains in all of these maximal interpretations (*i.e.* flatten the above results).

Notes

1 **related-components** is now a list of components from **components** which contain interpretations that are related to the word W .

2 U is now the Cartesian product of all interpretations that are in **related-components**. For example, if **related-components** holds

$$X = \{[(\langle 1, 2 \rangle), (\langle 1 \rangle, \langle 2 \rangle)], [(\langle 3, 4 \rangle), (\langle 3 \rangle, \langle 4 \rangle)]\}$$

then the result of **merge-components(X)** is

$$[(\langle 1, 2 \rangle, \langle 3, 4 \rangle), (\langle 1, 2 \rangle, \langle 3 \rangle, \langle 4 \rangle), (\langle 1 \rangle, \langle 2 \rangle, \langle 3, 4 \rangle), (\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle)]$$

Where in this example, “{ }” notates a list of components, “[]” a component, “()” an interpretation, “⟨ ⟩” a chain, and a number corresponds to a word.

3 `split-interpretation(W,l)` is the result of embedding *W* into the interpretation *l*, in all possible ways (into existing chains if possible, or to a new chain). The results are lists of new interpretations — components, so they are all appended together.

`components-related-to-word(components, word)`

Return all components in `components` that are related to `word`, where “related” means that `word` is a synonym, antonym or meronym of a word in some chain in some interpretation in the component. Another possibility is for the word to be either an offspring or a level 4 sibling (have a common parent with up to path of length 4 between them) of another word in the WordNet hyperonym graph.

`merge-components(components)`

Get the cross-product of `components`. In other words, create all possible interpretations that are a result of collecting one interpretation from each component in `components`, and merge them all to get the resulting component. See example in remark ***2*** above.

`split-interpretation(word, interp)`

Return a component (a list of interpretations) that consists of:

- The given `interp` with a new chain holding `word` by itself.
- The list of interpretation that is achieved by adding `word` to each chain in `interp` — only if it is possible to add it to that chain.

`max-active-interpretations`

This is a constant — the number of interpretations that can compete (co-occur together). It is currently to 10.

interpretation-score(interp)

Return the sum of chain scores of all chains in **interp**. The score of a chain **C** in **interp** is calculated by the sum of:

- foreach two words w_1, w_2 in **C** return
 - 10** if they are synonyms;
 - 8** if they are offsprings;
 - 7** if they are antonyms;
 - 4** if they are meronyms;
 - 2** if they are siblings.

merge-segment-chains(segments-chains)

Merge two chains if they contain at least one equal sense. Return a list consisting of these chains.

3.6.4 Running example

Here we give a running example of the **build-segment-chains** function.

Suppose a segment consists of the words “ x_1 ”, “ x_2 ”, “ x_3 ”, “ x_4 ”. The word “ x_1 ” has one sense, “ x_2 ” and “ x_3 ” have two senses, and are connected to each other by the first of them; “ x_4 ” has three senses — it is connected to “ x_1 ” by the first, and to the first sense of “ x_3 ” by the second.

After processing “ x_1 ”, **U** is empty, so a new component **A** with one interpretation with the one word chain “ x_1 ” is created — $[(\langle x_1 \rangle)]$.

After processing “ x_2 ”, **U** is also empty (the two words are not connected), and “ x_2 ” is contained in a new component, so we have two **components**: $\{[(\langle x_1 \rangle)], [(\langle x_2 \rangle)]\}$.

After getting “ x_3 ”, **U** contains one component $[(\langle x_2 \rangle)]$. The interpretation in this component is now split to two: one that contains a chain with “ x_2 ” and “ x_3 ”, and another contains a single word chain with “ x_2 ” and a single word chain “ x_3 ”. These two interpretations creates a new component, that replaces the old one, so now we have:

$$\{[(\langle x_1 \rangle)], [(\langle x_2, x_3 \rangle)], (\langle x_2 \rangle, \langle x_3 \rangle)]\}$$

Now there are two components. When “ x_4 ” is processed, all components are related to it. Now, \mathbf{U} contains the Cartesian product of both components, which is:

$$[(\langle x_1 \rangle, \langle x_2, x_3 \rangle), (\langle x_1 \rangle, \langle x_2 \rangle, \langle x_3 \rangle)]$$

Both interpretations are split according to the different senses of “ x_4 ”. The first interpretation is split to

$$\begin{aligned} &(\langle x_1, x_4 \rangle, \langle x_2, x_3 \rangle) \\ &(\langle x_1 \rangle, \langle x_2, x_3, x_4 \rangle) \\ &(\langle x_1 \rangle, \langle x_2, x_3, x_4 \rangle, \langle x_4 \rangle) \end{aligned}$$

and the second to

$$\begin{aligned} &(\langle x_1, x_4 \rangle, \langle x_2 \rangle, \langle x_3 \rangle) \\ &(\langle x_1 \rangle, \langle x_2 \rangle, \langle x_3, x_4 \rangle) \\ &(\langle x_1 \rangle, \langle x_2 \rangle, \langle x_3 \rangle, \langle x_4 \rangle) \end{aligned}$$

So the new value of **components** is one component which is composed of the above six interpretations.

If there are only “ x_1 ”, “ x_2 ”, “ x_3 ” and “ x_4 ” in the paragraph, then according to the highest value given by the score function — one interpretation is to be chosen.

3.6.5 Scoring Chains

Not all of the algorithm’s output chains represent topic concepts of the text, therefore one must first identify the strongest chains amongst. There is no formal way to evaluate chain strength, we therefore rely on an empirical methodology. We have developed an environment to compute and graphically visualize lexical chains to evaluate experimentally how they capture the main topics of the texts. Figure 3.4 shows how lexical chains are visualized to help human testers evaluate their importance.

We have collected data for a set of 30 texts extracted from popular magazines (“The Economist” and “Scientific American”), all are of popular science genre. For each text, we manually ranked chains in terms of relevance to the main topic. We then computed different formal measures on the chains, including: chain length, distribution in

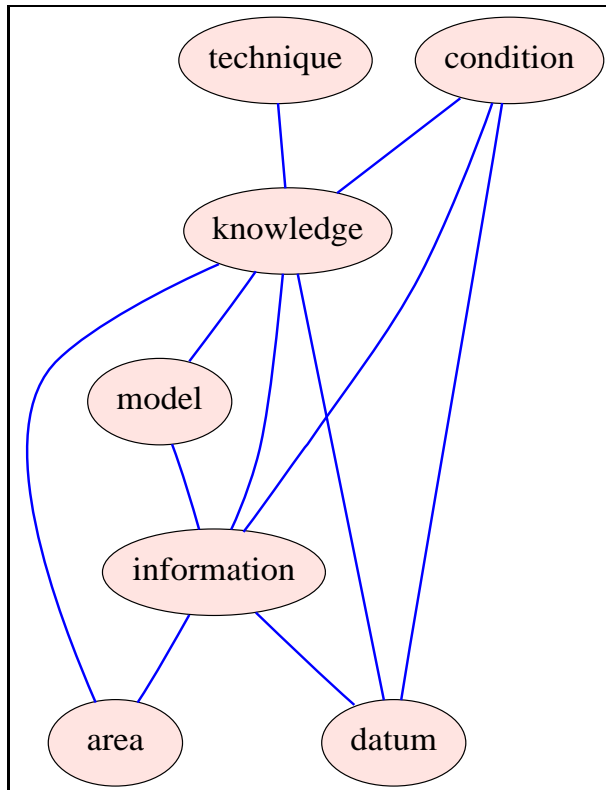


Figure 3.4: Visual representation of lexical chains.

the text, text span covered by the chain, density, graph topology (diameter of the words graph through WordNet relations) and number of repetitions. The results on our data set indicate that only the following parameters are good predictors of a chain strength:

Length: The number of occurrences of chain members.

Homogeneity index: One minus the number of distinct occurrences divided by the length.

We designed a score function for chains as:

$$\text{Score}(\text{Chain}) = \text{Length} * \text{Homogeneity}$$

When ranking chains according to their score, we estimate that strong chains are those which satisfy our “Strength Criterion”:

$$\text{Score}(\text{Chain}) > \text{Average}(\text{Scores}) + 2 * \text{StandardDeviation}(\text{Scores})$$

These are preliminary results but they are strikingly confirmed by our experience on 30 extensively analyzed texts. We have experimented with different normalization methods for the score function, but they do not seem to improve these results.

The average number of strong chains selected by this method is 5 for texts of 1055 words on average (minimum 474, maximum 3198), where 32 chains were originally generated on average.

3.7 Summary: Building Lexical Chains

In this chapter we have presented the DCA algorithm for building lexical chains on an arbitrary text, relying on WordNet as a lexical database. The DCA is characterized by the following design decisions:

- Knowledge source: WordNet and systematic relations only.
- Relatedness condition: constraints on the path length according to the type of edges.
- Word distribution: if two words are used in the description of the same subtopic, then the chance that they are related increases.
- Candidate selection: we select nouns marked by Brill’s part-of-speech tagger, together with noun compounds found in WordNet and noun compounds identified by a shallow parser.
- Clustering strategy: our algorithm defines a dynamic, non-greedy approach to the definition of which active chain can receive a candidate word.

We evaluate most of these design decisions in the following chapter, defining an intrinsic evaluation strategy for the lexical chainer first.

Chapter 4

Evaluating the Lexical Chainer

4.1 Evaluating Methods

4.1.1 Evaluation Framework

Our objective is to evaluate the quality of semantic clustering that is produced by the DCA algorithm. Using the terms of Karen Sparck Jones [15] this is a task-motivated [non-observable, internal to theory] evaluation, with intrinsic orientation. Chain quality cannot be evaluated directly. Instead, we evaluate the impact of observable parameters on chains.

Evaluation Design

- **Performance Factors**

System parameters:

- Dynamic vs. greedy
- Text division
- Candidate words
- Number of competitive alternatives.

Environment variables:

- Coherence/cohesion property of text

- Thesaurus properties: average number of senses, number of unknown words and connections.

- **Evaluation Criterion: Clustering Quality**

Performance measures

- Topic identification
- Disambiguation of important words.

As mentioned above there is a close connection between word-sense disambiguation and lexical chaining. However, according to our strategy, we have enough evidence to disambiguate only important words.

On the other hand, we are looking only at clusters that represent the main text topic, so disambiguation of main cluster words is an important attribute for measuring our criterion. This measure is divided into two sub-measures: topic identification and disambiguation. These are related to different clustering attributes, but they are not independent — correct identification of main terms is a preliminary condition for disambiguation accuracy (validity).

Application methods: The next question after setting the measures is what methods to be used for applying it. All methods we use are quantitative methods:

- Disambiguation ratio
- Sense reduction
- Sense disambiguation based on strong chain
- Precision/recall for topic identification
- Hard/weak metrics

- **Evaluation Data** The data that was used for evaluation consists of 37 scientific articles from the Semantic Concordance corpus (SemCor). The words in SemCor are tagged with word senses from WordNet. The tagging was done manually.

4.1.2 Topic Identification Measure

The goal of this measure is to identify that our disambiguation does not have an impact on the selection of important words. Our operative definition of “important words” are words that belong to chains whose score are greater than the cutoff score. The *application method* used is:

1. Build chains using a disambiguated corpus — *ideal chains*
2. Build chains using the algorithm being checked — *output chains*
3. Find important words derived from *ideal chains* and *output chains*. Calculate the precision and recall between these two sets.

Results: The results, presented in the following table, show high agreement in identification of important terms.

| | Precision | Recall |
|-----|-----------|--------|
| DCA | 92 | 72 |

We can conclude that our disambiguation strategy does not have an impact on the selection of important words.

4.1.3 Word Sense Disambiguation Ratio

In this section we describe our application methods for measuring word sense disambiguation.

Metrics for Disambiguation

The traditional disambiguation metric is a zero-one function — 1 for equal senses and 0 otherwise.

This metric is *strict* in the sense that it does not consider semantic distance between distinct senses. In addition, we use a weaker metrics that considers the fact that a word is disambiguated correctly if its sense is “related” to the right sense. The operational definition of relatedness is the same as the one used in the chaining algorithm (see page 21). How can this weak metrics contribute?

1. Weak metrics are more appropriate for evaluation of the chaining process, because we are satisfied by any sense that relates a word to its correct cluster. If there are several such senses, then choosing between them does not influence the resulting chains.
2. The difference between the evaluations based on these metrics can help to characterize errors in the disambiguation methods. In other words, to see how “hard” is an error.

Sense Reduction vs. Sense Disambiguation

According to our disambiguation strategy, a word can belong to two different chains. In other words, the algorithm does not make a final disambiguation decision, but only reduces the number of acceptable senses. For example, all occurrences of a particular word in some paragraphs can be disambiguated to sense one, and in other paragraphs to sense two. From 4.5 possible senses on the average for the candidate words, the algorithm reduces senses to 1.9. Therefore, one of the measures of disambiguation is sense reduction.

Another possibility is to evaluate complete disambiguation: for each word that has more than one sense, calculate how many times the right sense is selected. For example, if from 15 occurrences of the word in the text, 9 occurrences were disambiguated to the right sense and 6 occurrences to the wrong one, then the disambiguation of the word is 0.6.

This traditional disambiguation measure is not very accurate in our case — we use reduction to more than one sense, and this stands against the traditional assumption that a word has only one sense all over a text¹. We are interested in finding something that is between the first and the second measures — a method that reduces sense sets to single senses. In the case that our algorithm did not choose one, we choose the sense of the word that connects it to the stronger chain for the purpose of evaluation.

¹To demonstrate the fact that multi-sense words are rare, we saw that in our disambiguated corpus only 4.5% of all nouns have 2 senses and 0.5% have 3 senses in the same text.

For example, suppose a word w appears ten times in a text, seven occurrences in one chain C_1 , and three in another chain, C_2 — and suppose that the C_1 sense is correct. According to the first measure, w was 100% correctly disambiguated; according to the second it is 70% correctly disambiguated; and with the third measure it is 100% if C_1 is stronger and 0% if not.

4.2 Influence of Performance Factors

In this section we consider different factors that influence chaining: system parameters and environment variables.

4.2.1 Influence of Algorithm Parameters

We first check our primary assumptions about the values of the algorithm parameters: *candidate terms*, *segmentation* and *dynamic vs. greedy strategy*, using disambiguation ratio as an evaluation measure.

Candidate Terms

We first evaluate the influence of candidate terms. Our reason to use noun compounds instead of nouns is based on the assumption that they represent better topicality (concept) than single nouns.

We compare the influence of addition of noun compounds to candidate words instead of single nouns. The rest of the parameters are set in the following way: division according to Hearst's segmentation, and dynamic strategy.

Results: The results of this comparison are shown in Figure 4.1. As seen, the difference between using single noun candidate words and our strategy of using noun compounds is slightly in favor of single nouns, up to around the important 50% of all words. This is according to all three measures taken.

What can be learned from these results? In contradiction to our assumption, the addition of noun compounds does not improve

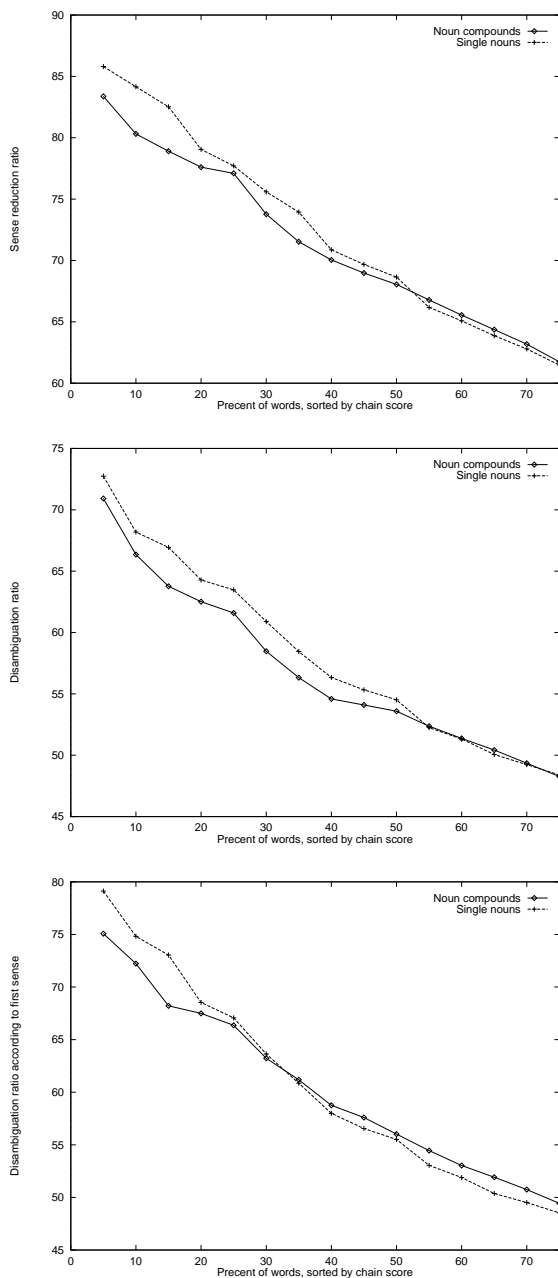


Figure 4.1: Noun compound vs. single nouns.

disambiguation. The interesting question is why single nouns yield better disambiguation results up to some boundary (which is specific to the application methods), and only beyond that boundary does our choice become better. A possible explanation can be the fact that a number of important nouns appear in the text both as modifiers and as heads, so if an important noun was disambiguated correctly we add all its appearances (including the ones in modifier roles) and in this way increase the disambiguation measure.

Segmentation

We now check our hypothesis about the influence of segmentation on the chaining process: using segments that are identified by Hearst's algorithm as text units improves clustering. The issues to be verified are:

- Does the division of the text into pieces improve chaining?
- Is Hearst's segmentation better than other possible ones?

Evaluation procedure

1. Compare segmentation against chaining the whole text;
2. Compare Hearst segmentation against use of all paragraph boundaries;
3. Compare Hearst segmentation against randomly chosen paragraph boundaries.

We treat dynamic and greedy strategy separately.

Results: The results are in Figures 4.2 through 4.5.

- **The Greedy strategy**

Sense reduction There is a meaningful improvement when using the division strategy over chaining the whole text, more than 10%. There is no significant difference between using paragraphs using Hearst's segments. Random segments give slightly worse results.

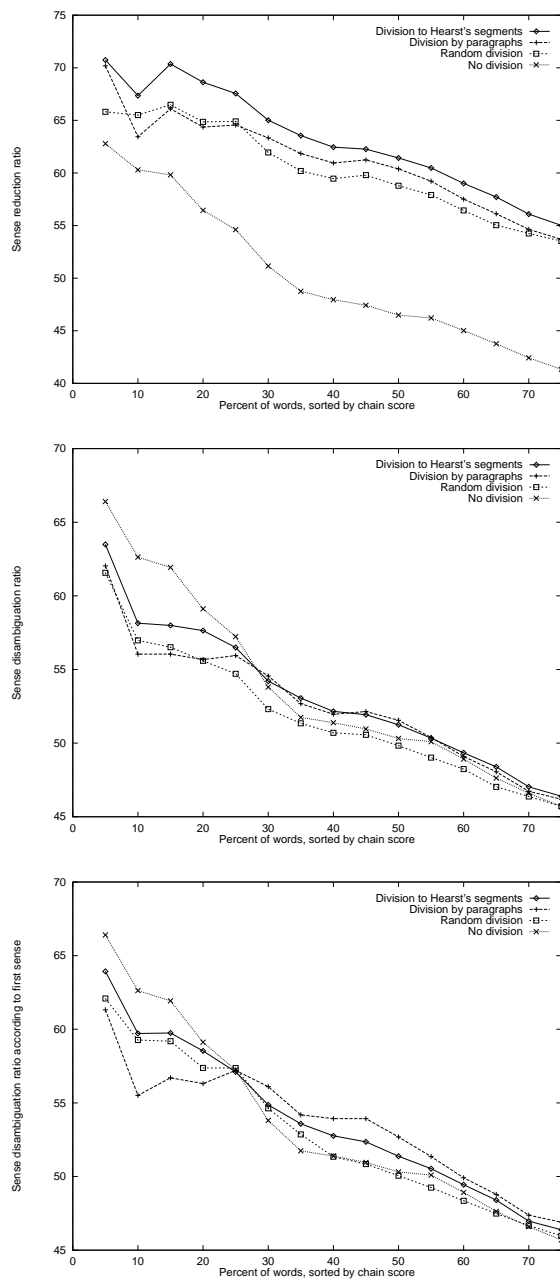


Figure 4.2: Division strategies comparison, the greedy case (strong metrics).

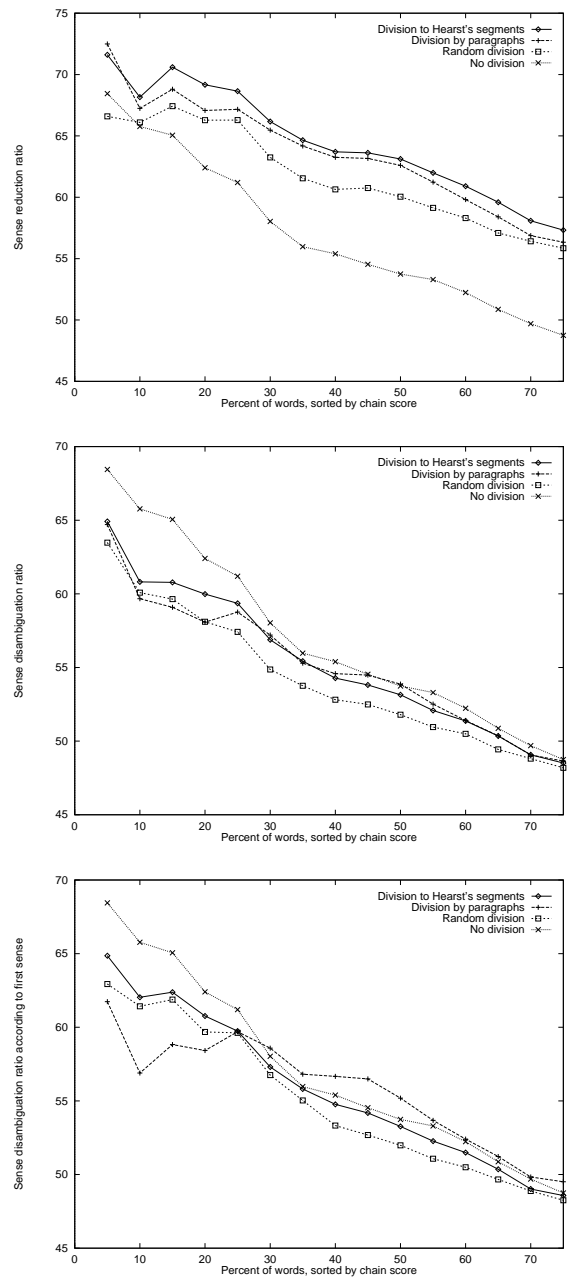


Figure 4.3: Division strategies comparison, the greedy case (weak metrics).

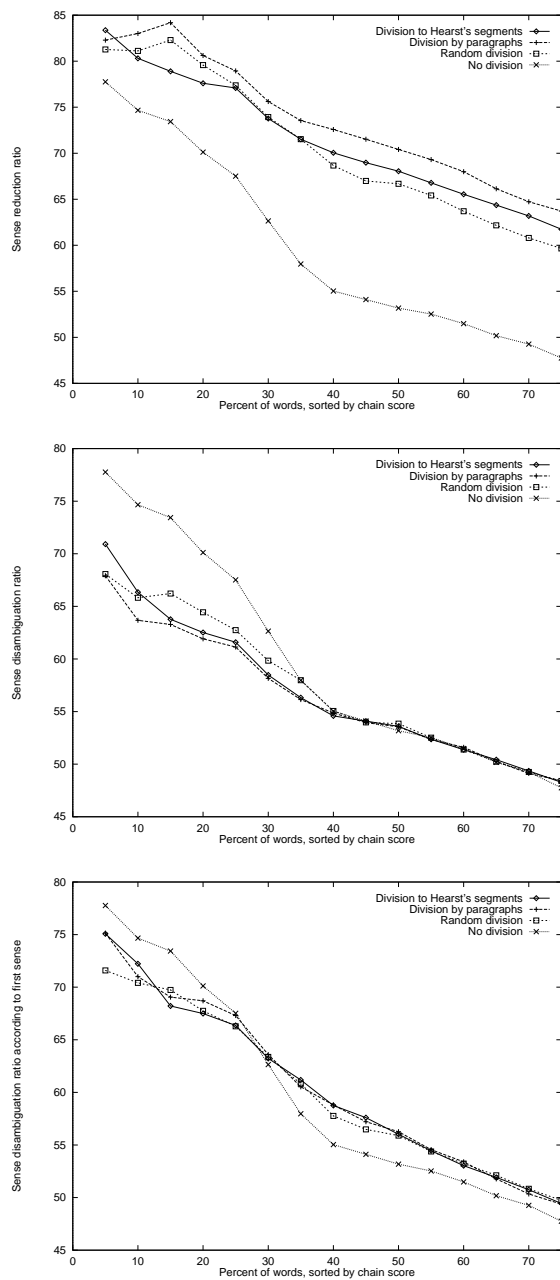


Figure 4.4: Division strategies comparison, the dynamic case (strong metrics).

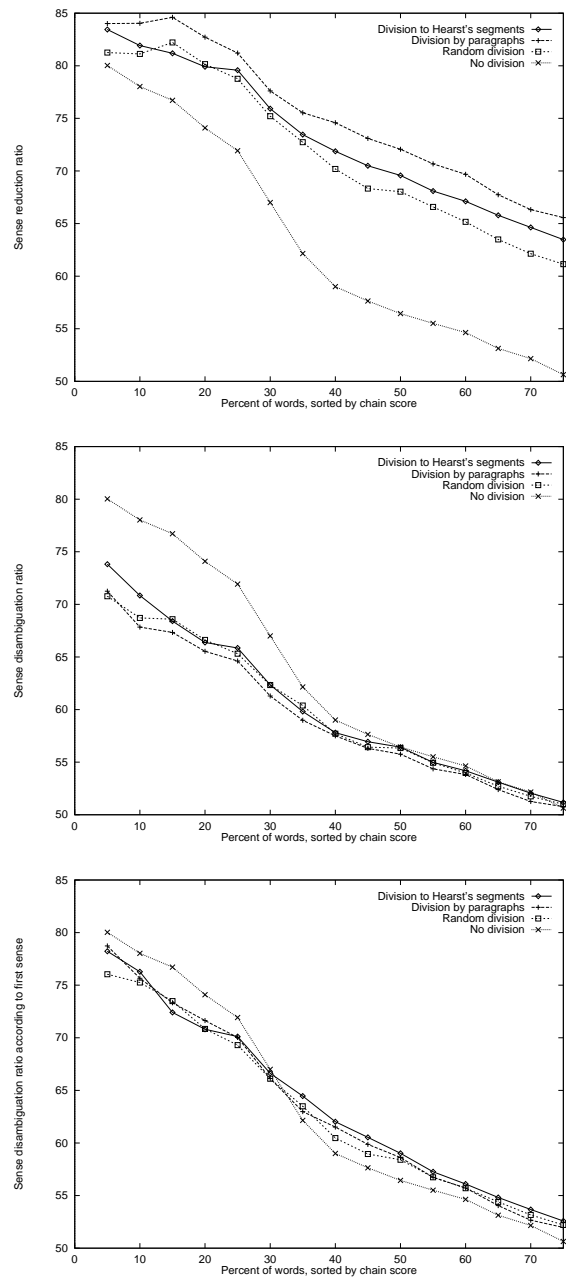


Figure 4.5: Division strategies comparison, the dynamic case (weak metrics).

Sense disambiguation Using the whole text strategy is slightly better. No difference between division on paragraphs and division on Hearst segments. Random segments give slightly worse results.

Sense disambiguation based on strong chain The four methods produced roughly the same results, with a slight advantage of no division when taking less than 20% of the important words.

- **The Dynamic strategy**

Sense reduction Using divided text is much better than using the whole text method — an improvement of 15%. Paragraph based segmentation is a bit better than Hearst segmentation.

Sense disambiguation Using the whole text strategy is better than using divided texts on the first 20% of important words, and there is no difference from 25% of words.

Sense disambiguation based on strong chain There was no significant difference among the four division methods.

What can be learned from these results? There is no significant influence of the segmentation on the greedy vs. dynamic strategies. Therefore, we do not distinguish between them in the following discussion.

- By the fact that sense disambiguation based on strong chains gives the same results over all four methods, we can assume that in this case the three division methods approximate the whole text chaining strategy, which tries to build the longest chains all over the text and choose senses that contribute to this.
- There is a significant change from the difference between using division and using the whole text method in sense reduction, and the differences in sense disambiguation and sense disambiguation which is based on strong chains. This shows that the dynamic strategy has a big potential in disambiguation, but additional

techniques and knowledge sources are needed to pinpoint the right sense after this first reduction step.

- The results contradict our assumption about the importance of Hearst segmentation in the chaining process. Division based on paragraphs is not worse, and sometimes better than based on Hearst segmentation. This can be explained by the genre of the articles used — scientific papers usually have good division to segments marked as paragraphs.
- Random division is not as bad as was expected. We have no explanation for this fact.
- The Difference between the usage of the two metrics was the same in all four methods — weak metrics add around 4% to the results.

To conclude — division contributes to chaining, and paragraphs seem to be good enough to improve chaining.

Dynamic vs. Greedy

In this section we check our hypothesis that dynamic strategy improves chaining. We compare the dynamic strategy against the greedy strategy, setting other parameters of the algorithm in the following way: division according to Hearst segmentation, and candidate words that are chosen are noun compounds.

Results: The results are in Figures 4.2 through 4.5.

All of our disambiguation measures shows a *significant advantage* of the dynamic strategy yields over the greedy one.

The difference in the sensitivity of the methods to the type of metric shows that the dynamic method tends to make less severe mistakes than the greedy method.

4.2.2 Influence of Environment Properties

We can divide environment properties to two groups: text properties and thesaurus properties. We are limited in doing real comparative analysis here, because the only thesaurus we can use is WordNet.

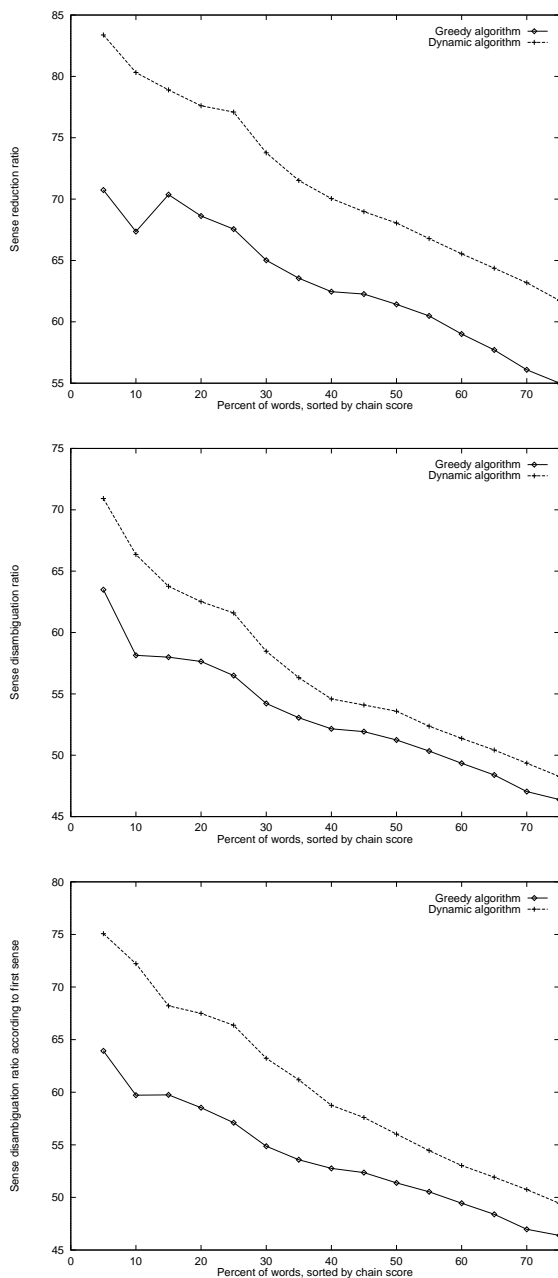


Figure 4.6: Greedy vs. dynamic strategies, using strong metrics.

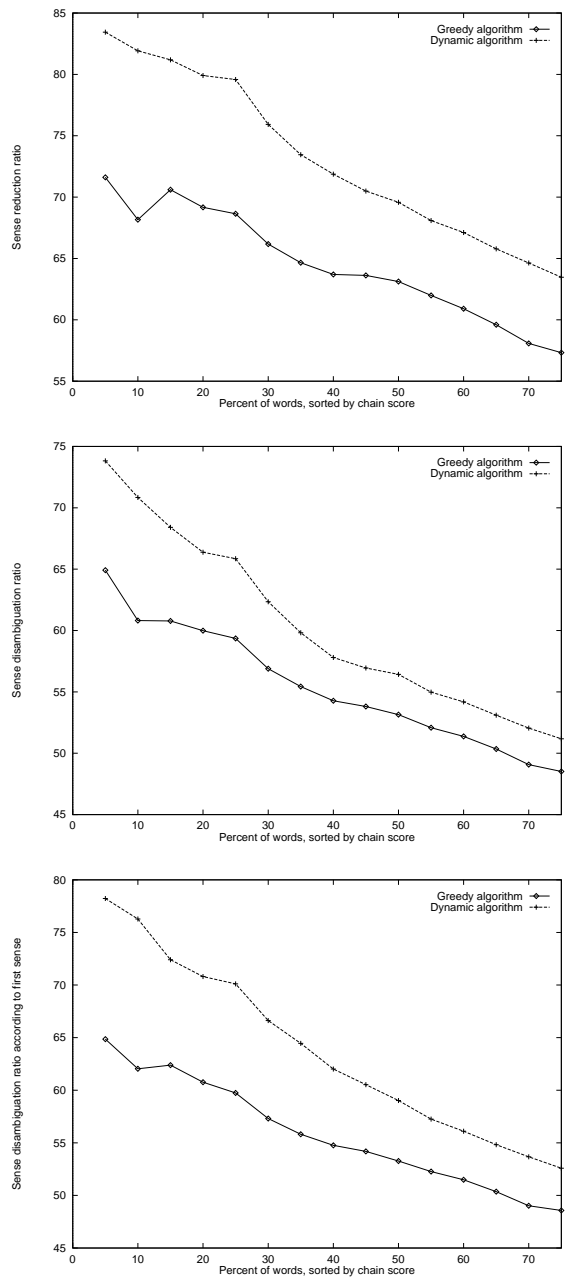


Figure 4.7: Greedy vs. dynamic strategies, using weak metrics.

Thesaurus Influence

The average number of senses in our evaluation data is 4.5. The maximal number was reached for the word “*line*” — 29 senses. WordNet does not contain any information about frequency of senses², it does not distinguish between frequent senses and “exotic” ones (*e.g.* “*machine*” with the sense of an “efficient person”). Our hypothesis is that reduction of senses to “*common*” senses can improve chain quality.

We check this hypothesis in the following way: we reduce the original senses of polysemous words in the disambiguated corpus to two senses — the correct one and one that is chosen randomly from the original senses. We also performed the same pruning of senses to three.

The results, according to three evaluation methods, show meaningful improvement in disambiguation when two senses are used: sense reduction gives 87% on 40% of the important words, sense disambiguation ratio are around 83% on 40% of important words. Pruning WordNet to three senses gives slightly worse results, but they are still significantly better than the results based on WordNet with 4.5 senses in average: sense reduction gives 84%, sense disambiguation ratio are around 78% on 40% of the important words.

After reducing the number of senses to two, the difference between the greedy and the dynamic strategy is 3%, and after reducing to 3 senses the difference is 5% (in the original WordNet the difference reaches 10%).

What can be learned from these results?

- The algorithm is very sensitive to the number of senses in the thesaurus. “Cleaning” WordNet of rare senses or some preprocessing, based on knowledge other than coherence, can be very useful.
- The greedy algorithm is much more sensitive to the number of senses. In the two senses case the two strategies are very similar.

²This work was written before the WordNet database was modified — now it includes some sense frequency information. Time constraints, and lack of updated disambiguated corpus did not allow testing our system with it.

This fact shows that on some kinds of thesaurus, the greedy strategy is sufficient. But as the number of senses grows, the dynamic strategy proves more robust than the greedy one.

Cohesion Properties

The cohesive property of the text is a dominant factor in chain construction. We assume that there is flow of connected information from sentence to sentence, and the words, related according to common sense, represent the same concept in a particular context. When the text is more cohesive, there is more data for building lexical chains.

In order to check how much cohesion influence the performance of our algorithm, we compare cohesive texts with non-cohesive texts. We create non-cohesive text by pasting sentences from two texts on different topics. We compare the disambiguation measures on source texts and on the words of the source texts, when they are part of the non-cohesive text.

The results proves our assumption. The difference in sense reduction was 30%. There was no significant difference between sensitivity to coherence between the greedy and the dynamic algorithm.

Conclusions: the algorithm is very sensitive to the cohesiveness of the text, therefore, it cannot be used on low-cohesive texts.

4.3 Summary: Lessons Learned from the Evaluation

- We found that there is a crucial dominance of environment factors over system parameters for the performance of our algorithm. In fact, changes in the thesaurus quality and text coherence features have an immense effect on the quality of chains produced by our algorithm.
- Our results also implies that all chaining algorithms that are derived from the generic algorithm presented above, must be used only on coherent texts.

- As mentioned above cleaning WordNet of rare senses, and aggregation of similar senses can dramatically increase the performance.
- In our evaluation we verified our parameter setting as most appropriate:
 - Dynamic strategy.
 - Segmentation using paragraphs.
 - Single nouns as candidate terms.
- An unexpected result that we observed is that using Hearst's segmentation got lower quality results, compared to using textual paragraphs as segments.

Chapter 5

Application of Lexical Chains to Summarization

In this chapter we explain the motivation using text cohesion information for summarization. We present a method for sentence extraction based on lexical chains.

5.1 Motivation

A summary text is a derivative of a source text condensed by selection and/or generalization of important content. The global process model has two major phases:

- *Interpretation* of the source text and its abstraction to *source meaning representation*;
- Building summary representation using the source meaning representation.

This model emphasizes the role of text representation and the central transformation stage. Within this framework, summarization systems can be characterized according to “the nature of their source representation, including its distance from the source text, its relative emphasis on *linguistic, communicative or domain information* and therefore the structural model it employs and the way this marks important content” [15].

There are several techniques that are used to summarize texts, some of them are based on shallow source representation:

1. For example, early summarization system [18] directly exploit linguistic source information, based on the intuition that the most frequent words represent the most important concepts of the text.

This representation abstracts text into frequency table.

2. Another method which is also based on linguistic information is the *cue phrase method*, which uses meta-linguistic markers (for example, “*in conclusion*”, “*the paper describes*”) to select important phrases [4]. The cue phrase method is based on the assumption that these bonus phrases provide “rhetorical” context for identifying important sentences.

The source abstraction in this case is cue words and sentences that contain them.

3. Another information type is used in the *location method* — headings, sentences in the beginning and end of the text contain important information to the summary.
4. Discourse representation of source text was used as source abstraction by Ono [25] and Marcu [20]. Their assumption is that “the concepts of rhetorical analysis and nuclearity can be used effectively for determining the most important units of the text” (Marcu). Both algorithms are based on rhetorical markers for trees construction.

On the other hand, summaries can be built on a deep semantic analysis of the text:

1. For instance, MUC-style templates were used as abstraction of the source text by (McKeown and Radev, [21]) to produce coherent summary of several texts describing the same event. Another example of such abstraction is the DeJong scripts, that relies on inflexible specification of the kind of information sought.

2. The last type of source abstraction is the most expressive, but very domain dependent and hard to compute. We are interested in application-independent techniques for summarization.

All four shallow techniques presented above are easily computed. As reported in (Piece, [26]), location and cue phrases produce better results than the statistical method, and can be accurately computed. Discourse tree representation seems to be the most promising among all shallow techniques.

However, the big limitation of location, cue phrases and discourse tree abstraction is dependence on the text genre: the number of rhetorical markers changes critically from “Scientific American” articles to political articles. Ono reports significant difference in accuracy when building the discourse representation from technical tutorial texts to newspaper texts. These techniques can be seen as high risk stocks: if you win — you get it all, but if your loose — you loose everything.

A method that is based on word distribution does not suffer from this problem: it is a property which is not influenced by the genre of the text. Frequency is a good indicator for words that represents important concepts — this true with most texts, independently of their style.

Frequency table as source abstraction is too simplistic a source representation. Our goal is to enrich the frequency table with additional information to build a source representation of good quality, but easy enough to compute.

Over-simplification can harm the quality of the source representation. As a trivial illustration, consider the following two sequences:

1. “*Dr. Kenny has invented an anesthetic machine. This device controls the rate at which an anaesthetic is pumped into the blood.*”
2. “*Dr. Kenny has invented an anesthetic machine. The Doctor spent two years on this research.*”

“*Dr. Kenny*” appears once in both sentences and so does “*machine*”. But sentence 1 is about the “*machine*”, and sentence 2 is about the “*doctor*”. This example indicates that if the source representation

does not supply information about semantically related terms, one cannot capture the “aboutness” of the text, and therefore the summary will not capture the main point of the original text.

The use of related words in text — lexical cohesion — is one of the devices that provide cohesion. As we described in Chapter 2 lexical cohesion on one hand can be easily calculated, on the other hand, it is one of the surface signs of discourse structure. These gives us motivation to investigate lexical chains as an abstraction of the source text for the purpose of producing a summary.

5.2 Building Summaries Using Lexical Chains

We now investigate how lexical chains can serve as a source representation of the original text to build a summary. The next question is how to build a summary representation from this source representation.

The most prevalent discourse topic will play an important role in the summary. We first present the intuition why lexical chains are a good indicator of the central topic of a text. Given an appropriate measure of strength, we show that picking the concepts represented by strong lexical chains gives a better indication of the central topic of a text than simply picking the most frequent words in the text (which forms the zero-hypothesis).

For example, we show in Appendix A.1 a sample text about Bayesian Network technology. There, the concept of network was represented by the words “*network*” with 6 occurrences, “*net*” with 2, and “*system*” with 4. The summary representation has to reflect that all these words represent the *same* concept; otherwise, the summary generation stage would extract information separately for each term. The chain representation approach avoids completely this problem, because all these terms occur in the same chain, which reflects the fact that they represent the same concept.

An additional argument for the chain representation as opposed to a simple word frequency model is the case when a single concept is represented by a number of words, each with relatively low frequency.

In the same Bayesian Network sample text, the concept of “*information*” was represented by the words “*information*” (3), “*datum*” (2), “*knowledge*” (3), “*concept*” (1) and “*model*” (1). In this text, “*information*” is a more important concept than “*computer*” which occurs 4 times. Because the “*information*” chain combines the number of occurrences of all its members, it can overcome the weight of the single word “*computer*”.

5.2.1 Extracting Significant Sentences

Once strong chains have been selected, the next step of the summarization algorithm is to extract full sentences from the original text based on chain distribution.

We investigated three alternatives for this step.

Heuristic 1:

For each chain in the summary representation choose the sentence that contains the first appearance of a chain member in the text. This is based on the intuition that a sentence in which a chain begins, gives the necessary information for the identification of the concept which the chain represents, and its particular context in the text.

This heuristic produced the following summary for the text is shown in Appendix A.1:

When Microsoft Senior Vice President Steve Ballmer first heard his company was planning to make a huge investment in an Internet service offering movie reviews and local entertainment information in major cities across the nation, he went to Chairman Bill Gates with his concerns. Microsoft's competitive advantage, he responded, was its expertise in Bayesian networks.

Bayesian networks are complex diagrams that organize the body of knowledge in any given area by mapping out cause — and — effect relationships among key variables and encoding them with numbers that represent the extent to which one variable is likely to affect another.

Programmed into computers, these systems can automatically generate optimal predictions or decisions even when key pieces of information are missing.

When Microsoft in 1993 hired Eric Horvitz, David Heckerman and Jack Breese, pioneers in the development of Bayesian systems, colleagues in the field were surprised.

The problem with this approach is that all words in a chain reflect the same concept, but to a different extent. For example, in the AI chain, (Appendix A.2, Chain 3) the token “*science*” is related to the concept “*AI*”, but the words “*AI*” and “*field*” are more suitable to represent the main topic “*AI*” in the context of the text. That is, not all chain members are good representatives of the topic (even though they all contribute to its meaning).

Heuristic 2:

We therefore define a criterion to evaluate the appropriateness of a chain member to represent its chain based on its frequency of occurrence in the chain. We found experimentally that such words, call them *representative* words, have a frequency in the chain no less than the average word frequency in the chain. For example, in the third chain the representative words are “*field*” and “*AI*”.

We therefore define a second heuristic based on the notion of representative words: for each chain in the summary representation, choose the sentence that contains the first appearance of a representative chain member in the text.

In this special case this heuristic gives the same result as the first one.

Heuristic 3:

Often, the same topic is discussed in a number of places in the text, so its chain is distributed across the whole text. Still, in some text unit, this global topic is the central topic (focus) of the segment. We try to identify this unit and extract sentences related to the topic from this segment (or successive segments) only.

We characterize this text unit as a cluster of successive segments with high density of chain members. Our third heuristic is based on this approach.

For each chain, we find the text unit where the chain is highly concentrated. Extract the sentence with the first chain appearance in this central unit. Concentration is computed as the number of chain members occurrences in a segment divided by the number of nouns in the segment. A chain has high concentration if its has the maximum density of all chains in the text segment. A cluster is group of successive segments such that every segment contains chain members.

Note that in all these three techniques only one sentence is extracted for each chain (regardless of its strength).

For most texts we tested, the first and second techniques produce the same results, but when they are different, the output of the second technique is better. Generally, the second technique produces the best summary. We checked these methods on our 30 texts data set. Surprisingly, the third heuristic, which intuition predicts as the most sophisticated, gives the least indicative results. This may be due to several factors: our criteria for ‘centrality’ or ‘clustering’ may be insufficient or, more likely, the problem seems to be related to the interaction with text structure. The third heuristics tends to extract sentences from the middle of the text and to extract several sentences from distant places in the text for a single chain.

5.3 Summary: Sentence Extraction Based on Lexical Chains

In this chapter we presented an algorithm for sentence extraction based on lexical chains. This is on one hand computable and on the other hand contains more information than regular shallow techniques.

In the next chapter we compare our summarizer results with two other available summarization systems recently made available (Summer 97) and find our approach to give results significantly closer to human judges (using majority rule) than the other systems.

Chapter 6

Summary Evaluation

Most evaluations of summarization systems use an intrinsic method [4, 26, 17, 20, 30, 25]. The typical approach is to create an “ideal” summary, either by professional abstractors or by merging summaries provided by multiple human subjects using methods such as majority opinion, union, or intersection. The output of the summarizers is then compared with the “ideal” summary. Precision and recall are used to measure the quality of the summary¹.

6.1 Description of the Experiment

The goal of experiment is to compare similarity between lexical-chain based summarizer and human summaries. To study agreement of human subjects, 40 documents were selected; for each document, 10 summaries were constructed by 5 human subjects using sentence extraction. Each subject constructed 2 summaries of a document: one at 10% length and the other at 20%². For convenience, percent of length was computed in terms of number of sentences.

We also chose two additional automatic summarizers: the Microsoft summarizer (in Word97) and a summarizer based on discourse structure

¹In [14] we show the limitations of current evaluation methods, however, we still use these methods in this work because there is no reasonable alternative yet.

²According to [14] ideal summary based evaluation is extremely sensitive to the required summary length. Therefore, we use an evaluation of 10% and 20% summaries in order to decrease the bias of the length factor.

| Document Number | | 536075 | | | | | | | | | | | | | | | |
|-----------------|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|---|
| Document Title | | State-Sponsored Death Squads Blocking Third World Development | | | | | | | | | | | | | | | |
| Sent Num | Sub1 | | Sub2 | | Sub3 | | Sub4 | | Sub5 | | SysA | | SysB | | SysC | | |
| | 10—20 | | 10—20 | | 10—20 | | 10—20 | | 10—20 | | 10—20 | | 10—20 | | 10—20 | | |
| 1 | + | + | + | + | + | + | | | | + | + | | | | | + | + |
| 2 | + | + | + | + | | + | | | | + | + | | | + | + | | |
| 3 | | | | + | | | | | | | | | | | | | |
| 4 | | | | | | | | + | + | | | | | | | + | + |
| 5 | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | |
| 7 | | | | | | + | + | | | | | + | + | | | + | + |
| 8 | | | | | | | | | | | | | | | | | + |
| 9 | | | | | | | | | | | | | | | | | + |
| 10 | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | + | | |
| 12 | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | + | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | |
| 16 | | + | | | | | | + | + | | | | | | | | |

Table 6.1: A sample summary database for a document.

[20]. We ran them on the 40 documents to generate 10% and 20% summaries for each document.

A total of 16 summaries were produced for each document. Table 6.1 shows a sample summary database for a document. The documents were selected from the TREC collection [5]. They are news articles on computers, terrorism, hypnosis and nuclear treaties. The average length of the articles is 30 sentences. Human subjects are graduate students in the Department of Computer Science at Columbia University, Cornell University, and Beer-Sheva University in Israel.

6.2 Results and Analysis

6.2.1 Agreement Among Human Subjects

We measured agreement among human subjects using *percent agreement*, a metric defined by [32] for the sense disambiguation task, but also used in other applications such as discourse segmentation [11, 8]. Percent agreement is the ratio of observed agreements with the majority opinion to possible agreements with the majority opinion. For our experiments, agreement among 3 or more subjects is a majority opinion. The total possible agreements with the majority opinion is the number of human subjects times the number of sentences in a document.

Observed agreement equals to the number of times that a subject’s decision agrees with the majority opinion, including both the decision to extract the sentence and not to extract the sentence. The results are shown in Table 6.2.

| Length | Avg. Agreement | Max | Min |
|--------|----------------|------|-----|
| 10% | 96% | 100% | 87% |
| 20% | 90% | 100% | 83% |

Table 6.2: Agreement among 5 human subjects for 40 documents.

We can draw two conclusions from the above results: (1) when human subjects are required to summarize an article within the *same* short length, they are quite consistent with what should be included. This is indicated by the high percentage of agreement for the 10% length summaries. (2) The degree of agreement among human subjects tends to decrease as the length of summary increases. This is shown by lower percent agreement among *same* human subjects when summary length increases from 10% to 20% in the experiment. Note this statement is not valid when the length of summary is out of certain range. For example, the percent agreement apparently increases as the length increases from 90% to 100%.

The above observation agrees with those patterns noticed by other researchers [3]: human subjects are quite consistent with respect to what they perceive as being the most important but less consistent with respect to what they perceive as being less important.

The percent agreement in our experiment is surprisingly high compared to results presented by other researchers. [20] found percent agreement of 13 judges over 5 texts from Scientific America is 71%. [12] found that extracts selected by four different human judges had only 25% overlap. [30] found that the most important 20% paragraphs extracted by 2 subjects have only 46% overlap. Two main reasons for this high percent agreement are the style of TREC articles and our restrictions on uniform length. The 40 documents used in the experiment have very similar text structure: an introduction followed by details, examples, facts, then a secondary point, so on and so forth.

| | Microsoft | | Lexical Chain | | Discourse Structure | |
|-----|-----------|--------|---------------|--------|---------------------|--------|
| | Prec | Recall | Prec | Recall | Prec | Recall |
| 10% | 33 | 37 | 61 | 67 | 46 | 64 |
| 20% | 32 | 39 | 47 | 64 | 36 | 55 |

Table 6.3: Evaluation of summarization programs.

The other reason is our restriction on uniform summary length. This eliminates differences due to perception of optimal summary length; in most other experiments, researchers did not require human judges to create a summary of a given length.

6.2.2 Statistical Significance

Using the same methodology in [11, 8, 20], we applied Cochran’s test to our data. For our application, Cochran’s test evaluates the null hypothesis that the total number of human subjects extracting the same sentence is randomly distributed. Cochran’s statistic Q approximates the χ^2 distribution with $j - 1$ degrees of freedom, where j is the number of elements in the data set, for our application, the number of sentences in a document. Our results show that the agreement among subjects is highly significant. That is, the probability that human subjects extract the same sentence is much higher than would be expected by chance. For all 40 documents, the probability is very low: $p < 10^{-6}$.

6.2.3 Systems Comparison

The “ideal” summary was constructed by taking the majority opinion of five human summaries at the same length, the precision and recall were used as similarity measures. The results are shown in Table 6.3.

According to our data, our lexical chain summarizer is closer to the human constructed summaries.

Chapter 7

Conceptual Maps

In this chapter we discuss the subject of “conceptual maps” — representation of a source text using links among lexical chains. First, we introduce the work of Hasan — “Coherence and Cohesive Harmony”, as a theoretical basis for building conceptual maps. Then, we present our algorithm for computing conceptual maps.

7.1 Chain Interaction and “Cohesive Harmony”

Up to this point, we discussed the contribution of lexical chains to text cohesion, but the issue of chain interaction in the text was ignored. This subject was introduced by Hasan [28] in the context of her research of correlation between coherence and “cohesive harmony”. The goal of this research was to explore whether cohesion can provide a reliable measurement of coherence.

The first assumption was that the degree of coherence correlated with the numerical or the categorical variation of cohesive ties. This hypothesis was tested by comparing the cohesiveness measure of the text with the number of cohesive ties in it.

The test material was children’s stories with different levels of coherence A10, A9 and A13 (these are in Tables 7.1 to 7.3). The judges ranked A10 as the most coherent of the three and A13 at the least.

The ratio between *peripheral tokens* (PT) — tokens which are not

| |
|---|
| <p>there was once a little girl and a little boy and a dog and the sailor was their daddy and the little doggy was white and they like their doggy and they stroke it and they fed it and he run away and then (um the little dog) daddy (um) had to go on a ship and the children missed'em and they began to cry</p> |
|---|

Table 7.1: A10

| |
|--|
| <p>there was a girl and a boy there was a dog and a sailor the dog was a furry dog and the girl and the dog were sitting down and the sailor was standing up and the teddy-bear was lying down asleep and the sailor was looking at (the dog) bear the little girl was laying down too she wasn't asleep and the boy was sitting up he was looking at the bear too</p> |
|--|

Table 7.2: A9

| |
|---|
| <p>once upon a time (there was two little) there was a little girl and a boy and they went aboard a ship and the sailor said to them to go and find a carriage don't go on the ship here because I'm trying to dive but the dog came along and threw himself into the sea and then he came back and (all) they all went home and had a party and they lived happily ever after</p> |
|---|

Table 7.3: A13

| | PT | RT | TT |
|-----|----|----|----|
| A10 | 3 | 40 | 43 |
| A9 | 2 | 30 | 32 |
| A13 | 12 | 34 | 46 |

Table 7.4: Measures of cohesive ties in the texts.

subsumed in chains and *relevant tokens* (RT) — tokens subsumed in chains, was chosen as a possible measure of the text coherence. Table 7.4 shows no significant correlation of this measure to the degree of coherence perceived by readers in these texts. Other proposed metrics, such as the number of ties per clause and the length of the chains, do not reflect the difference in coherence between the texts as well. The final assumption was that chain interaction have to be considered.

A chain is constructed based on a semantic principle which creates unity amongst its members. Each chain represents a “relatively self-contained center of unity.” When these centers of unity are brought together through chain interaction, “cohesive harmony” is achieved. Hasan emphasizes the dominant role of chain interaction as a source of unity in the text.

According to Hasan, chain interaction occurs only if two or more members of a chain stand in the same functional relation with two or more members of the other chain. Not all chain members participate in chain interaction. Hasan calls the chain members which participate in chain interaction as *central tokens* (CT). The hypothesis is that the CTs of a text are directly relevant to the coherent development of the topic in the text. Hasan expresses the degree of cohesive harmony as the ratio between total texts tokens to central tokens in the text. Another measure of coherence that she proposes is the ratio of peripheral tokens to central tokens, so that the higher the ratio of CT to PT, the more coherent the text would be. Results presented in Table 7.5 support these predictions.

By analyzing an 80 texts data, Hasan found that any ranking by the measure of cohesive harmony — percentage of the CT over TT — “consistently correlated with reader’s judgment on how the texts ranked

| | PT | RT | TT | CT | CT/TT% | CT/PT |
|-----|----|----|----|----|--------|-------|
| A10 | 3 | 40 | 43 | 31 | 72.09% | 10 |
| A9 | 2 | 30 | 32 | 11 | 34.37% | 5 |
| A13 | 12 | 34 | 46 | 14 | 30.43% | 1 |

Table 7.5: Measures of cohesive ties and chain interaction the texts.

vis a vis each other in coherence.” An additional conclusion from the data is that in texts, which were “deemed unquestionably coherent, the CTs consistently formed above 50% of the TTs.”

Summary

To summarize, the theory of Hasan

1. implies importance of chain interaction for text cohesiveness;
2. chains interaction is a frequent phenomena in cohesive texts, so it can be computed.

7.2 Building Conceptual Maps

7.2.1 Definition of a Conceptual Map

The importance of understanding of concept interaction in the text was also noted by researchers in reading comprehension. Conceptual maps are widely used as helping tools for pupils in understanding texts in biology and history classes. A conceptual map is the representation of text content that makes explicit the relations between important concepts of the text. A map is a graph, in which vertices represent concepts, and the edge labels indicates term relations. An example of such a map is in Figure 7.1.

The calculation of such maps is a challenging task. The assumption that conceptual maps are computable is based on the conclusion from Hasan’s research that if a text is cohesive, there is enough surface evidence for the identification of chain interaction. It also follows the

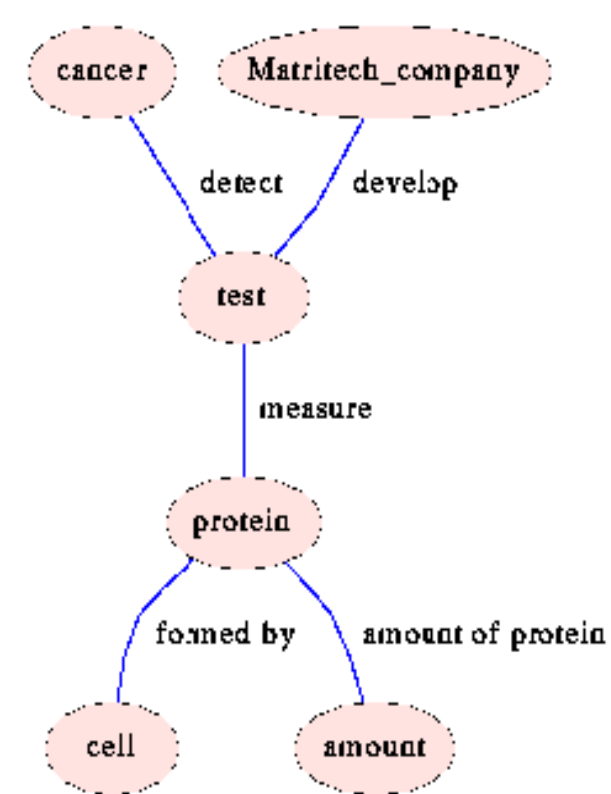


Figure 7.1: Example of a conceptual map.

intuition that, if two concepts, not related to one another according to common knowledge, are related in the text, then an explicit relation between them must exist in the text.

7.2.2 Algorithm for Conceptual Map Construction

We propose the following algorithm to build conceptual maps once lexical chains have been identified:

1. Calculate chains and select important chains.
2. For every two chains find all units in which words of these two chains co-occur.
3. For every chain, choose a word that represents the concept. (We currently select the most frequent one.)
4. Put the edges between two concepts that interact — have at least two co-occurrences, according to Hasan.

The crucial point of this algorithm is the second step. Currently we use as a unit element the whole sentence, and this increase the number of related concepts, especially if a text consists of multi-clause sentences. Partial solution to this problem is to divide complex sentences to clauses, and use clause as a unit element in our second step. Currently, there is no robust tool available for us for clause division. Precise recognition of chain interaction requires information about the syntactic roles of words in the sentence, this requires using a parser. If a parser is available, the majority of chain interactions can be identified from the parser's output by regular expressions.

Examples:

| | | |
|-------------------------|---|------------------------------|
| $\text{verb}(w_1, w_2)$ | w_1, w_2 are arguments of the same verb | <i>“Michael created FUF”</i> |
| $\text{prop}(w_1, w_2)$ | w_1, w_2 are members of the same propositional phrase | <i>“Owen from CogenTex”</i> |

A more problematic case for identification are the cases where words do not interact directly, but their groups do interact. For example, in the sentence *“a machine to control the rate of drug was invented by a*

group of researchers from HCI”, the groups “*a machine to control the rate of drug*” and “*a group of researchers from HCI*” are related as arguments of the verb “invent”. Semantically, the concepts “*machine*” and “*researchers*” interact in this sentence (“the researchers invented the machine”), but syntactically, they do not have any direct interaction. A quick solution can be to consider indirect interaction as a case of chain interaction. However, according to this decision, the concepts “rate” and “HCI” will be considered related, while they are not. An accurate list of syntactic relations which imply chain interaction is a subject for future research.

To conclude, our computational model make it possible to build conceptual maps, but the use of a parser and specification of syntactic relations of chain interaction will significantly increase performance.

Chapter 8

Contributions, Limitations and Future Work

8.1 Contributions

8.1.1 The Chaining Algorithm

- We implemented a new algorithm for computing chains — a dynamic algorithm based on text segmentation.
- We presented a generic computational model for chain computation, which generalizes various existing lexical chain algorithms.
- An intrinsic evaluation is performed by identifying how successful the lexical chainer is in disambiguating nouns in the context. We verified that our value settings for system parameters were optimal.
- The influence of Thesaurus quality and text properties were found to be crucial for the chaining algorithm.

8.1.2 Summarization

- We presented a new summarization algorithm, based on a model of the topic progression in the text derived from lexical chains.

- An extrinsic evaluation was performed, that was used to determine to the extent to which the sentences extracted by the summarizer match those that human judges would extract. We compared our summarizer with two other available summarization systems and found our approach to give results significantly closer to human judges (using the majority rule) than the other systems.
- We presented an alternative method of source text representation based on interaction of main text concepts — “conceptual maps”.

8.2 Limitations & Future Work

8.2.1 The Summarization System

We have identified the following main problems with our summarization method:

- Sentence granularity — all of our methods extract whole sentences as single units. This has several drawbacks: long sentences have significantly higher likelihood to be selected, they also include many constituents which would not have been selected on their own merit. The alternative is extremely costly, it involves parsing of the sentences, the extraction of only the central constituents from the source text and the regeneration of a summary text using text generation techniques.
- Extracted sentences contain anaphora links to the rest of the text. This has been investigated and observed by [2]. Several heuristics have been proposed in the literature to address this problem [26], [27] and [2]. The strongest seems to be to include together with the extracted sentence the one immediately preceding it. Unfortunately, when we select the first sentence in a segment, the preceding sentence does not belong to the paragraph and its insertion has a detrimental effect on the overall coherence of the summary. A preferable solution would be to replace anaphora with their referent, but again this is an extremely costly solution.

- Our method does not provide any way to control the length and level of detail of the summary. In all of the methods, we extract one sentence for each chain. The number of strong chains remains small (around 5 or 6 for the texts we have tested, regardless of their length), and the remaining chains would introduce too much noise to be of interest in adding details. The best solution seems to be to extract more material for the strongest chains.

Our method is obviously partial in that it only considers lexical chains as a source representation, and ignores any other clues that could be gathered from the text. Combination of lexical chains with discourse structure and coreference relations can significantly improve the performance.

8.2.2 **More Future Work**

- Modify the system to take advantage of the new WordNet database.
- Embed a generation tool in order to improve the output quality.
- Use a parser for improving the quality of the generated conceptual maps.
- The lexical chains that we generate can be used to create automatic concept maps for texts. These can be used in many ways to create new text-intelligent tools.

Appendix A

Bayesian Networks Text

A.1 The Raw Text

When Microsoft Senior Vice President Steve Ballmer first heard his company was planning to make a huge investment in an Internet service offering movie reviews and local entertainment information in major cities across the nation, he went to Chairman Bill Gates with his concerns.

After all, Ballmer has billions of dollars of his own money in Microsoft stock, and entertainment isn't exactly the company's strong point.

But Gates dismissed such reservations. Microsoft's competitive advantage, he responded, was its expertise in Bayesian networks.

Asked recently when computers would finally begin to understand human speech, Gates began discussing the critical role of "Bayesian" systems.

Ask any other software executive about anything Bayesian and you're liable to get a blank stare.

Is Gates onto something? Is this alien-sounding technology Microsoft's new secret weapon?

Bayesian networks are complex diagrams that organize the body of knowledge in any given area by mapping out cause-and-effect relationships among key variables and encoding them with numbers that represent the extent to which one variable is likely to affect another.

Programmed into computers, these systems can automatically generate optimal predictions or decisions even when key pieces of information are missing.

When Microsoft in 1993 hired Eric Horvitz, David Heckerman and Jack Breese, pioneers in the development of Bayesian systems, colleagues in the field were surprised. The field was still an obscure, largely academic enterprise.

Today the field is still obscure. But scratch the surface of a range of new Microsoft products and you're likely to find Bayesian networks embedded in the software. And Bayesian nets are being built into models that are used to predict oil and stock prices,

control the space shuttle and diagnose disease.

Artificial intelligence (AI) experts, who saw their field discredited in the early 1980s after promising a wave of "thinking" computers that they ultimately couldn't produce, believe widening acceptance of the Bayesian approach could herald a renaissance in the field.

Bayesian nets provide "an overarching graphical framework" that brings together diverse elements of AI and increases the range of its likely application to the real world, says Michael Jordon, professor of brain and cognitive science at the Massachusetts Institute of Technology.

Microsoft is unquestionably the most aggressive in exploiting the new approach. The company offers a free Web service that helps customers diagnose printing problems with their computers and recommends the quickest way to resolve them. Another Web service helps parents diagnose their children's health problems.

The latest version of Microsoft Office software uses the technology to offer a user help based on past experience, how the mouse is being moved and what task is being done. "If his actions show he is distracted, he is likely to need help," Horvitz says. "If he's been working on a chart, chances are he needs help formatting the chart".

"Gates likes to talk about how computers are now deaf, dumb, blind and clueless. The Bayesian stuff helps deal with the clueless part," says Daniel T. Ling, director of Microsoft's research division and a former IBM scientist.

Horvitz and his two Microsoft colleagues, who were then classmates at Stanford University, began building Bayesian networks to help diagnose the condition of patients without turning to surgery.

The approach was efficient, says Horvitz, because you could combine historical data, which had been meticulously gathered, with the less precise but more intuitive knowledge of experts on how things work to get the optimal answer given the information available at a given time.

Horvitz, who with two colleagues founded Knowledge Industries to develop tools for developing Bayesian systems, says he and the others left the company to join Microsoft in part because they wanted to see their theoretical work more broadly applied.

Although the company did important work for the National Aeronautics and Space Administration and on medical diagnostics, Horvitz says, "It's not like your grandmother will use it".

Microsoft's activities in the field are now helping to build a groundswell of support for Bayesian ideas.

People look up to Microsoft," says Pearl, who wrote one of the key early texts on Bayesian networks in 1988 and has become an unofficial spokesman for the field. "They've given a boost to the whole area".

Microsoft is working on techniques that will enable the Bayesian networks to "learn" or update themselves automatically based on new knowledge, a task that is currently cumbersome.

The company is also working on using Bayesian techniques to improve upon popular AI approaches such as "data mining" and "collaborative filtering" that help draw out

A.2. BAYESIAN NETWORK TEXT: THE STRONGEST CHAIN⁸¹

relevant pieces of information from massive databases. The latter will be used by Microsoft in its new online entertainment service to help people identify the kind of restaurants or entertainment they are most likely to enjoy.

A.2 Bayesian Network Text: the Strongest Chain

The Criterion is 3.58, here are the five strong chains:

CHAIN 1: Score = 14.0

microsoft: 10 concern: 1 company: 6
entertainment-service: 1 enterprise: 1
massachusetts-institute: 1

CHAIN 2: Score = 9.0

bayesian-system: 2 system: 2 bayesian-net: 2
network: 1 bayesian-network: 5 weapon: 1

CHAIN 3: Score = 7.0

ai: 2 artificial-intelligence: 1
field: 7 technology: 1 science: 1

CHAIN 4: Score = 6.0

technique: 1 bayesian-technique: 1 condition: 1
datum: 2 model: 1 information: 3 area: 1
knowledge: 3

CHAIN 5: Score = 3.0

computer: 4

Bibliography

- [1] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *ACL/EACL-97 summarization workshop*, pages 10–18, Madrid, 1997.
- [2] William J. Black. Parsing, linguistic resources and semantic analysis, for abstracting and categorization. 1994.
- [3] Johnson Ronald E. Recall of prose as a function of structural importance of linguistic units. *Journal of Verbal Learning and Verbal Behaviour*, 9:12–20, 1970.
- [4] H. P. Edmunson. New methods in automatic abstracting. *Journal of the ACM*, 16(2):264–285, 1969.
- [5] Donna Harman. Trec. In *An Overview of The Third Text Retrieval Conference*, Gaithesburg, MD, 1994. National Institute of Standards and Technology.
- [6] Michael Hasan and Ruqaiya Halliday. *Cohesion in English*. Longman, London, 1976.
- [7] Michael Hasan and Ruqaiya Halliday. *An Introduction to Functional Grammar*. Edward Arnold, London, 1985.
- [8] M. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pages 9–16, Las Cruces, New Mexico, 1994.

- [9] Graeme Hirst and David St-Onge. Lexical chains as representation of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database and some of its applications*. Cambridge, MA: The MIT Press, 1997 [to appear].
- [10] M. Hoey. *Patterns of Lexis in Text*. Oxford University Press, Oxford, 1991.
- [11] Passonneau Rebecca J. and Diane J. Litman. Intention-based segmentation: human reliability and correlation with linguistic cues. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics (ACL-93)*, pages 148–155, Ohio, 1993.
- [12] Rath G. J., Resnick A., and Savage R. The formation of abstracts by the selection of sentences: Part 1: sentence selection by man and machines. *American Documentation*, 12(2):139–141, 1961.
- [13] Hobbs Jerry. Coherence and coreference. Technical Report Technical note 168, SRI International, 1978.
- [14] Hong-Yan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Summarization evaluation methods: Experiments and analysis. In *Proceedings of AAAI-98 Symposium*, 1998 [to appear].
- [15] Karen Sparck Jones. What might be in summary? *Information Retrieval*, 1993.
- [16] Karen Sparck Jones and J. R. Galliers. *Evaluating natural language processing systems: an analysis and review*. New York: Springer, 1996.
- [17] Julian Kupiec, Jan Pederson, and Francine Chen. A trainable document summarizer. In *SIGIR '95*, pages 68–73, Seattle, Washington, 1995.
- [18] H. P. Luhn. The automatic creation of literature abstracts. In Schultz, editor, *H. P. Luhn: Pioneer of Information Science*. Spartan, 1968.

- [19] W. C. Mann and S. Thompson. Rhetorical structure theory: description and constructions of text structures. In Gerard Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pages 85–96. Martinus Nijhoff Publishers, 1987.
- [20] Daniel Marcu. From discourse structures to text summaries. In *ACL/EACL-97 summarization workshop*, pages 82–88, Madrid, 1997.
- [21] Kathleen McKeown and Dragomir Radev. Generating summaries of multiple news articles. In *SIGIR 95 Proceedings*, 1995.
- [22] Seiji Miike, Etsuo Itoh, Kenji Ono, and Kazuo Sumita. A full-text retrieval system with a dynamic abstract generation function. In *SIGIR '94*, pages 152–161, Seattle, Washington, 1994.
- [23] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312, 1990.
- [24] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of the text. *Computational Linguistics*, 17(1):pp. 21–45, 1991.
- [25] Kenji Ono, Kazuo Sumita, and Seiji Miike. Abstract generation based on rhetorical structure extraction. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, volume 1, pages 344–384, Kyoto, Japan, 1994.
- [26] C. D. Paice. Constructing literature abstracts by computer: techniques and prospects. *Information Processing and Management*, 26(1):171–186, 1990.
- [27] C. D Paice and G. D. Husk. Towards the automatic recognition of anaphoric features in english text: The impersonal pronoun “it”. *Computer Speech and Language*, 2:pp. 109–132, 1991.

- [28] Hasan R. *Reading Comprehension*, chapter Coherence and Cohesive Harmony. 1984.
- [29] P Roget. *Roget's International Thesaurus*. Harper and Row Publishers Inc., 4th edition, 1977.
- [30] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–208, 1997.
- [31] Mark A. Stairmand. *A Computational Analysis of Lexical Cohesion with Applications in Information Retrieval*. PhD thesis, Center for Computational Linguistics, UMIST, Manchester, 1996.
- [32] Gale William, Kenneth W. Church, and David Yarowsky. Estimating upper and lower bounds on the performance of word-sense disambiguation programs. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pages 249–256, 1992.