UNIVERSITY OF BEIRA INTERIOR
Department of Computer Science

ALEXIA - Acquisition of Lexical Chains for Text Summarization

Cláudia Sofia Oliveira Santos

A Thesis submitted to the University of Beira Interior to require the Degree of Master of Computer Science Engineering

Supervisor: Prof. Gaël Harry Dias
University of Beira Interior

Covilhã, Portugal
February 2006

Acknowledgments

There are many people, without whose support and contributions this thesis would not have been possible. I am specially grateful to my supervisor, Gaël Dias for his constant support and help, for professional and personal support, and for getting me interested in the subject of Natural Language Processing and for introducing me to the summarization field. I would like to express my sincere thanks to Guillaume Cleuziou, for all his help and suggestions about how to use his clustering algorithm, for making available its code and always being there at an e-mail distance.

I would like to thank all my friends, but in a special way to Ricardo, Suse, Eduardo and Rui Paulo for all their help and support when sometimes I thought I would not make it; for being really good friends.

I would also like to thank the HULTIG research group and the members of the LIFO of the University of Orléans (France) in particular Sylvie Billot and one more time to Guillaume who helped me feeling less lonely in France and helped me in the beginning of this work.

Finally, I am eternally indebted to my family. I would specially like to thank my parents and my sister, Joana, who were always there for me and always gave me support and encouragement all through my course, and always expressed their love. At last, but not least, I would like to sincerely thank my husband, Marco, who never stopped believing that I could do it, always encouraged me to go on and loved me unconditionally. I am definitely sure that I could not have made all this without their stimulus, their support and specially their love. They are responsible for this thesis in more ways than they know and more than I can describe. This thesis is dedicated to them.

Thank you all!
Cláudia

Abstract

Text summarization is one of the more complex challenges of Natural Language Processing (NLP). It is the process of distilling the most important information from a source to produce an extract or abstract with a very specific purpose: to give the reader an exact and concise knowledge of the contents of the original source document.

It is generally agreed that automating the summarization process should be based on text understanding that mimics the cognitive process of humans. It may take some time to reach a level where machines can fully understand documents. In the interim we must use other properties of text, such as lexical cohesion analysis, that do not rely on full comprehension of the text. An important idea related to cohesion is the notion of Lexical Chain. Lexical Chains are defined as sequences of semantically related words spread over the entire text.

A Lexical Chain is a chain of words in which the criterion for inclusion of a word is some kind of cohesive relationship to a word that is already in the chain (Morris and Hirst 1991). Morris and Hirst proposed a specification of cohesive relations based on the Roget's Thesaurus. Hirst and St-Onge (1997), Barzilay and Elhadad (1997), Silber and McCoy (2002), Galley and McKeown (2003) construct Lexical Chains based on WordNet relations (Miller, 1995). In this thesis, we propose a new method to build Lexical Chains using a lexico-semantic knowledge base, automatically constructed, as resource. We propose a method to group the words of the texts of a document collection into meaningful clusters to extract the various themes of the document collection. The clusters are then hierarchically grouped given rise to a binary tree structure within which words may belong to different clusters.

Our lexico-semantic knowledge base is a resource in which words with similar meanings are grouped together. In order to determine if two concepts are semantically related we use our knowledge base and a measure of semantic similarity based on the ratio between the information content of the most informative common ancestor and the information content of both concepts (previously proposed by Lin (1998)). We use this semantic

similarity measure to define a relatedness criterium in order to assign a given word to a given chain in the lexical chaining process. This assignment of a word to a chain is a compromise between the approaches of Hirst and St-Onge (1997) and Barzilay and Elhadad (1997).

The main steps of the construction of Lexical Chains are as follows: the construction of Lexical Chains begins with the first word of a text. To insert the next candidate word, its relations with members of existing Lexical Chains are checked. If there are such relations with all the elements of a chain then the new candidate word is inserted in the chain. Otherwise a new lexical chain can be started. This procedure includes disambiguation of words, although we allow a given word to have different meanings in the same text (i.e. belonging to different Lexical Chains).

Our experimental evaluation shows that meaningful Lexical Chains can be constructed with our lexical chaining algorithm. We found that the text properties are not crucial as our algorithm adapts itself to different genres/domains.

Resumo Alargado

Com o aumento desmedido da *World Wide Web* e da quantidade de serviços de informação e disponibilização, deparamo-nos actualmente com uma acumulação excessiva de textos de diversas naturezas que se traduz na incapacidade de os consultar na sua íntegra. Os motores de busca fornecem meios de acesso a enormes quantidades de informação, listando os documentos considerados relevantes para as necessidades do utilizador. Uma simples procura pode retornar milhões de documentos como "potencialmente relevantes", que o utilizador terá que consultar de forma a efectivamente julgar a sua relevância. Esta explosão de informação resultou num problema bem conhecido de sobrecarga de informação. A tecnologia da sumarização automática de textos torna-se indispensável para lidar com este problema.

Com o reconhecimento desta necessidade, a investigação e o desenvolvimento de sistemas que sumarizem automaticamente textos transformaram-se consideravelmente num foco de interesse e de investimento na investigação pública e privada.

A Sumarização é uma das áreas mais bem sucedidas do Processamento de Linguagem Natural (PLN) e de acordo com (Mani, 2001) o objectivo da sumarização automática é *"considerando um texto fonte, extrai-se dele o conteúdo mais relevante, o qual é apresentado de uma forma condensada de acordo com as necessidades do utilizador ou da aplicação"*. Poderemos obter sumários diferentes gerados a partir da mesma fonte dependendo da funcionalidade ou uso.

Concorda-se, em geral, que automatizar o procedimento de sumarização deve ser baseado na compreensão do texto imitando os processos cognitivos dos seres humanos. Poderá demorar algum tempo até se conseguir alcançar um nível onde as máquinas possam compreender os documentos na sua totalidade. Provisoriamente, devemos tomar partido de outras propriedades do texto, tais como a análise da coesão lexical, que não depende da total compreensão do texto. A coesão lexical fornece um bom indicador da estrutura do discurso de um documento, coesão essa que é usada pelos sumarizadores humanos de forma a identificar as parcelas do texto mais relevantes.

Mais especificamente, a coesão lexical é um método para identificar parcelas do texto ligadas entre si, tendo como base vocabulário semanticamente relacionado. Um método para descobrir estes relacionamentos entre palavras consiste na utilização de uma técnica linguística designada por encadeamento lexical, onde as Cadeias Lexicais são definidas como sequências de palavras semanticamente relacionadas difundidas por todo o texto.

Desde que as Cadeias Lexicais foram previamente propostas por Morris e Hirst (1991), que estas têm sido usadas com sucesso como uma ferramenta eficaz para a Sumarização Automática de textos. A Sumarização de textos baseia-se no problema de seleccionar as parcelas mais importantes do texto e no problema de gerar sumários coerentes. Todas as avaliações experimentais mostram que os sumários podem ser construídos com base num uso eficiente de Cadeias Lexicais.

Uma Cadeia Lexical é uma cadeia de palavras em que o critério de inclusão de uma palavra é uma espécie de relacionamento coesivo com uma palavra que já se encontre nessa cadeia (Morris e Hirst, 1991). Morris e Hirst propuseram uma especificação das relações coesivas baseadas no thesaurus Roget. Hirst e St-Onge (1997), Barzilay e Elhadad (1997), Silber e McCoy (2002), Galley e McKeown (2003), constroem Cadeias Lexicais a partir de relações do WordNet (Miller, 1995). Nesta tese, em vez de se usar o recurso linguístico padrão WordNet, o nosso algoritmo identifica relacionamentos lexicais coesivos entre palavras baseando-se na evidência do *corpus*, utilizando uma base de conhecimento lexico-semântica automaticamente construída. Ao contrário das aproximações que usam o WordNet como fonte de conhecimento, examinamos também as relações lexicais coesivas que não podem ser definidas nos termos de relações de thesaurus, mas são consideradas "intuitivamente" relacionadas devido à sua co-ocorrência no texto.

De forma a seleccionar as palavras candidatas ao processo de encadeamento, etiquetamos as palavras com o TnT *tagger* (Brants, 2000) e extraimos os nomes compostos utilizando o software SENTA (Dias, 2002). Por fim os nomes, os nomes compostos e os nomes próprios são filtrados e considerados como palavras candidatas ao processamento das Cadeias Lexicais.

Cada palavra candidata poderá ter múltiplos significados possíveis que poderão somente ser determinados considerando o contexto em que cada uma delas ocorre. Para este

propósito, usamos o algoritmo de *overlapping-clustering* PoBOC (Cleuziou *et al*., 2004) de forma a construir uma estrutura em árvore binária que agrupe as palavras mais similares de modo a que os diferentes sentidos de uma dada palavra possam ser identificados em diferentes grupos. O algoritmo PoBOC tem como entrada uma matriz de similaridade calculada com uma nova medida de similaridade informativa proposta por (Dias e Alves, 2005) que tem em consideração as co-ocorrências das palavras. O resultado deste algoritmo é uma hierarquia de conceitos na qual um objecto pode pertencer a diversos grupos, produzindo um conjunto de grupos sobrepostos. Desta maneira, as palavras que são agrupadas na mesma classe representam um sentido ou um conceito particular da palavra. O número final de grupos é a *priori* desconhecido.

A nossa base de conhecimento lexico-semântica é um recurso no qual as palavras com significados semelhantes são agrupadas juntas. De forma a determinar se dois conceitos são semanticamente relacionados usamos a nossa base de conhecimento e uma medida de similaridade semântica baseada na razão entre o conteúdo informativo do nó superior comum aos dois conceitos e o conteúdo informativo de ambos os conceitos (previamente proposta por Lin (1998)). Utilizamos esta medida de similaridade semântica de forma a definir um critério de relacionamento a fim de atribuir uma dada palavra a uma dada cadeia no processo de encadeamento lexical. Esta atribuição de uma palavra a uma cadeia é um compromisso entre as metodologias de Hirst e St-Onge (1997) e de Barzilay e Elhadad (1997).

As principais etapas na construção de Cadeias Lexicais são as seguintes: a construção de Cadeias Lexicais inicia-se com a primeira palavra do texto; para introduzir a próxima palavra candidata são verificadas as suas relações com os membros das Cadeias Lexicais já existentes. Se tais relações existirem com todos os elementos de uma cadeia, então a nova palavra candidata é inserida na cadeia. De outra forma, uma nova Cadeia Lexical é iniciada. Este procedimento anula qualquer ambiguidade que uma palavra possa apresentar, embora nós permitamos que uma dada palavra tenha diferentes significados no mesmo texto.

A nossa avaliação experimental mostra que Cadeias Lexicais significativas podem ser construidas com o nosso algoritmo de encadeamento lexical. Observamos ainda que,

as propriedades do texto não são cruciais pois o nosso algoritmo adapta-se a diferentes géneros/domínios.

Contents

List of Figures

Chapter 1

Introduction

With the rapid growth of the World Wide Web and on-line information services, more and more information is available and accessible on-line. Search engines provide a means to access huge volumes of information by retrieving the documents considered relevant to the user's query. A simple query can routinely return millions of documents as "potentially relevant". Even with search engines, the user has to go through the entire document content to judge its relevance. There is no time to read everything, and yet we have to make decisions about what is important or not based on the information available. This explosion of information has resulted in a well recognized information overload problem. The technology of automatic text summarization is indispensable to deal with this problem.

In recognition of this need, the research and development of systems to automatically summarize texts has become the focus of considerable interest and investment in both research and commercial sectors.

Summaries are frequently used in our daily lives to serve variety of purposes. Headlines of news articles, market reports, movie previews, abstracts of journal articles, TV listings, minutes of a meeting, are some of the commonly used forms of summaries. Several advanced tools have been developed in recent times using summarization techniques to meet certain requirements. Newsblaster (McKeown *et al.*, 2003) allows the users to be updated about the interesting events happening around the world, without the need to spend time searching for the related news articles. They group the articles from various

news sources into event-related clusters, and generate a summary for each cluster. The meeting summarizer (Waibel *et al*., 1998) combines speech recognition and summarization techniques to browse the contents of a meeting. Healthdoc (Hirst *et al*., 1997) helps physicians by providing a "recommended treatment", for particular patient's symptoms, from the vast online medical literature. Broadcast news navigator (Maybury and Merlino, 1997) is capable of understanding the news broadcast and present the user with the condensed version of the news. IBM's Re-Mail (Rohall *et al*., 2004) can summarize the threads of e-mail messages based on simple sentence extraction techniques. It is hard to imagine everyday life without some form of summarization.

Summarization is one of the most successful areas of Natural Language Processing (NLP) and according to (Mani, 2001) the goal of automatic summarization is *"to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs"*. Summaries generated are dependent on various factors e.g., different summaries can be generated for the same input source depending on their functionality and usage.

It is generally agreed that automating the summarization procedure should be based on text understanding that mimics the cognitive process of humans. It may take some time to reach a level where machines can fully understand documents. In the interim, we must take advantage of other properties of text, such as lexical cohesion analysis, that do not rely on full comprehension of the text. Lexical cohesion provides a good indicator for the discourse structure of a document, used by human summarizers to skim through the document and identify the salient portions.
More specifically, lexical cohesion is a method to identify connected portions of the text based on the relations between words in the text, indicated by the use of semantically related vocabulary. One method of uncovering these relationships between words is to use a linguistic technique called lexical chaining, where Lexical Chains are defined as sequences of semantically related words spread over the entire text.

Since Lexical Chains were first proposed by Morris and Hirst (1991), they have been successfully used as an effective tool for Automatic Text Summarization. Text summarization addresses both the problem of selecting the most important portions of text and

the problem of generating coherent summaries. All the experimental evaluations show that efficient summaries can be built based on an efficient use of Lexical Chains. These researches showed that Lexical Chains are a strong intermediate representation of a document, and hence should perform well in text summarization application that would benefit from a more meaningful representation of a document than a bag-of-words representation.

Lexical chaining requires identification of the semantic relations to determine the compatibility of a word with respect to a chain. Since lexical cohesion is realized in texts through the use of related vocabulary, knowledge resources such as thesauri, ontologies and dictionaries have been used as a means of identifying related words. Almost all the methods to compute Lexical Chains use WordNet [1] as lexical source to identify the semantic relations between the word senses.

WordNet (Miller, 1995) is an online lexical database which design is inspired by current psycholinguistic theories of human lexical memory. WordNet is divided into 4 distinct word categories: nouns, verbs, adverbs and adjectives. The most important relationship between words in WordNet is synonymy. A unique label called a synset number identifies each synonymous set of words (*a synset*) in WordNet. Each node or synset in the hierarchy represents a single lexical concept and is linked to other nodes in the semantic network by a number of relationships. Senses are listed in decreasing order of frequency, meaning that the most commonly used sense (in a given corpus) occurs first.

Morris and Hirst (1991) described but did not implement an algorithm for computing Lexical Chains. A first computational model of Lexical Chains was introduced by Hirst and St-Onge (1997). This linear time algorithm, however, suffers from inaccurate Word Sense Disambiguation (WSD), since their greedy strategy immediately disambiguates a word as it is first encountered.

The application of lexical chaining in summarization was first implemented by Barzilay and Elhadad (1997). They significantly alleviated the disambiguation problem of the Hirst and St-Onge algorithm at the cost of a worse running time; computational inefficiency is due to the processing of many possible combinations of word senses in the text in order to decide which assignment is the most likely at the end of the process of candidate Lexical

[1] http://cogsci.princeton.edu/ wn

Chains building. They use Lexical Chains to create intermediate representations in order to identify important sentences in a document by retrieving those in which strong chains occur. This method has exponential efficiency and as a consequence, computing Lexical Chains for documents of any reasonable size is a hard problem.
Later, Silber and McCoy (2002) implemented the Barzilay and Elhadad (1997) algorithm in linear time and space for efficient identification of Lexical Chains in large documents. They create a text representation in the form of *meta-chains*, i.e., chains that capture all possible Lexical Chains in the document. After creating the meta-chains, they use a scoring algorithm to identify the Lexical Chains that are most relevant to the document, by eliminating from the meta-chains unnecessary overhead information, and generate the Lexical Chains representing the document.
Galley and McKeown (2003) identified Lexical Chains after performing Word Sense Disambiguation. This new linear time algorithm for lexical chaining adopts the assumption of one sense per discourse. This algorithm can be decomposed into three steps: building a representation of all possible interpretations of the text (similar to meta-chains in the case of Silber and McCoy, 2002), disambiguating all words, and finally building the Lexical Chains.

One common point of all these works is that Lexical Chains are built using WordNet as the standard linguistic resource.

Unfortunately, systems based on linguistic knowledge bases are limited. In particular, domain specific dictionaries and thesauri are difficult to find and are largely obsolete by the time they are available. Certain word usages may be particular to a period of time, which are unlikely to be captured by these resources. Furthermore, dictionaries capture a particular form of lexical knowledge which is often very different from the sort needed to specifically relate words or sentences. They include many usages that are very unfrequent in a particular corpus or genre of documents.

One of the most striking reasons why we do not use WordNet as our knowledge source is that we want to build a system independent of the genre or language. In fact, linguistic resources are not available for the majority of languages so that their application is drastically limited and as a consequence do not apply to less favored languages that may

transform themselves into endangered languages. Moreover, certain categories such as those relating to plants and animals are more developed than others. WordNet is missing a lot of explicit links between intuitively related words. Fellbaum (1998) refers to such obvious omissions in WordNet as the "tennis problem" where nouns such as *nets* and *rackets* and *umpires* are all present in the taxonomy, but WordNet provides no links between these related tennis concepts. Another complaint commonly encountered when using WordNet is that its level of sense granularity is too fine. Polysemy is represented in WordNet as a list of different synset numbers for a particular syntactic form. The basic problem in computing chains using WordNet is the high degree of polysemy of English words, resulting in many possible chains being formed. The last reason for not using WordNet is that most of compound nouns are not present in WordNet, e.g. *quantum computer*.

In order to solve these problems, we propose to automatically construct from a collection of documents a lexico-semantic knowledge base with the intuit of identifying lexical cohesive relationships between words based on corpus evidence. These relationships are then used to build the Lexical Chains of any documents based on the adaptation of Barzilay and Elhadad (1997) and Hirst and St-Onge (1997) algorithms.

Our lexico-semantic knowledge base is a resource in which words with similar meanings are grouped together. For that purpose we use the overlapping clustering algorithm PoBOC (Cleuziou *et al*, 2004) to build a tree structure among the most similar words so that different senses of a given word can be identified with different clusters. The algorithm PoBOC takes as input a similarity matrix calculated with a new informative similarity measure proposed by (Dias and Alves, 2005) that takes into account word co-occurrences. The final output is a hierarchy of concepts in which an object may belong to several clusters, i.e. a set of overlapping clusters. The final number of clusters is *a priori* unknown.

In order to determine if two concepts are semantically related we use our knowledge base and a measure of semantic similarity proposed by Lin (1998). This measure is based on the ratio between the information content of the most informative common ancestor and the information content of both concepts.

We use this semantic similarity measure to define a relatedness criterium in order to assign

a given word to a given chain in the lexical chaining process.

Unlike the approaches which use WordNet as a knowledge source, we also examine lexical cohesive relationships that cannot be defined in terms of thesaural relationships, but are considered "intuitively" related due to their regular co-occurrence in text.

As a summary, our approach can be applied on any input text thus avoiding the problems of genre/language-dependent systems that need to be tuned each time one of these parameters changes (genre or language).

The architecture of our lexical chainer is shown in Figure 1.1., where each of its components will be presented in this thesis.

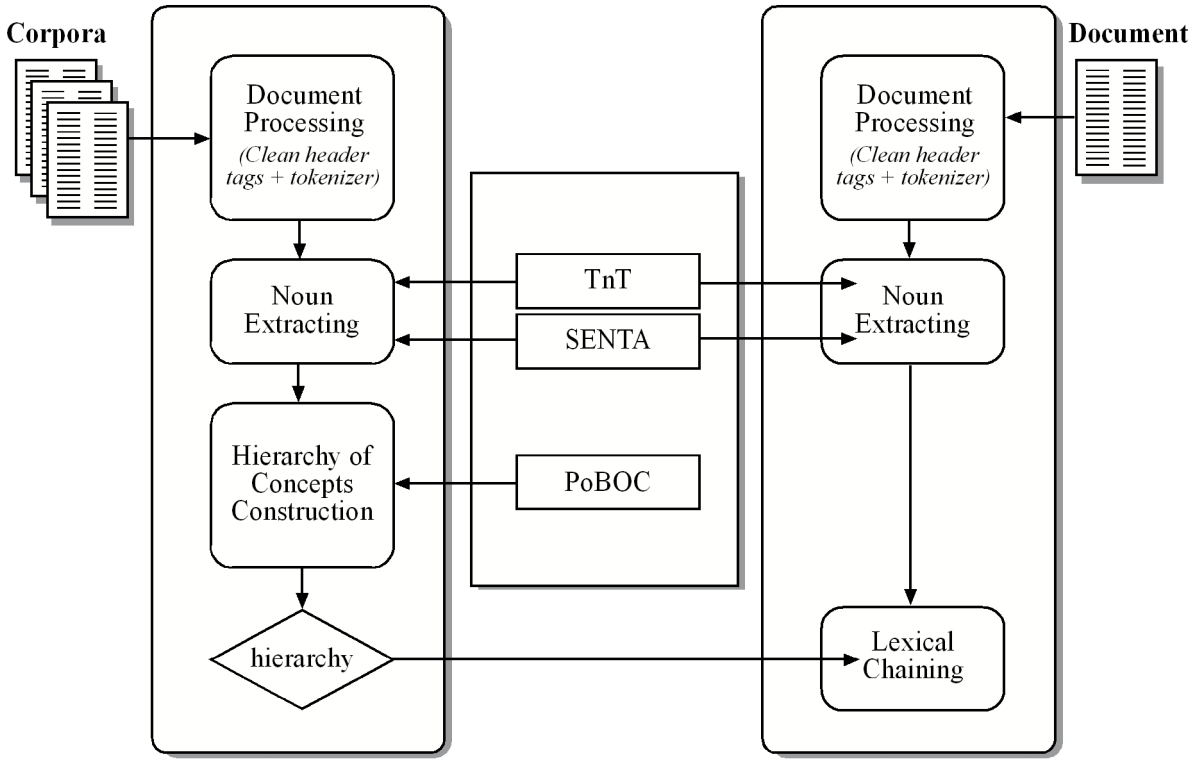

Figure 1.1: Architecture of the lexical chainer system.

1.1 Chapter Outline

The thesis consists of six chapters, which description is given below.

We first presented background material on summarization and Lexical Chains.

Chapter 2 provides an introduction to Summarization and its basic notions and presents the related work in the field, with special attention to the approaches that use Lexical Chains. We then compare our approach to existing ones.

Chapter 3 describes in detail the construction process of the hierarchical lexico-semantic knowledge base.

Chapter 4 presents our algorithm for lexical chaining based on an adaptation of the Lin measure (Lin, 1998).

Chapter 5 presents some qualitative and quantitative results for evaluation of our lexical chaining algorithm.

Finally, Chapter 6 presents some concluding remarks and future work.

Chapter 2

Automatic Text Summarization and Lexical Chaining

Text summarization is one of the most complex challenges of Natural Language Processing. It is the process of distilling the most important information from a source to produce an extract or abstract with a very specific purpose: to give the reader an exact and concise knowledge of the contents of the original source document.

2.1 Basic notions of summarization

2.1.1 Audience

Summaries of texts can be divided into different categories, some of them harder to automate than others. One division is based on the origin of the text in the summary, extractive or abstractive. An *extract* is a summary that consists of sentences copied entirely from the source. Thus a typical extract at a condensation rate of 25% will take some 25% of the material in the document. We can conceive this property as 25% of the words in the document, or 25% of the sentences in the document, or 25% of the paragraphs.
In contrast, an *abstract* is a summary where at least some new text is generated by the summarizer and is not present in the source.

2.1.2 Relation to source

Summaries can also be categorized by their purpose. The intended function of the summary is usually taken into consideration. They can be *indicative*. These summaries are meant to give the reader an idea as to whether it would be worthwhile reading the entire document or not. The topic and scope of the text should be expressed without much of an explanation. They can also be *informative*. This type of summary explains a certain concept to the maximum possible detail at a given compression rate. These types of summaries are often contrasted with *evaluative* or *critical* summaries, which criticize the document. For example, they can express an opinion on the methods employed and the validity of the results, in the case of a scientific paper for example.

2.1.3 Function

Another distinction between summaries can be made based on the intended audience. Traditionally, *generic summaries* are intended to be read by a broad readership community and contain the information considered salient for the author's viewpoint. *User-focused summaries* are generated to be read by a specific group of people having interests in a specific topic, query or concepts. These summaries include information relevant to the user's interests irrespective of its salience in the document. A user-focused summarizer usually includes parameters to influence this weighting.

2.1.4 Compression rate

An important parameter to summarization is the level of compression desired, called *compression rate*, and can be defined as the ratio of the summary length to the source length. Note that while this measure is defined in terms of length, which is easily measured, it could also perhaps be defined in terms of information content. As the compression decreases, more information is lost. Traditionally, compression rates range from 5% to 30%.

2.1.5 Coherence

A crucial concept in summarization is the notion of *coherence*, i.e., the way the parts of the text gather together to form an integrated whole.

An incoherent text is one which is disjointed, where the sentences do not flow together to form a coherent whole. This can be due to anaphors that are unsolved in the text, gaps in the reasoning, sentences which repeat the same or similar idea (*redundancy*), a lack of good organization, etc.

Of course, the level of tolerance for incoherence in a summary will vary with the application. Some applications will accept summaries which are just fragmentary (e.g., a list of words, or phrases). In addition, coherence depends a lot on the output format. It may contain passages of coherent text, or it may contain fragments.

In any given application, the importance of these parameters will vary. It is unlikely that any summarizer will handle all these parameters. In summary, Text Summarization can be illustrated as in Figure 2.1.

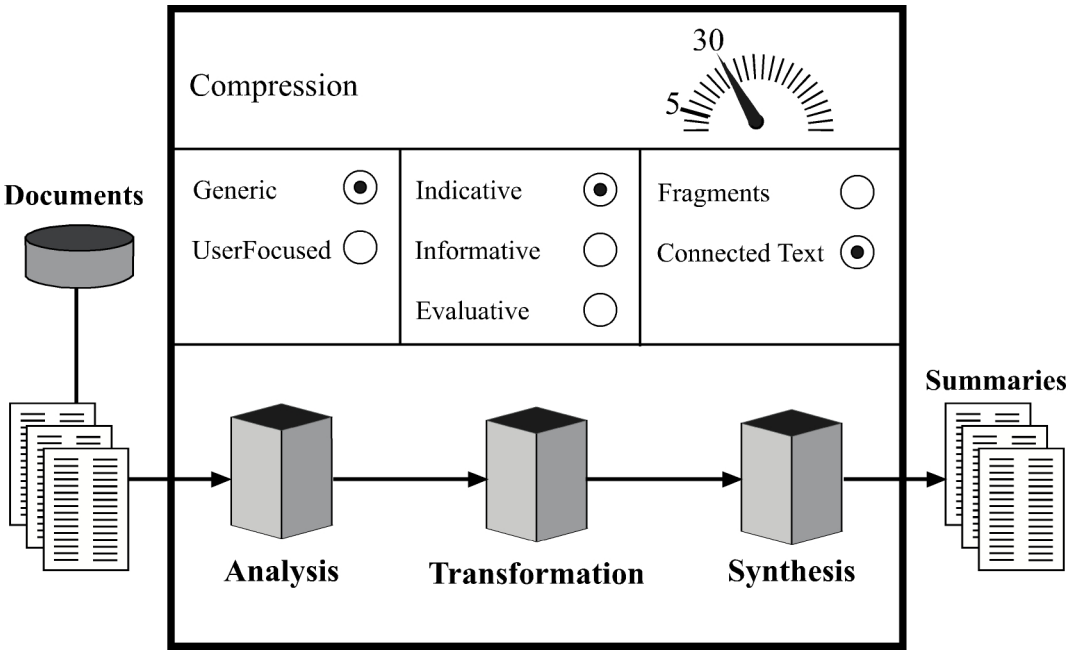

Figure 2.1: Architecture of a Text Summarization System

2.2 Summarization using Lexical Chains

When reading any text it is obvious that it is not merely made up of a set of unrelated sentences, but that these sentences are in fact connected to each other through the use of two linguistic phenomenon, namely *cohesion* and *coherence*. As Morris and Hirst (1991) point out, cohesion relates to the fact that the elements of a text "tend to hang together", while coherence refers to the fact that "there is sense (or intelligibility) in a text".

Observing the interaction between textual units in terms of these properties is one way of analyzing the discourse structure of a text. Coherence is a discourse property that describes the meaning of the text based on the macro-level relations, such as *elaboration*, *explanation*, *cause*, *contrast* and *result*, between sentences or clauses or phrases. In contrast to cohesion, coherence is difficult to identify without a complete understanding of the text and complex inferences. Even humans find it more difficult to identify and agree on textual coherence because, although identifying cohesion and coherence are subjective tasks, coherence requires a definite interpretation of meaning, while cohesion requires only an understanding that terms are about "the same thing".
Consider the example (Hobbs, 1985), where it is difficult to identify the exact coherence relation:

1. *John can open the safe*.

2. *He knows the combination*.

Hobbs (1985) identifies the relation between the two sentences as elaboration, but Morris and Hirst (1991) claim that the relation could also be explanation. They proceeded to state that the precise identification of the coherence relation depends on the belief and context of the reader.
By identifying the relation between the words *safe* and *combination*, a cohesive relation could be established between the two sentences. There is, 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, 1991). But generally cohesion is evident when sentences relate coher-

ently and this trait can be exploited by using cohesion relations as means of identifying coherent parts of the text. Morris and Hirst hypothesize that cohesion is a useful indicator of coherence in text especially since the identification of coherence itself is not computationally feasible at present.

Cohesion, as defined by Halliday and Hasan (1976), enables us to capture the "theme" of the document. It can be defined as the property of the text to hang together as one large grammatical unit, based on relations between words. The most easily identifiable and the most frequent type of cohesion is Lexical Cohesion. Lexical Cohesion arises from the selection of vocabulary items and the semantic relationships between them. For example, in *"I parked outside the library, and then went inside the building to return my books"*, we can identify a generalization relationship between *library* and *building* and a has-part relationship between *library* and *books*.

Lexical chaining is a method of representing the lexical cohesive structure of a text. Lexical Chains are in essence sequences of semantically related words spread over the entire text spanning a topical unit of the text (Morris and Hirst, 1991), where lexical cohesive relationships between words are established using an auxiliary knowledge source such as a dictionary, a thesaurus or an automatically constructed knowledge base, like in our system.

Since Lexical Chains were first proposed by Morris and Hirst (1991), they have been used to address a variety of Information Retrieval (IR) and Natural Language Processing applications. Moreover, in the context of this thesis, Lexical Chains have been successfully used as an effective tool for Automatic Text Summarization as an intermediate source text representation.
Several methods have been proposed to perform both manual (Morris and Hirst, 1991) and automatic computation of Lexical Chains (Hirst and St-Onge, 1997) (Barzilay and Elhadad, 1997) (Silber and McCoy, 2002) (Galley and McKeown, 2003). We review some of the principal chaining approaches proposed in the literature for practical application on Text Summarization.

2.2.1 Morris and Hirst's algorithm

In 1991, Morris and Hirst published their seminal paper on Lexical Chains with the purpose of illustrating how these chains could be used to explore the discourse structure of a text. At the time of writing their paper no machine-readable thesaurus was available so they manually generated Lexical Chains using the Roget's Thesaurus.
The Roget's Thesaurus is one of a category of thesauri (like the Macquarie Thesaurus) that were custom built as aids to writers who wish to replace a particular word or phrase with a synonymous or near-synonymous alternative. Unlike a dictionary, they contain no gloss definitions, instead they provide the user with a list of possible replacements for a word and leave it up to the user to decide which sense is appropriate. An index entry for each word consists in a list of synonyms and near-synonyms for each of its coarse-grained senses followed by a list of category numbers that are related to these senses. A category in this context consists of a list of related words and pointers to related categories. Two words are related if any of the following relationships rules apply: they have a common category in their index entries; one word has a category in its index entry that contains a pointer to a category of the other word; both words have categories in their index entries that point to a common category.

Since then lexical chaining has evolved from an idea on paper to a fully automated process that captures cohesive relationships. In their original chaining algorithm, Morris and Hirst (1991) manually identified lexical cohesive relationships between chain words using a version of the Roget's Thesaurus. Subsequent attempts to automate the chaining process have predominantly focussed on finding Lexical Chains using WordNet (Miller, 1995).

2.2.2 Hirst and St-Onge's algorithm

A first computational model of Lexical Chains was introduced by Hirst and St-Onge (1997). They proposed the first algorithm to automatically compute Lexical Chains, using WordNet as lexical source. Their intention was to use Lexical Chains as a means of detecting malapropisms in text. They define a malapropism as "the confounding of one

word with another word of similar sound or spelling that has a quite different meaning". They provided the following example, "an ingenuous machine for peeling oranges" where "ingenuous" is confused with "ingenious".

Hirst and St-Onge's biggest contribution to the study of Lexical Chains is the mapping of WordNet relations and paths (transitive relationships) to Morris and Hirst's word relationship types. Hirst and St-Onge classified the relations into three categories:

1. Extra-strong relations: between a word and its repetition, e.g. "men" and "man".

2. Strong relations: between two words connected by a WordNet relation, such as for example, synonymy (*machine, device*). Strong relations can also indicate a shared hypernym/hyponym or meronym/holonym, such that one word is a parent-node or child-node of the other in the WordNet topology.

3. Medium-strength relations: relationships with allowable paths in WordNet (with a maximum path length equal to 5).

They limit the chaining process to nouns but, they do not use a part-of-speech tagger "in order to avoid the slowdown and the error that would have resulted". Instead, the algorithm selects only those words that contain noun entry in WordNet to compute Lexical Chains. Each candidate word sense is included in one Lexical Chain, in which it has relation with the last entered chain member. In case of multiple compatible chains, extra strong relations are preferred over the strong relations, both of which are preferred over the medium-strength relations. Once the word sense is inserted into a chain with the appropriate sense, all the non-compatible senses of the word are discarded. 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. In case no compatible chain is found then a new chain is created with all the senses of the word.

Hirst and St-Onge base their malapropism detector on the following hypothesis: words that do not form Lexical Chains with other words in a text are potential malapropisms, as they appear to be semantically dissimilar to the general context of the text. Once these potential malapropisms have been detected the algorithm then tries to find slight spelling

variations of these words that fit into the overall semantics of the document. Hence, if one of these spelling alternatives forms a relationship with one of the Lexical Chains, then the original word was incorrectly used and the variation of the word was the intended use.

There are two categories of chaining approaches proposed in the literature: those that attempt to create all possible chains and then choose the best of these chains (Barzilay and Elhadad, 1997; Silber and McCoy, 2002); and those that disambiguate terms before noun clustering begins resulting in a single set of chains (Galley and McKeown, 2003). We now examine these approaches in detail.

2.2.3 Barzilay and Elhadad's algorithm

Barzilay and Elhadad (1997) proposed the first dynamic method to compute Lexical Chains.

Hirst and St.Onge's algorithm uses a greedy disambiguation procedure where a word sense is determined only by the senses of words that occur before it in the text. In contrast, a non-greedy approach waits until all words in the document are processed and then calculates the appropriate senses of all the words. Barzilay and Elhadad's form chains using a non-greedy disambiguation procedure.

Barzilay and Elhadad were the first to discuss the advantages of a non-greedy chaining approach. They argued that the most appropriate sense of a word could only be chosen after examining all possible lexical chain combinations that could be generated from a text. Because all possible senses of the word are not taken into account, except at the time of insertion, potentially pertinent context information that appears after the word is lost.

Barzilay and Elhadad's method differs from the Hirst and St-Onge (1997) method in the following aspects:

a) A non-greedy disambiguation heuristic to select the appropriate senses of chain members.

b) Selection of candidate words for chain computation: they consider both nouns and compound nouns identified by Brill's part-of-speech tagger (Brill, 1992), while

Hirst and St'Onge only select tokens that happen to occur as nouns in WordNet. This eliminates the wrong inclusion of the words such as *read*, which have both noun and verb entries in WordNet. Compound nouns are identified using the shallow-based parser (developed by Ido Dagan's team at Bar-Ilan University) which uses a simple characterization of noun sequences. They reduce compound nouns to their head noun. For example, in *quantum computer* which is not present in WordNet, the word *quantum* is not selected. Thus, *quantum computer* is related to *machine* as a *computer*.

c) Segmentation of the text: Using Hearst (1997) TextTiling algorithm, they divided the text into smaller segments. This enhances the analysis of the document content for better understanding of the various topics in the text.

Their dynamic algorithm begins by extracting nouns and compound nouns. As each target word arrives, a record of all possible chain interpretations is kept and the correct sense of the word is decided only after all chain combinations have been completed.
Their process of lexical chaining consists of the following steps: select a set of candidate words from the text; for each of the candidate words, find an appropriate chain to receive the candidate word, relying on a relatedness criterion among members of the chains and the candidate words; if such a receiving chain is found, insert the candidate word in this chain and update it accordingly otherwise, create a new chain.

This method of retaining all possible interpretations, until the end of the process, causes the exponential growth of the time and space complexity. To reduce this algorithmic complexity, Barzilay and Elhadad's dynamic algorithm continually assigns each chain interpretation a score. Each interpretation score is equal to sum of all its chain scores. Each chain score is determined by the number and the weight of the relations between chain members. Under the assumption that the text is cohesive, the higher scoring interpretation is retained as the best possible interpretation.
When the number of interpretations for a particular word sense exceeds a certain threshold (i.e. 10 chains), "weaker interpretations" with lower scores are removed from the chaining process. Further reductions in the runtime of the algorithm are also achieved. In

particular, relationships between words are only permitted if words occur in the same text segment. Hearst's (1997) TextTiling algorithm was used to segment the document into sub-topics or text segments.

Finally, they build chains in every segment according to relatedness criteria, and merge chains from the different segments using much stronger criteria for connectedness only: two chains are merged across a segment boundary only if they contain a common word with the same sense. The intra-segment relatedness criterion is less strict: members of the same synsets are related; a node and its offspring in the hyperonym graph are related; siblings in the hyperonym graph are related only if the length of the path is less than a threshold.

Once all chains have been generated only the strongest chains are retained.

Barzilay and Elhadad (1997) provide a rigorous justification of their chain weighting scheme. In particular, they use a human evaluation to determine what chain characteristics are indicative of strong chains (representing pertinent topics) in a text, i.e. chain length, chain word distribution in the text, chain span, chain density, graph topology (of chain word relationships in WordNet) and the number of word repetitions in the chain. Barzilay and Elhadad found that the best predictors of chain importance or strength were: the *chain length* (the number of occurrences of members of the chain) and the *homogeneity index* (one minus the number of distinct occurrences of words in the chain divided by the length of the chain). A single measure of chain strength is calculated by combining chain length with homogeneity index. See Equation (2.1).

$$Score(Chain) = Length \times HomogeneityIndex \tag{2.1}$$

When ranking chains according to their score, they evaluate that "strong" chains are those which satisfy the "Strength Criterion". Chain scores that exceed an average chain scores plus twice the standard deviation of this average are considered "strong" chains, and they will be retained, as shown in Equation (2.2).

$$Score(Chain) > Average(Scores) + 2 \times StandardDeviation(Scores) \tag{2.2}$$

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. The main focus is that chains represent pertinent themes in a document, significant sentences that should be included in a summary can therefore be identified by examining the distribution of chains throughout the text. Sentences are then extracted from chains based on one of three heuristics:

1. For each chain in the summary representation choose the sentence that contains the first appearance of a chain member in the text. The problem with this approach is that all words in a chain reflect the same concept, but to a different extent.

2. For each chain choose a "representative" word (i.e. the term with the highest frequency of occurrence), then extract the first sentence in the text that contains a representative word for each of the chains.

3. Often, the same topic is discussed in a number of places in the text, so its chain is distributed across the whole text. For each chain, find the text unit where the chain is highly concentrated. The sentence with the first chain appearance in this central unit is then extracted.

Note that in all these three techniques only one sentence is extracted for each chain. The better results are generally obtained with the second heuristic. This heuristic produces the best summaries.

A data set of 40 TREC news articles containing roughly 30 sentences each was chosen for the evaluation. Five human subjects were asked to produce summaries of length 10 % and 20 % respective of the length of the original document. Barzilay and Elhadad then compared the similarity of these manually constructed summaries to those generated by their lexical chain-based system, the Microsoft Summarizer available in Word97, and Marcu's summarizer based on discourse structure analysis (Marcu, 1997).
Results from these experiments showed that Barzilay and Elhadad's lexical chain-based summaries were closer to human generated summaries than either of the other two systems.

Results from these experiments showed that Lexical Chains are a strong intermediate representation of a document, and hence should perform well in other applications that would benefit from a more meaningful representation of a document than a bag-of-words representation.

Barzilay and Elhadad (1997) argued that disambiguating a term after all possible links between it and the other candidate terms in the text have been considered was the only way to ensure that the optimal set of Lexical Chains for that text would be generated. However, with this potential improvement in Lexical Chain quality comes an exponential increase in the runtime of the basic chaining algorithm, since all possible chaining scenario must be considered. This has lead to a number of recent initiatives to develop a linear time algorithm that attempts to improve chaining accuracy without over-burdening CPU and memory resources.

2.2.4 Silber and McCoy's algorithm

An important extension of Barzilay and Elhadad's work has been Silber and McCoy's (2002) linear time version of their lexical chaining algorithm. They make two modifications to Barzilay and Elhadad's algorithm in order to reduce its runtime.

The first modification relates to the WordNet searching strategy used to determine word relationships in the taxonomy. In the original implementation (Barzilay and Elhadad, 1997) use the source code accompanying WordNet to access the database, resulting in a binary search of the input files. Silber and McCoy note that chaining efficiency could be significantly increased by re-indexing the noun database by line number rather than file position, and saving this file in a binary indexed format.
Consequently, this also meant writing their own source code for accessing and taking advantage of this new arrangement of the taxonomy.

Their second modification to Barzilay and Elhadad's algorithm relates to the way in which "chain interpretations" are stored, where the original implementation explicitly stores all interpretations (except for those with low scores), resulting in a large runtime storage overhead. To address this, Silber and McCoy's implementation creates "a struc-

ture that implicitly stores all chain interpretations without actually creating them, thus keeping both the space and time usage of the program linear".

In particular, they create an internal representation, *meta-chains*. Words are inserted into those meta-chain entries with which they have the relations such as *identical, synonymy, hypernym/hyponym, sibling* (if the words have the same hypernym). Each meta-chain value is equal to the *offset* value in WordNet.

Once all chain interpretations, or *meta-chains* as Silber and McCoy refer to them, are created, their algorithm must then decide which meta-chains are members of the optimal set of Lexical Chains for that document. To decide this, their algorithm makes a second pass through the data taking each noun in the text, and deciding which meta-chain it contributes the most to (based on the meta-chain scores). The strength of a noun's contribution to a chain depends on two factors: how close the word is in the text to the word in the chain to which it is related, and how strong the relationship between the two words is. For example, if a noun is linked by a hypernym relationship to a chain word that is one sentence away then it gets assigned a score of 1, if the words are 3 sentences away the score is lowered to 0.5. Silber and McCoy define an empirically-based scoring system for each of the WordNet relationships found between terms during chaining. The subsequent steps of their algorithm proceed in a similar way to Barzilay and Elhadad one's, where only chains that exceed a threshold (twice the standard deviation plus the average chain score) are selected for the final stage of the summarisation process. See Equation (2.2).

Due to these improvements in time/space complexity, Silber and McCoy approach does not impose an upper limit on document size.

2.2.5 Galley and McKeown's algorithm

Galley and McKeown (2003) propose a chaining method that disambiguates nouns prior to the processing of Lexical Chains.

This algorithm can be decomposed into three steps: building a representation of all possible interpretations of the text (similar to meta-chains of Silber and McCoy), disambiguating all words, and finally building the Lexical Chains.

At first, an implicit representation of all possible word interpretations in the text called *disambiguation graph*, is created in linear time, where each node corresponds to a distinct sense of a word. Edges connecting the nodes represent the weighted relation between the two particular senses. Like most other techniques these relationships between words are weighted with respect to two factors: the strength of the semantic relationship between them, and their proximity in the text. Distance factors for each type of semantic relation prevent the linkage of words that are too far apart in the text.

Galley's weighting scheme is nearly identical to Silber and McCoy's (2002). However, their relationship-weight assignments are in general lower for terms that are further apart in the text.

Once every word is processed, the disambiguation graph is used to identify the sense of the word based on its relations. Nouns are disambiguated by summing the weights of all the edges or paths emanating from each sense of a word to other words in the text.
The word sense with the highest score is considered the most probable sense and all remaining redundant senses are removed from the graph. Once each word has been fully disambiguated, Lexical Chains are generated from the residual edges from the disambiguation graph.

Another interesting aspect of Galley and McKeown's paper is that they evaluate the performance of their lexical chaining algorithm with respect to the disambiguation accuracy of the nouns in the resultant Lexical Chains. Their evaluation shows that, their algorithm was more accurate than Barzilay and Elhadad and Silber and McCoy ones, with this respect.

2.2.6 Our approach

Instead of using the standard linguistic resource WordNet, our algorithm identifies lexical cohesive relationships between words based on corpus evidence using an automatically constructed lexico-semantic knowledge base. Unlike the approaches which use WordNet as a knowledge source, we also examine lexical cohesive relationships that cannot be defined in terms of thesaural relationships, but are considered "intuitively" related due to

their regular co-occurrence in text.

In order to select candidate words for the chaining process we tag the words with the TnT tagger (Brants, 2000) and extract compound nouns using the SENTA software (Dias, 2002). Finally nouns, compound nouns and proper nouns are filtered and considered as candidate words for further processing of Lexical Chains.

Each candidate word may have multiple possible meanings that can only be determined by considering the context in which it occurs. We disambiguate this word by grouping it with other similar words using the clustering algorithm PoBOC (Cleuziou *et al*., 2004). In that way, words that are grouped together in the same class represent a particular word sense or concept.

In the case of all other systems except Hirst and St-Onge (1997), compound nouns are considered as possible candidate words, only if they have a valid entry in WordNet. If the compound noun does not have a valid entry, the modifiers are removed and only the main noun is considered. By using SENTA, we consider all compound nouns that are present in the text.

In order to build Lexical Chains we could have used the approach of Silber and McCoy (2002) to create a meta-chain for each cluster, but we do not have the same level of detail as in WordNet in our knowledge base. In their algorithm, words are inserted into those meta-chain entries with which they have the relations such as *identical, synonymy, hypernym/hyponym, sibling*. In our case, with our lexico-semantic knowledge base this could not be the case. In fact, we propose a lexical chainer based on a mixture of Barzilay and Elhadad (1997) and Hirst and St-Onge (1997) algorithms. In fact, we allow a word to have many senses in the same text, so that it can participate in many Lexical Chains.

In order to assign a given word to a given Lexical Chain we first calculate the average of all semantic similarities between all the concepts/clusters present in the Lexical Chain and in the word. To determine if two concepts/clusters are semantically related we use a measure of semantic similarity proposed by (Lin, 1998) based on the ratio between the information content of the most informative common ancestor and the information content of both concepts in our lexico-semantic knowledge base.

In fact, two words are semantically related if the semantic similarity between two

clusters is greater than a given threshold. This threshold is the average of all semantic similarities between all clusters and is weighted by a given constant called parameter.

Finally, to identify strong chains, a chain score is computed as the sum of similarities between all the chain members. In particular, we do not consider the factor of how close the words are in the text because this factor works as an "ad hoc" heuristic and the other reason is that for future work we want to use a Topic Segmentation System (Dias and Alves, 2005) which should solve this problem of closeness of words in the text.

To reinforce the idea of how our lexical chainer process is computed, we show again its architecture in Figure 2.2.

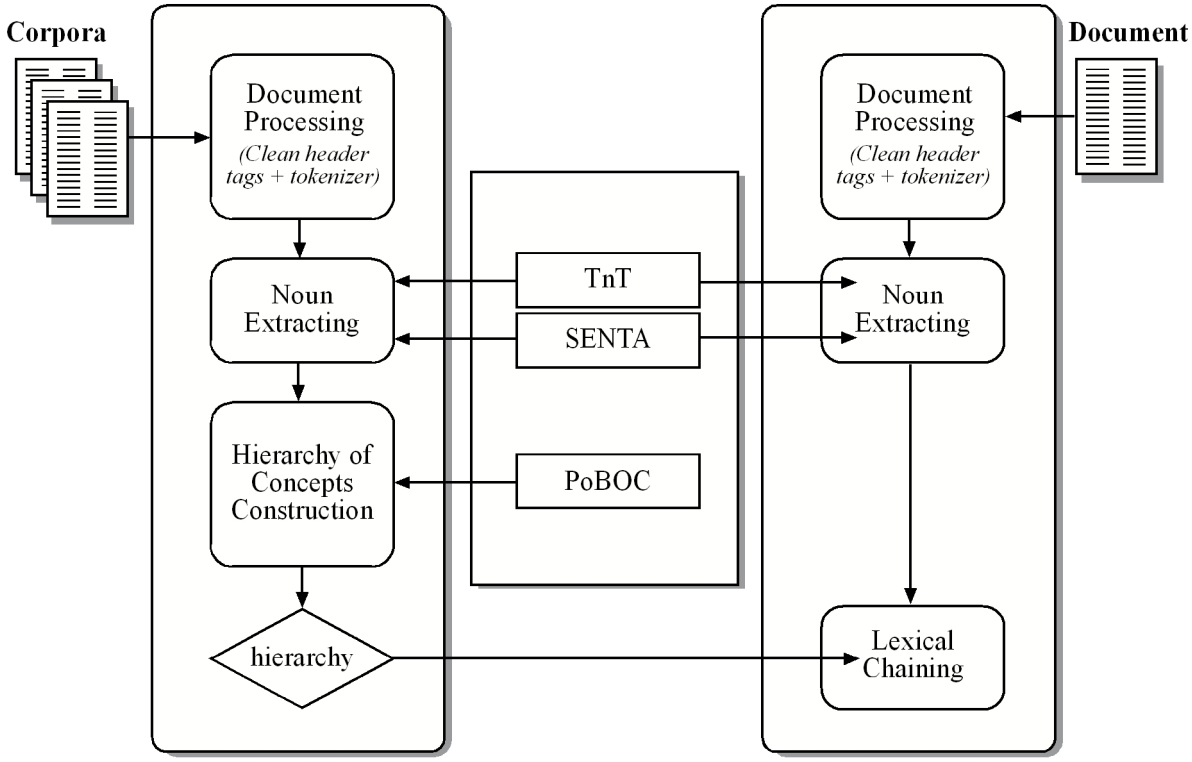

Figure 2.2: Architecture of the lexical chainer system.

Chapter 3

Construction of the Lexico-Semantic Knowledge Base

Just like mentioned before, previous algorithms to compute Lexical Chains construct lexical cohesion relations based on WordNet (Miller, 1995). The most obvious difference with our work is that our system identifies lexical cohesive relationships between words by using an automatically constructed lexico-semantic knowledge base, based on a background collection of documents, a context corpus.
These acquired relations will enable us to further build chains of candidate words that are semantically related.

The construction of a lexico-semantic knowledge source requires huge corpora. A problem in Text Summarization is the lack of training corpora. Ideally, there would be plenty of examples of texts and their corresponding summary. One solution is provided by DUC [1], from which we can obtain (document, abstract) pairs. The abstract provided is a good resource for evaluation in Summarization. In order to build our lexico-semantic knowledge base, we consider the original documents in four different domains: Economy, Politics, Sport and War.

In a first stage, a new lexico-semantic knowledge base is developed for each domain in order to use as specific as possible information, although this algorithm can be extended

[1] DUC2004 http://www-nlpir.nist.gov/projects/duc/

for cross-domain information.

3.1 Preprocessing

To start the construction process, the original collection of texts is first formated to filter the header tags and extract the textual information. Input texts, free of header tags, are then tokenized to separate each word into individual tokens, with the aim to select words to consider in the construction of the knowledge base. Tokens are then morpho-syntactically tagged, compound words extracted, and finally nouns, compound nouns and proper nouns filtered for further processing.

3.1.1 Tagging

The tagging module involves classifying words according to the part of speech they represent. In this process, known as *part-of-speech (PoS) tagging*, words are considered individually, and no semantic structure is considered or assigned by the tagger. The tagger used in our system is the TnT tagger developed by Thorsten Brants (2000), which is based on trigram statistics.

TnT, the short form of Trigrams'n'Tags, is a very efficient statistical part-of-speech tagger that is trainable on different languages and virtually any tagset. The component for parameter generation trains on tagged corpora and the system incorporates several methods of smoothing and of handling unknown words.

TnT is not optimized for a particular language. Instead, it is optimized for training on a large variety of corpora and adapting the tagger to a new language, new domain, or new tagset is very easy. Additionally, TnT is optimized for speed. These two considerations were absolutely important to the sake of staying as language-independent as possible and assure speed of our application.

The tagger is an implementation of the Viterbi algorithm for second order Markov models (Rabiner, 1989). The main paradigm used for smoothing is a linear interpolation and the respective weights are determined by deleted interpolation (Brown *et al.*, 1992). Unknown words are handled by a suffix trie and successive abstractions (Samuelsson,

1993).

The tagging process requires two files containing the model parameters for lexical and contextual frequencies, and an input file with the text to be tagged in the format one token per line. In our system we use *tnt/models/susanne* as the language model that is trained on the Susanne Corpus (Sampson, 1995), consisting of about 150.000 tokens. This English model comes with the distribution of TnT.

The average part-of-speech tagging accuracy is between 96% and 97%, depending on language and tagset, which is at least *en par* with the state-of-the-art results found in the literature.

A detailed description of the techniques used in TnT is given in the TnT home page [2] (in particular, see the TnT manual [3]).

3.1.2 Extraction of N-ary Textual Associations

After tagging the texts, we need to identify multi-word expressions (or collocations) that should be automatically extracted from corpora, in order to enable their incorporation in the set of the candidate words of the lexical chainer.

A multi-word lexical unit is defined according to (Choueka, 1988) as a *"sequence of two or more consecutive words, that has characteristics of a syntactic and semantic unit, and whose exact and unambiguous meaning or connotation cannot be directly derived from the meaning or connotation of its components"*, and according to Smadja (1993) as a *"recurrent combination of words that co-occur more often than expected by chance and that correspond to arbitrary word usage"*.

Usually, these units appear frequently in real texts. Let us take the illustrative example of a frequently co-occurring multi-word unit: *Tony Blair*. It is clear for us that this 2-gram is a compound proper name i.e., a lexical unit. In fact, when the word *Tony* appears in a text, it is likely that the word *Blair* will follow it, and the probability of the word *Tony* appearing in the position immediately prior to *Blair* is very high too. And so we may think of a "glue" sticking those 2 words together.

[2] http://www.coli.uni-saarland.de/ thorsten/tnt/

[3] http://www.coli.uni-saarland.de/ thorsten/publications/Brants-TR-TnT.pdf

However, it is also possible to find sentences where the words *Tony* and *Blair* do not appear together. This means that there is some word "dispersion" in the next position after word *Tony* as well as there is some kind of "dispersion" for the word that may precede *Blair*. However these two words still have a strong "glue" sticking them together as they frequently co-occur. Moreover we claim that this "glue" has a sufficiently high value for *Tony Blair* to be considered a 2-gram unit.
Another example of multi-word unit is *hot dog*, this collocation should never be broken down into *hot* and *dog*, since the meaning of *hot dog* cannot be compositionally derived from the individual words.

The software used for this stage is the SENTA software (Software for the Extraction of N-ary Textual Associations). SENTA is a software that has been designed by Dias (2002) and is a statistically-based extractor of textual associations. In order to extract multi-word units, this system conjugates one association measure based on the concept of normalized expectation called the Mutual Expectation (Dias *et al.*, 1999) with one multi-word acquisition process based on an algorithm of local maxima called the GenLocalMaxs (Dias, 2002).
On one hand, the Mutual Expectation evaluates the degree of cohesiveness that links together all the textual units contained in an n-gram (i.e. $\forall n, n \geq 2$). On the other hand, the GenLocalMaxs retrieves the candidate terms from the set of all the valued n-grams by evidencing local maxima of association measure values.

Mutual Expectation

The normalized expectation existing between n words is the average expectation of one word occurring in a given position knowing the presence of the other $n-1$ words also constrained by their positions. The underlying concept is based on the conditional probability presented in Equation (3.1).

$$p\big(X = x|Y = y\big) = \frac{p\big(X = x, Y = y\big)}{p\big(Y = y\big)} \tag{3.1}$$

where $p\big(X = x, Y = y\big)$ is the joint discrete density function between the two random variables X, Y and $p\big(Y = y\big)$ is the marginal discrete density function of the variable Y. However, this definition does not accommodate the n-gram length factor. Naturally, an n-gram is associated to n possible conditional probabilities. The Normalized Expectation, based on a normalization of the conditional probability, proposes an elegant solution to represent in a unique formula all the n conditional probabilities involved by an n-gram. It is necessary to calculate an average conditional probability. For that purpose (Dias, 2002) introduces the concept of the Fair Point of Expectation (FPE) which is defined in Equation (3.2).

$$FPE\big([p_{11}u_1p_{12}u_2...p_{1i}u_i...p_{1n}u_n]\big) = \frac{1}{n}\Bigg(p\big([p_{12}u_2...p_{2i}u_i...p_{2n}u_n]\big) + \\ + \sum_{i=2}^{n} p\big([p_{11}u_1...\widehat{p_{1i}}\widehat{u_i}...p_{1n}u_n]\big)\Bigg) \tag{3.2}$$

where $[p_{11}u_1p_{12}u_2...p_{1i}u_i...p_{1n}u_n]$ is a generic n-gram where p_{11} is equivalent to zero and p_{1i} (for $i = 2, ..., n$) denotes the signed distance that separates the textual unit u_i from its pivot u_1; $p\big([p_{12}u_2...p_{2i}u_i...p_{2n}u_n]\big)$, for $i = 3, ..., n$ is the probability of the occurrence of the (n-1)-gram $[p_{12}u_2...p_{2i}u_i...p_{2n}u_n]$ and $p\big([p_{11}u_1...\widehat{p_{1i}}\widehat{u_i}...p_{1n}u_n]\big)$ [4] is the probability of the occurrence of one (n-1)-gram containing necessarily the first unit u_1 of the n-gram.

So, the normalized expectation of a generic n-gram is defined as being a "fair" conditional probability and is realized by the introduction of the Fair Point of Expectation into the general definition of the conditional probability. (See Equation (3.3)).

$$NE\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big) = \frac{p\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big)}{FPE\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big)} \tag{3.3}$$

where $p\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big)$ is the probability of occurrence of the n-gram $[p_{11}u_1...p_{1i}u_i...p_{1n}u_n]$ and $FPE\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big)$ its normalized expectation.

[4] The "^" corresponds to a convention frequently used in Algebra that consists in writing a "^" on the top of the omitted term of a given succession indexed from 1 to n.

However, (Daille, 1995) shows that one effective criterion for multi-word unit identification is simple frequency. From this assumption, between two n-grams with the same normalized expectation, the most frequent n-gram is more likely to be a relevant multi-word unit. So, the Mutual Expectation between n words is defined based on the normalized expectation and the relative frequency of the particular n-gram $[p_{11}u_1...p_{1i}u_i...p_{1n}u_n]$, $f\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big)$. (See Equation (3.4)).

$$ME\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big) = f\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big) \times NE\big([p_{11}u_1...p_{1i}u_i...p_{1n}u_n]\big) \tag{3.4}$$

GenLocalMaxs

The GenLocalMaxs algorithm (Dias, 2002) elects the multi-word units from the set of all the valued n-grams based on two assumptions. First, all the association measures show that the more cohesive is a group of units, the higher its score will be. Second, multi-word units are highly associated localized groups of words. As a consequence, an n-gram is a multi-word lexical unit if its association measure value is a local maximum, i.e. if its association measure value is higher or equal than the association measure values of all its sub-groups of $(n-1)$ words and all its super-groups of $(n+1)$ words.

One important property of the GenLocalMaxs algorithm is that it can be tested with any association measure that respect the following assumption: the more cohesive a sequence of words is, the higher its association measure value will be. The other important property is that GenLocalMaxs allows extracting multi-word units obtained by composition of one or more other units. For example, the GenLocalMaxs conjugated with the Mutual Expectation, elects the multi-word unit *Prime Minister Tony Blair* built from the composition of the extracted units *Prime Minister* and *Tony Blair*. This situation is illustrated in Figure 3.1.

The association measure value of *Prime Minister Tony* should be lower than the one for *Prime Minister* as there are many possible words, because there are many Prime Ministers other than *Tony*, that may occur after *Prime Minister*. Thus, the association measure

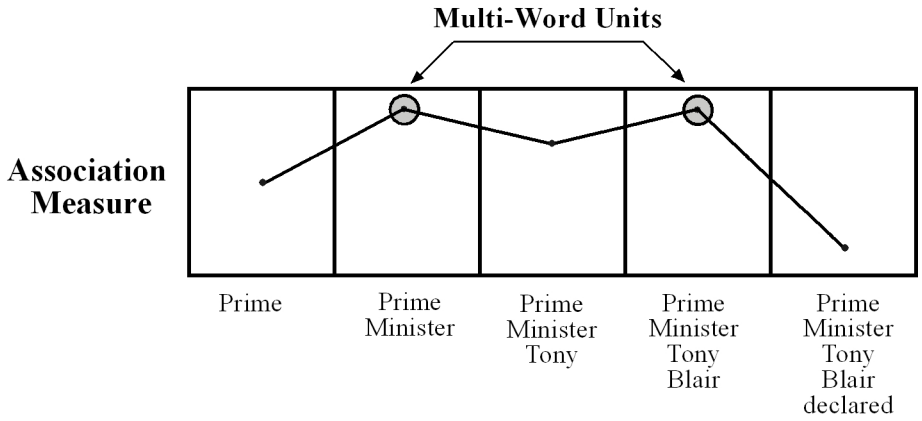

Figure 3.1: Election of multi-word units by Composition

of any super-group containing the unit *Prime Minister* should theoretically be lower than the association measure for *Prime Minister*. But, if the first name of the *Prime Minister* is *Tony*, the expectation to appear *Blair* is very high and the association measure value of *Prime Minister Tony Blair* should then be higher than the association measure values of all its sub-groups and super-groups, no word can be expected to strengthen the overall unit *Prime Minister Tony Blair*.

Once more, the choice of SENTA was clear as it is language-independent and provides a fast implementation (Gil and Dias, 2003).

3.1.3 Noun Retrieving

The noun retrieving component is an optimization, rather than a required component, of the construction of our lexico-semantic knowledge base. The nouns come from the source corpus and are identified by the TnT tagger. In our study, we also consider multi-word units identified with SENTA that are compound nouns (e.g. *President Bill Clinton*, in this case a proper noun). For that purpose, we have used a set of well-known heuristics to retrieve the compound nouns using the idea that groups of words that correspond to *a priori* defined syntactical patterns (Noun+Adj, Noun+Prep+Noun etc...) can be identified as compound nouns (Habert and Jacquemin, 1993).

Nominal unit retrieving improves the accuracy of lexical chainer by selectively consid-

ering nouns, compound nouns and proper nouns as candidate words to compute Lexical Chains. This is based on the intuition that nouns characterize the event in the document and that most of the previous researches propose this choice in way to consider the candidate words. Indeed, nouns usually convey most of the information in a written text. They are the main contributors to the "aboutness" of a text. So, identifying lexico-semantic connections between nouns is an adequate means of determining cohesive ties between textual units. However, we acknowledge that verbs and adjectives should also be tackled in future work.

3.2 Word-Context Vectors

Word-context vectors have been explored as an automated method for representing information based on the local context of words in texts. A word-context vector is a high dimensional vector consisting of real-valued components. The closeness of vectors in the space is equivalent to the closeness of the subject content. Terms that are used in a similar local context will have vectors that are relatively close to each other. Word-context vectors have proven to be a promising method for knowledge representation. Their ability to automatically generate representations from a text has led to their incorporation into this system.

For the specific task of construction of the lexico-semantic knowledge base, we have first implemented the Symmetric Conditional Probability (SCP) (Silva *et al.*, 1999) to build a set of vectors of context for each noun in a text. The intentions of using this association measure is to infer with maximum reliability the "true" co-occurrence between two words. The similarity between words is measured numerically, based on co-occurrence frequency. We use a measure that, takes into account word co-occurrences automatically acquired from corpora, in order to avoid the accessibility to existing linguistic resources such as electronic dictionaries, lexico-semantic databases, thesauri or ontology. So, that we remain as language-independent as possible.
This measure is defined in the next Equation (3.5)

$$SCP(w_1, w_2) = p(w_1|w_2) \times p(w_2|w_1) = \frac{p(w_1, w_2)^2}{p(w_1) \times p(w_2)} \tag{3.5}$$

where $p(w_1, w_2)$, $p(w_1)$ and $p(w_2)$ are respectively the probabilities of occurrence of the nominal units (w_1, w_2) and the nominal units w_1 and w_2 in the corpus.

The distance between words is one of the factor that influences relatedness. Consider two words *machine* and *device*, the probability that they are actually connected is much higher if *machine* and *device* belong to adjacent text segments than two different parts of the text. The SCP measure between two words w_1 and w_2 is calculated within a fixed word-context window (of any size defined by the user) in order to determine $p(w_1, w_2)$ in a corpus so that we can evaluate the degree of cohesiveness between this words outside the context of the document. In our experiments, we have defined this size to 20 words as we will show in the Evaluation chapter.
The higher the association measure between words is, the more related they should be. High values indicate lexical closeness. So, we represent each nominal unit with the respective word-context vector that gathers the N most related nominal units for the sibling word.

In Figure 3.2 we can see two examples of the word-context vectors of two nominal units *judge* and *jail*. These word-context vectors are composed by the N best scored nominal units relatively to each considered nominal unit.

judge	guilt	innocence	coalition	ruling	giant	Court	trust	Rally	comments	Saturday
jail	Security	act	associates	law	charge	police	Anwar	Sept	risk	trial

Figure 3.2: N best scored nominal units, with N=10.

3.3 Similarity Matrix

The next step of our system is to calculate word similarity. So, for each word in the text, the system calculates its similarity with all the next words based on the Informative Similarity Measure proposed by (Dias and Alves, 2005) applied to the word-context vectors. In particular, it includes in its definition the SCP measure, in order to introduce a certain degree of semantic knowledge in our system. Like that, word co-occurrence information is directly embedded in the calculation of the similarity between words.

3.3.1 Weighting score

Having obtained the set of terms that represent each document, the next step is to recognize all words that are relevant. Indeed, each word is assigned a numerical weight, depending on its importance.

Most of the previous researches only rely on frequency and/or the well-known *tf.idf* measure proposed by (Sparck-Jones, 1972; Salton *et al.*, 1975) of their lexical items. However, we consider that better weighting measures can be used. The importance of a word in a document does not only depend on its frequency. Frequency can only be reliable for technical texts where ambiguity is drastically limited and word repetition largely used.

The weighting score of any word in a document can be directly derived from an adaptation of the proposed score in (Dias and Alves, 2005). We just considered the combination of two main heuristics: the well-known *tf.idf* measure proposed by (Sparck-Jones, 1972; Salton *et al.*, 1975) and a new density measure that calculates the density of each word in the text i.e. if the occurrences of the same word are close to each other in the text or not. As a matter of fact, by combining these two scores, we deal with the two main factors that must be taken into account to define the relevance of a word: its semantic importance and its localization in the document.

The *tf.idf* score

First, a recap of what *tf.idf* values mean. The assumption of this score is that term importance is proportional to the frequency of the term in the document, and inversely propor-

tional to the total number of documents in the corpus where the term occurs. The basic idea is to evaluate the importance of a word within a document based on its frequency (i.e. frequent words within a document may reflect its meaning more strongly than words that occur less frequently) and its distribution across a collection of documents (i.e. terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection). This score is used to pick out terms that distinguish one document from others in the corpus. The *tf.idf* weight of a term is expressed as a combination of term frequency multiplied by the reciprocal of document frequency. Given a word w and a document d, the term frequency is the number of occurrences of w in d, $tf(w,d)$, divided by the number of words in d, $|d|$. We then compute the inverse document frequency of w by taking the $\log_2$ of the ratio of N, the number of documents in the corpus, to the document frequency of w, i.e. the number of documents in the corpus in which the word w occurs, $df(w)$. (See Equation (3.6)).

$$tf.idf(w,d) = \frac{tf(w,d)}{|d|} \times \log_2 \left(\frac{N}{df(w)} \right) \tag{3.6}$$

A high *tf.idf* value for a term indicates that it could be relevant, since it occurs often in the document but less often in general. The term frequency in the given document shows how important the term is in this document. The document frequency of the term shows how generally important the term is. A high weight in a *tf.idf* ranking scheme is therefore reached by a high term frequency in the given document and a low document frequency of the term in the whole collection. As a result, a word occurring in all documents of the collection will have an inverse document frequency 0 giving it no chance to be a relevant word. On the opposite, a word which occurs very often in one document but in very few other documents of the collection will have a high inverse document frequency as well as a high term frequency and thus a high *tf.idf* score. Consequently, it will be a strong candidate for being a relevant word within the document.

The word density score

The basic idea of the word density measure is to evaluate the dispersion of a word within a document. So, very disperse words will not be as relevant as dense words. In order to evaluate the word density, we use a measure based on the distance (in terms of words) of all consecutive occurrences of the word in the document, to see if the occurrences of the same word are close to each other in the text or not. This measure is defined in Equation (3.7).

$$dens(w,d) = \sum_{k=1}^{|w|-1} \frac{1}{\ln(\text{dist}(\text{occur}(w,k), \text{occur}(w,k+1)) + \text{e})} \tag{3.7}$$

For any given word w, its density $dens(w,d)$ in document d, is calculated from all the distances between all its occurrences, $|w|$. So, $occur(w,k)$ and $occur(w,k+1)$ respectively represent the positions in the text of two consecutive occurrences of the word w and $dist(occur(w,k), occur(w,k+1))$ calculates the distance that separates them in terms of words within the document. Thus, we get a density function that gives higher scores to highly dense words. As a result, a word which occurrences appear close to one another, will show small distances and as a result a high density. On the opposite, a word which occurrences appear far from each other, will show high distances and as a result a small word density.

The weighting score

The weighting score of any word in a document can be directly derived from the previous two heuristics. This score is defined in Equation (3.8).

$$weight(w, \sum d_w) = \overline{tf}.idf(w, \sum d_w) \times \overline{dens}(w, \sum d_w) \tag{3.8}$$

where $\overline{tf}$ and $\overline{dens}$ are respectively the averages of *tf* and *dens* in all the documents and $\sum d_w$, the number of documents in the corpus in which the word w occurs.

Generally, the more frequent and dense a term is within a collection of documents, the more important it is in representing the content of any document of the domain. Thus, a relevant word should evidence a high *tf.idf* score and a high *density* score. We present in Figure 3.3, the scoring of the word-context vectors after the weighting process is performed.

judge	0.000512	0.000512	0.000512	0.00138	0.000487	0.000993	0.000181	0.000583	0.000777	0.000256
	guilt	**innocence**	**coalition**	**ruling**	**giant**	**Court**	**trust**	**Rally**	**comments**	**Saturday**

Figure 3.3: Weighting process of the word-context vector of *judge*.

3.3.2 The Informative Similarity Measure

The next stage aims at determining the similarity between words. Semantic relations are important because the essential content of texts is not revealed by "words" alone. One reason why discovering *similarity* is important is that people use a wide variety of terminology, even when talking about the same concepts. Not only can many different words be used to describe the same concept but each individual word can have a variety of meanings. These are two aspects of word sense variability: many ways of expressing one concept, and many concepts expressed by one word. Comparing the contexts in which words appear allows us to judge similarity between words.

There are a number of ways to compute the similarity between two words. Theoretically, a similarity measure can be defined as follows. Suppose that $X_i = (X_{i1}, X_{i2}, X_{i3}, , X_{ip})$ is a row vector of observations on p variables associated with a label i. The similarity between two words i and j is defined as $S_{ij} = f(X_i, X_j)$ where f is some function of the observed values. In the context of our work, X_i and X_j are being represented as p-dimension vectors, those vectors are the word-context vectors with terms related to i and j. Our goal here is to find the appropriate f function that will accurately evaluate the similarity between two words.

If, for example, each similarity is calculated under the cosine measure, defined in Equation (3.9), which measures the distance between the two vectors

$$S_{ij} = cos(X_i, X_j) = \frac{\sum_{k=1}^{p} X_{ik} \times X_{jk}}{\sqrt{\sum_{k=1}^{p} {X_{ik}}^2} \times \sqrt{\sum_{k=1}^{p} {X_{jk}}^2}} \tag{3.9}$$

only the equal terms of the row vectors X_i and X_j will be taken into account i.e. if both word-context vectors do not have words in common, they will not be similar at all and will receive a cosine value of 0. However, this is not tolerable. Indeed, it is clear that both sentences (1) and (2) are similar although they do not share any word in common:

1. *Ronaldo defeated the goalkeeper once more.*

2. *Real Madrid striker scored again.*

The most interesting idea to avoid word repetition problems is certainly to identify lexical cohesion relationships between words. Indeed, systems should take into account semantic information that could, for instance, relate *Ronaldo* to *Real Madrid striker*.
In order to avoid this problem, we use an appropriate informative similarity measure proposed by (Dias and Alves, 2005) called InfoSimBA and defined in Equation (3.10).
The basic idea of this measure is to integrate into the cosine measure, the word co-occurrence factor inferred from a collection of documents with the SCP association measure.

$$S_{ij} = InfoSimBA(X_i, X_j) = \\ = \frac{A}{\left(\sqrt{\sum_{k=1}^{p}\sum_{l=1}^{p} X_{ik} \times X_{il} \times SCP(W_{ik}, W_{il})} \times \sqrt{\sum_{k=1}^{p}\sum_{l=1}^{p} X_{jk} \times X_{jl} \times SCP(W_{jk}, W_{jl})}\right) + A} \tag{3.10}$$

where $SCP(W_{ik}, W_{jl})$ is the Symmetric Conditional Probability value between W_{ik}, the word that indexes the word-context vector of the document i at position k, W_{jl},

the word that indexes the word-context vector of the document j at position l and $A = \sum_{k=1}^{p} \sum_{l=1}^{p} X_{ik} \times X_{jl} \times SCP(W_{ik}, W_{jl})$.

If we take the word-context vectors X_i and X_j respectively of the words i and j, for each word in the X_i vector, then for each word in the X_j vector, we calculate the product of their weights and then multiply it by the degree of cohesiveness existing between those two words calculated by the SCP association measure. As a result, the more relevant and cohesive the words will be, the more they will contribute for the cohesion within the text.

The value of each similarity is registered in a upper triangular similarity matrix represented in Figure 3.4.

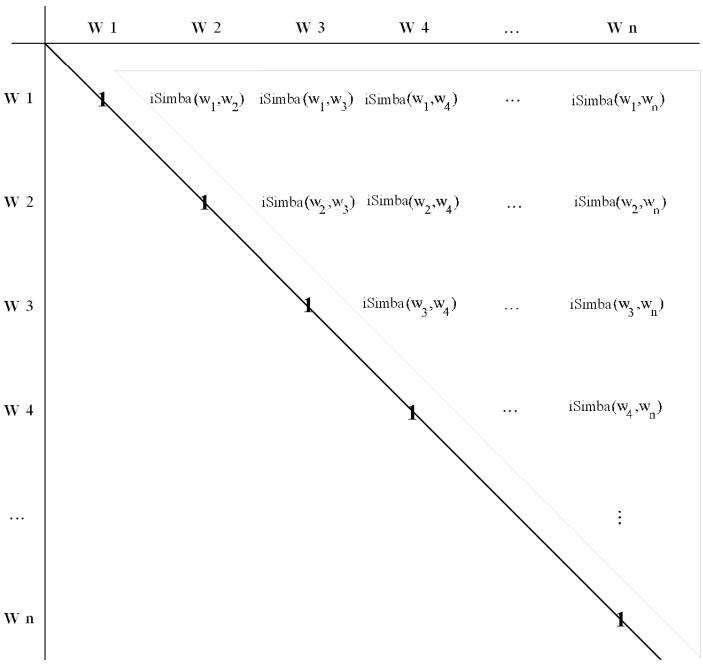

Figure 3.4: Informative Similarity Matrix.

3.4 Determination of clusters of words: PoBOC

Most words in natural language have multiple possible meanings that can only be determined by considering the context in which they occur. Given a target word used in a number of different contexts, word sense disambiguation is the process of grouping these

instances of the target word together by determining which contexts are the most similar to each other. This is motivated by (Miller and Charles, 1991) (Harris *et al*, 1989), who hypothesize that words with similar meanings are often used in similar contexts. Hence, word sense disambiguation is reduced to the problem of finding classes of similar contexts such that each class represents a single word sense. Put another way, contexts that are grouped together in the same class represent a particular word sense.

Clustering is the task that consists in organizing a set of objects into classes, so that similar objects belong to the same cluster and dissimilar ones belong to different clusters. Several ways of clustering have been explored and one sort of algorithms are *hard-clustering* techniques, where each object is assigned to a single cluster. Conversely, the *fuzzy-clustering* methods propose an organization in which each object participate to the definition of each cluster. This last approach is well-known for the richness of its description compared to *hard-clustering* methods. However, hard clusters are usually preferred for the simplicity of their definitions in a post-processing perspective.

The algorithm PoBOC (Pole-Based Overlapping Clustering) (Cleuziou *et al*., 2004) is a method which can be viewed as a compromise between *hard* and *fuzzy-clustering* approaches. Rather than assigning an object to only one cluster, this approach allows an object to belong to one or several clusters (final clusters thus intersect). This type of algorithm is sometimes denoted as *soft-clustering* but the authors prefer to use the term *overlapping-clustering*.

PoBOC is used on the semantic clustering task which consists in structuring lexical units in a semantic perspective.

The algorithm PoBOC takes the previous calculated similarity matrix as input and builds in output a hierarchy of concepts in which an object may belong to several concepts, i.e. produces a set of overlapping clusters hierarchically organized.

From the similarity matrix S defined on $X \times X$ over a set of objects $X = \{x_1, ..., x_n\}$, the algorithm proceeds as follows:

1. construction of a similarity graph $G_S(X, V)$ with V the set of edges,

2. extraction of complete sub-graphs from $G_S(X, V)$, the "poles",

3. multi-assignment of the objects to the poles,

4. hierarchical organization of the obtained groups.

In the first step, the *similarity graph* denoted by $G_S(X, V)$ is based on the set of objects as vertices. V is the set of edges so that there is an edge between two objects x_i and x_j if x_i belongs to the neighborhood of x_j and vice versa. In our case, vertices are nominal units and edges are conditioned by the similarity measure that relates nominal units to each other.
So, we say that x_i is connected to x_j if:

$$s(x_i, x_j) \geq \max \left\{ \frac{1}{n} \sum_{x_k \in X} s(x_i, x_k), \frac{1}{n} \sum_{x_k \in X} s(x_j, x_k) \right\} \tag{3.11}$$

In this definition, an edge exists between x_i and x_j if their similarity is greater than both the average similarity between x_i and the whole set of objects and the average similarity between x_j and the whole set of objects. This step allows to take into account the density around each object and to isolate the outliers.

The second step uses two main heuristics to extract the poles from the similarity graph. Poles correspond to homogeneous areas into the similarity graph. Their construction requires building a set of cliques in the similarity graph.
A pole P_k is a subset of X such that the sub-graph $G_S(P_k, V(P_k))$ is a clique-graph, i.e. $\forall x_i \in P_k, \forall x_j \in P_k,\ (x_i, x_j) \in V(P_k)$, with $V(P_k)$ the subset of edges associated with P_k.
Poles are directly derived from the similarity graph $G_S(X, V)$ first selecting a particular point and then to add connected vertices to the pole in construction.

The choice of starting points is performed by a first heuristic which proceeds as follow. The first vertex chosen x^1 is the one having the lower average similarity with other objects, among the set of vertices having at least one connected vertex. Let E included into X ($E \subset X$) be the set of vertices having at least one connected vertex:

$$x^1 = \arg\min_{x_i \in E} \frac{1}{|X|} \sum_{x_j \in X} s(x_i, x_j) \tag{3.12}$$

The next vertices $\{x^2, ..., x^l\}$ are chosen in order to reduce the similarity with poles previously built:

$$x^k = \arg\min_{x_i \in E} \frac{1}{k-1} \sum_{m=1,...,k-1} \frac{1}{|P_m|} \sum_{x_j \in P_m} s(x_i, x_j) \tag{3.13}$$

The process is iterated until no "well-defined" vertice can be found, when the sum in the previous equation is greater than the average similarity of the whole set of objects. The number of poles is determined automatically by this heuristic, and corresponds to the number of final clusters.

Once a starting vertice x_i in $G_S(X, V)$ is fixed, a second heuristic define the way to add vertices in order to derive a pole as defined previously, a clique-graph.
Searching for maximal cliques in a graph is a NP-complete problem. So the heuristic consists in approximating the maximal clique-graph which contains x_i. A clique is obtained starting from a single vertex and repeatedly adding the nearest neighbour (vertex) until it is not possible to find a vertex x connected to each vertex of the clique under construction.

The next step is the one which assigns each object from X to one or several poles among the set of poles $P = \{P_1, ..., P_l\}$. A new heuristic is used for this assignment stage, so that an object x_i is only assigned to its most similar poles. This multi-assignment step (objects are assigned to poles) plays an important role in the construction of overlapping clusters in PoBOC. The advantage of assigning an object to several clusters is, in that way we can have a word with several senses in different clusters. Each cluster should represent optimally a sense of the word.
In other *soft-clustering* algorithms, the assignment method is often based on a global threshold applied on a membership matrix obtained with a *fuzzy-clustering* method. In the PoBOC algorithm this threshold is not global but local (or dynamical).

Let U be the membership matrix on $P \times X$ where the similarity between an object x_j and a pole P_i is the average similarity between x_j and each object from P_i

$$u(x_j, P_i) = \frac{1}{|P_i|} \sum_{x_k \in P_i} s(x_j, x_k) \tag{3.14}$$

Given an object x_j , $P_{j,1}$ is the most similar pole for x_j $\big(P_{j,1} = \arg\max_{P_i \in P} u(x_j, P_i)\big)$, $P_{j,2}$ the second most similar pole for x_j and so on, $P_{j,l}$ is the least similar pole for x_j. The following condition $assign(x_j, P_{j,k})$ is used to test whether the object x_j is assigned to the pole $P_{j,k}$:
$assign(x_j, P_{j,k})$ iff one of the following properties is satisfied :

i) $k = 1$,

ii) $1 < k < l, u(x_j, P_{j,k}) \geq \frac{u(x_j, P_{j,k-1}) + u(x_j, P_{j,k+1})}{2}$ and $\forall k' < k, assign(x_j, P_{j,k'})$

The first property allows to assign each object to at least one pole (the most similar), and the second allows to assign an object x_j to a pole $P_{j,k}$ by considering the similarity with the previous pole $(P_{j,k-1})$ and the next pole $(P_{j,k+1})$ w.r.t the order associated to x_j. For each object, the set of poles is ordered with respect to its average similarity with the object.

Finally, clustering with PoBOC results in l overlapping clusters.

The clustering algorithm acts as a search strategy that dictates how to proceed through the instances. With the extracted concepts, the next challenge is to organize the relationships among them. Our approach is to generate a concept hierarchy where concepts can be classified and easily observed. We intend to automatically organize a set of words into a concept hierarchy, where an overall structure of them can be observed and important information retrieved. So, the last step is the hierarchical organization of groups.
The algorithm starts with the groups previously built $C = \{C_1, ...C_l\}$ where $C_i = \{x_j|$ assigned to $P_i\}$. Since the similarity matrix is normalized, we have $\forall x_i \in X, s(x_i, x_i) = 1$ and we define the similarity between two groups in Equation (3.15).

$$sim(C_k, C_m) = \frac{1}{|C_k|.|C_m|} \sum_{x_i \in C_k} \sum_{x_j \in C_m} s(x_i, x_j) \tag{3.15}$$

Clustering involves generating a concept hierarchy from estimating the similarities among these concepts. The organization is built as follows: the two most similar groups are agglomerated and this process is repeated until we get only one group. We start with each

instance in a separate cluster and merge a pair of clusters at each iteration until there is only a single cluster remaining.

This organization is represented by a binary tree structure where the leaves correspond to the initial set of groups. Words in the same cluster have high tendency to occur in a similar lexical environment. So, different senses of a given word can be identified within different clusters.

In summary, the final output of the algorithm PoBOC is a hierarchy of concepts in which an object may belong to several clusters/concepts. This four main steps of the algorithm are illustrated in Figure 3.5.

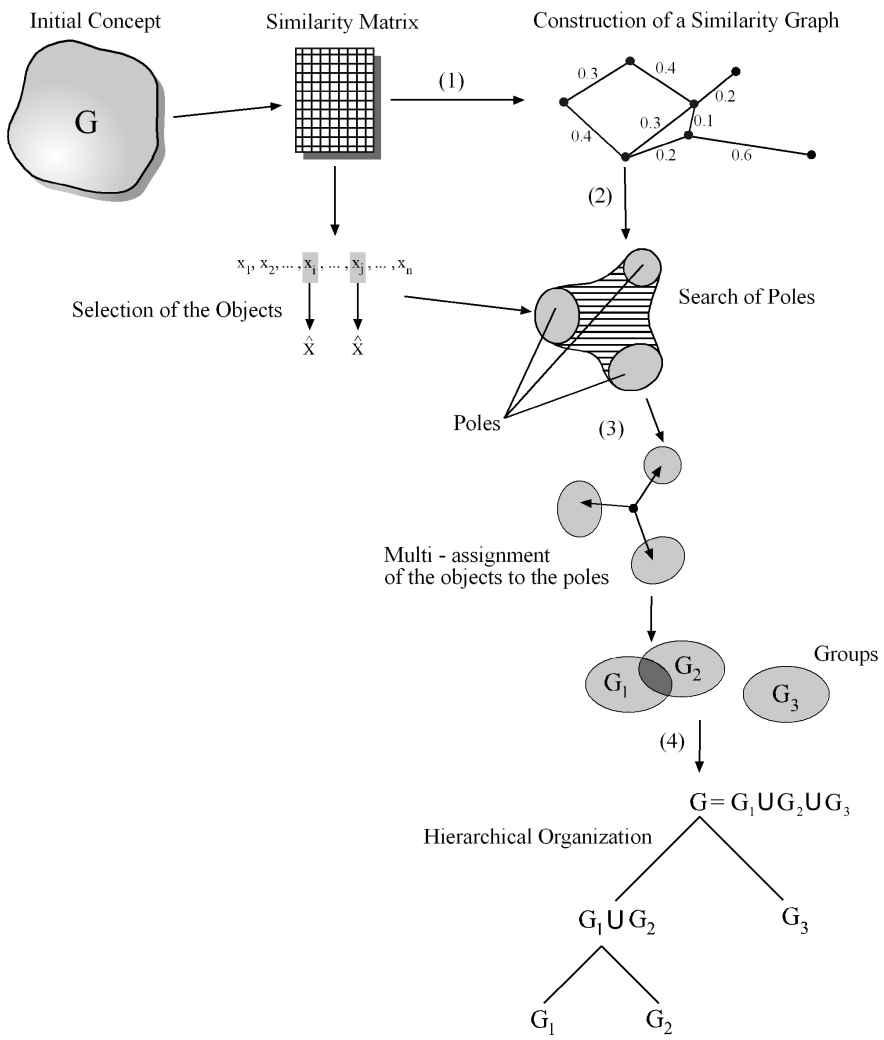

Figure 3.5: PoBOC algorithm.

Chapter 4

ALEXIA: Lexical Chaining for Text Summarization

4.1 Similarity measure in a knowledge base

Semantic resources such our lexico-semantic knowledge base provides information about semantic relations between words and can be used to determine conceptual similarity.

In order to determine if two concepts are semantically related we use our knowledge base and a measure of semantic similarity to obtain information about relationships between concepts.

Semantic similarity (Resnik, 1995) refers to similarity between two concepts in a knowledge representation. Semantic similarity represents a special case of semantic relatedness: for example, *cars* and *gasoline* would seem to be more closely related than, say, *cars* and *bicycles*, but the latter pair are certainly more similar.

A natural way to evaluate semantic similarity in a knowledge base is to evaluate the distance between the nodes corresponding to the items being compared. The relation between two concepts are represented in terms of distance. The shorter the path from one node to another, the more similar they are.

The semantic similarity measure we use in our study is based on the notion of information content proposed by (Lin, 1998). This measure is based on the ratio between

the information content of the most informative common ancestor and the information content of both concepts.

Similarity is an important and widely used concept. Previous definitions of similarity are tied to a particular application or a form of knowledge representation. This measure allows to deal with different domains as long as there is a probabilistic model and is thus adapted to our problem.

Let C be the set of concepts in a knowledge base. Intuitively, one key to the similarity between two concepts is the extent to which they share information in common, indicated by a highly specific concept that subsumes them both. The semantic similarity between two clusters C_1 and C_2 is not about the clusters themselves. We define $sim(C_1, C_2)$ to be the similarity between x_1 and x_2 if all we know about x_1 and x_2 is that $x_1 \in C_1$ and $x_2 \in C_2$.

The two statements $x_1 \in C_1$ and $x_2 \in C_2$ are independent because the selection of a generic C_1 is not related to the selection of a generic C_2. The amount of information contained in $x_1 \in C_1 \wedge x_2 \in C_2$ is

$$-\log P(C_1) - \log P(C_2) \tag{4.1}$$

where $P(C_1)$ and $P(C_2)$ are probabilities that a randomly selected object belongs to C_1 and C_2, respectively.

The commonality between x_1 and x_2 is $x_1 \in C_0 \wedge x_2 \in C_0$, where C_0 is the most specific cluster that subsumes both C_1 and C_2. Therefore, (Lin, 1998) proposes the similarity measure based on information content in Equation (4.2).

$$simLin(x_1, x_2) = \frac{2 \times \log P(C_0)}{\log P(C_1) + \log P(C_2)} \tag{4.2}$$

As an example, the similarity between the concepts of *hill* and *coast* would be:

$$simLin(hill, coast) = \frac{2 \times \log P(geological - formation)}{\log P(hill) + \log P(coast)} = 0.59$$

based on Figure 4.1 which is a fragment of the WordNet. The number attached to each node C is $P(C)$.

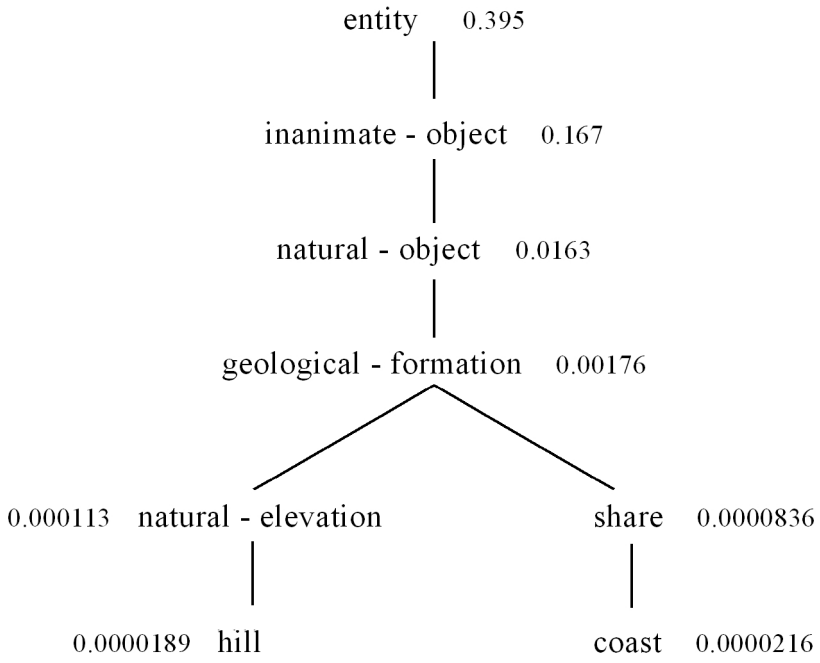

Figure 4.1: A fragment of WordNet

By adapting Lin's (1998) similarity measure to our case, we obtain the similarity between the concepts C_{305} and C_{306} of our hierarchical tree for the domain of Economy as follows,

$$sim(C_{305}, C_{306}) = \frac{2 \times \log P(\frac{10}{2843})}{\log P(\frac{5}{2843}) + \log P(\frac{6}{2843})} = 0.9$$

based on a fragment of our knowledge base presented in Figure 4.2.

In particular, the number attached to each node C is $P(C)$ and 2843 is the total number of distinct words in our concept hierarchy.

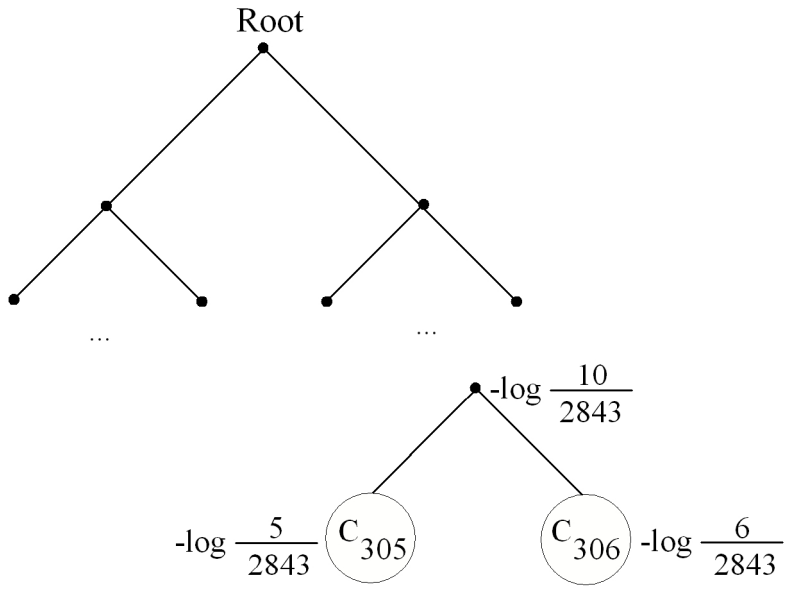

Figure 4.2: A fragment of the knowledge base

The considered clusters are represented in the next form

Node 111110100 (leaf) : {life, effort, stability, steps, negotiations}

Node 111110101 (leaf) : {steps, restructure, corporations, abuse, interests, ministers}

The cluster which subsumes this cluster is the node

Node 11111010 (d=0.942) : {life, effort, stability, steps, restructure, corporations, abuse, interests, ministers, negotiations}

The average of all semantic similarities between all the concepts define a relatedness criterium in order to assign a given word to a given chain. Two words are semantically related if the semantic similarity between two clusters is greater than a given threshold based on this average. We will see the definition of this threshold in the following section.

4.2 Lexical Chainer Process

Lexical Chains are sets of lexical items, which are conceptually related to each other and propose the content of a document by its lexical cohesion.

Morris and Hirst (1991) suggested to create Lexical Chains from conceptually similar lexical items in order to determine the discourse structure of texts.
In this thesis, we propose to use our lexico-semantic knowledge base to build Lexical Chains.

4.2.1 Preprocessing

To start the lexical chaining process, the input text is first formatted to filter the header tags and extract the textual information. The input text, free of header tags, is then tokenized to separate each word into individual tokens, with the aim to select the words for the construction of the Lexical Chains. In order to select candidate words for the chaining process we follow the same method used in the knowledge base construction. We consider nouns, compound nouns and proper nouns as candidate words to compute Lexical Chains.

4.2.2 The Algorithm

Our chaining algorithm is based on both approaches of Barzilay and Elhadad's (1997) and Hirst and St-Onge (1997).
So, our chaining model is developed according to all possible alternatives of word senses. We present our algorithm as follows:

Algorithm to compute Lexical Chains

Begin with no chain.
  for all candidate words (distinct nouns, compound nouns and proper nouns) **do**
    for all its senses **do**
      a) among present chains find the sense that is closest
        and link the new word to this chain. Remove
        unappropriated senses of the new word and the chain
        members.
      b) if no sense is close enough, start a new chain.
    end for
  end for
End.

4.3 Assignment of a word to a Lexical 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, individual*} 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 and St-Onge (1997) 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 words is inserted with all its possible senses in WordNet.

The greedy disambiguation strategy implemented in this algorithm has some limitations. Consider the word *Mr.*, belongs only to one synset, so it is disambiguated from the beginning. According to Hirst and St-Onge a chain for this word is created holding one sense:
[lex "$Mr.$", sense $\{mister,\ Mr.\}$].

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 process 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. Barzilay and Elhadad propose a chaining model according to all possible alternatives of word senses and then choose the best one among them. For 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"* and *"grammatical category of pronouns and verb forms"*. The choice of sense for *person* splits the chain world to two different interpretations. The word *machine* has 5 senses. In its first sense, *"an efficient person"*, it is related to the senses *person* and *Mr.*. If we continue the process and insert the words *"micro-computer"*, *"device"* and *"pump"*, the number of alternatives greatly increases. Under the assumption that the text is cohesive, they define the best interpretation as the one with the most connections (edges in the graph).

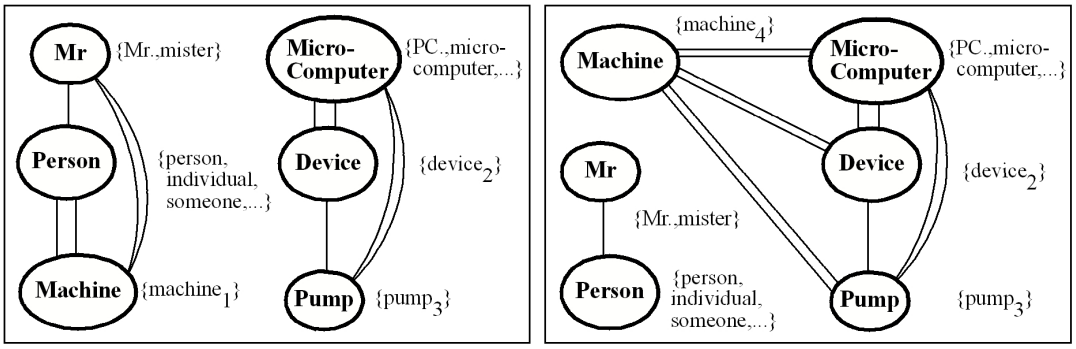

Figure 4.3: Possible interpretations.

In this case, the second interpretation is selected, which predicts the right sense for *machine*. (See Figure (4.3)).

This approach has some limitations too. Consider the word *computer science*. This word is considered in many contexts, like *Neural Networks* and also in *Genetic Algorithms*. According to Barzilay and Elhadad the word *computer science* can only be inserted in one chain although it could contribute to both chains.

To solve these limitations we propose a chaining algorithm as a compromise between these two approaches: Hirst and St-Onge (1997) and Barzilay and Elhadad (1997).

4.3.1 Threshold

The relatedness criterium that we take into account to build our Lexical Chains is the relationships outlined by the average semantic similarity between all the concepts. In order to assign a given word to a given chain we first calculate the average of all the semantic similarities related to a particular knowledge base. If the semantic similarity between two clusters is greater than a given threshold based on this average semantic similarity then the words are semantically related.

When the semantic similarity between two clusters satisfies this criterium then the words in the chain and the candidate word are semantically related. This relatedness criterium is represented by the following Equation (4.3).

$$sim(C_1, C_2) > constant \times \frac{\sum_{i=0}^{n} \sum_{j=i+1}^{n} \text{sim}(C_i, C_j)}{\frac{n^2}{2} - n} \tag{4.3}$$

The normalization of the average semantic similarity is based on the upper triangular matrix defined by these similarities defined in Figure 4.4.

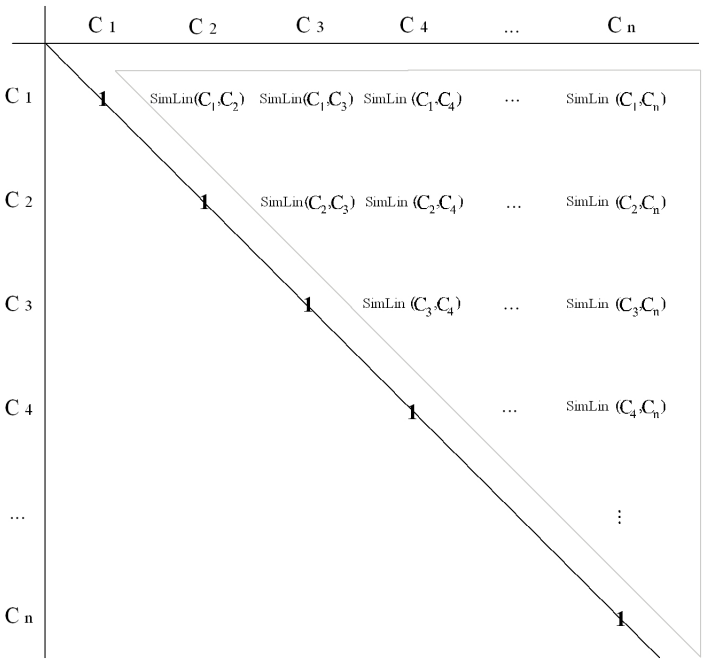

Figure 4.4: Architecture of the lexical chainer system.

4.3.2 Similarity between clusters

According to Barzilay and Elhadad, 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.

The example below illustrates this method. First, a node for *crisis* is created with its "sense" ($Cluster_{31}$). The next candidate word is *recession* which has two "senses"

($Cluster_{29}$ and $Cluster_{34}$). Considering the constant parameter equal to 0.81 (considering the constant 5 and the average semantic similarity 0.12) and the relatedness criterion for membership to a chain, we have the following similarities:

$$sim(C_{31}, C_{29}) = 0.87$$

$$sim(C_{31}, C_{34}) = 0.82$$

The choice of the sense for *recession* splits the chain word into two different interpretations as shown in Figure 4.5, as both similarities overtake the given threshold 0.81.

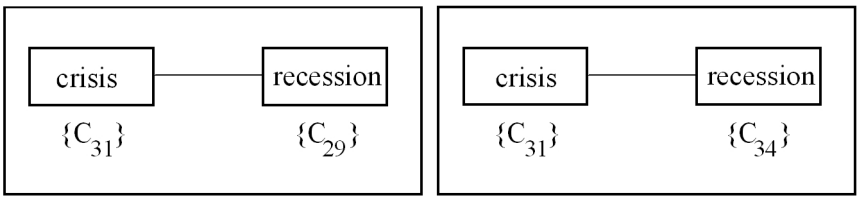

Figure 4.5: Interpretations 1 and 2.

The next candidate word *trouble* has also "two senses" ($Cluster_{29}$ and $Cluster_{32}$). All the words in a chain influence each other in the selection on their respective senses.

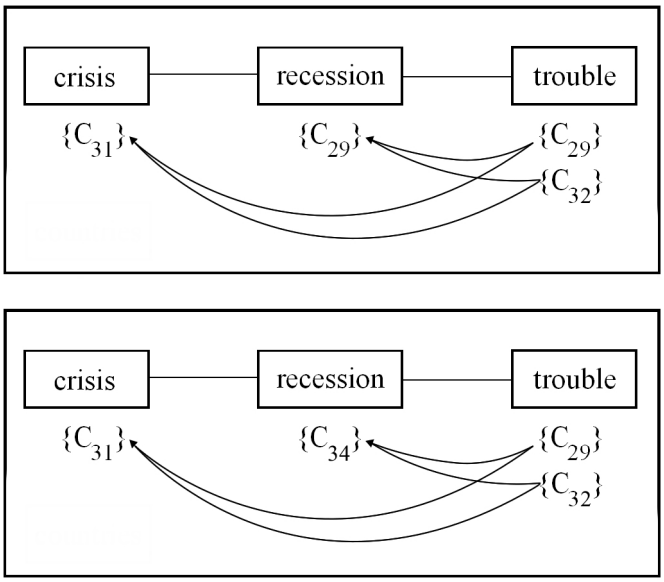

Figure 4.6: Selection of senses.

In that way, three cases can happen: all similarities overtake the threshold and we must consider both representations; only the similarities related to one representation overtake the threshold and we only consider this representation or none similarities overtake the threshold and we do not consider any representation and create a new lexical chain. We show these situations in the following example with respectively similarities for the two interpretations.

In particular, interpretation 1 had the following similarities

$$sim(C_{31}, C_{29}) = 0.87$$

$$sim(C_{31}, C_{32}) = 0.75$$

$$sim(C_{29}, C_{29}) = 1.0$$

$$sim(C_{29}, C_{32}) = 0.78$$

The sum of all these similarities is:

$$sumInt1 = sim(C_{31}, C_{29}) + sim(C_{31}, C_{32}) + sim(C_{29}, C_{29}) + sim(C_{29}, C_{32}) = 3.4$$

$$3.4 \div 4 = 0.85 > 0.81$$

The interpretation 2 had the following similarities

$$sim(C_{31}, C_{29}) = 0.87$$

$$sim(C_{31}, C_{32}) = 0.75$$

$$sim(C_{34}, C_{29}) = 0.54$$

$$sim(C_{34}, C_{32}) = 0.55$$

The sum of all these similarities is:

$$sumInt2 = sim(C_{31}, C_{29}) + sim(C_{31}, C_{32}) + sim(C_{34}, C_{29}) + sim(C_{34}, C_{32}) = 2.71$$

$$2.71 \div 4 = 0.68 \not> 0.81$$

The word *trouble* is inserted in the chain with the appropriate sense $(Cluster_{29})$ as it maximizes the overall similarity of the chain. The chain members are then updated. In this example, the interpretation with $(Cluster_{32})$ is discarded as is the cluster $(Cluster_{34})$ for *recession*. This processing is described in Figure 4.7.

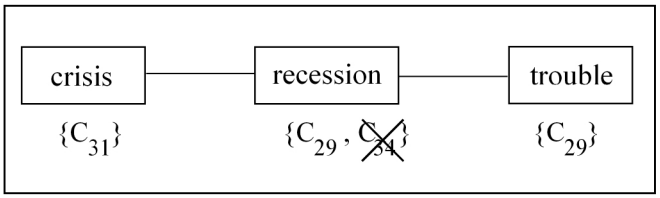

Figure 4.7: Selection of appropriate senses.

The next candidate word is *countries* which has the sense $Cluster_{300}$. Considering the relatedness criterion for membership to a chain

$$sim(C_{31}, C_{300}) = 0.11$$

$$sim(C_{29}, C_{300}) = 0.12$$

we conclude that this candidate word is not related to any word in the first chain, so we create a new chain for it with a single interpretation.

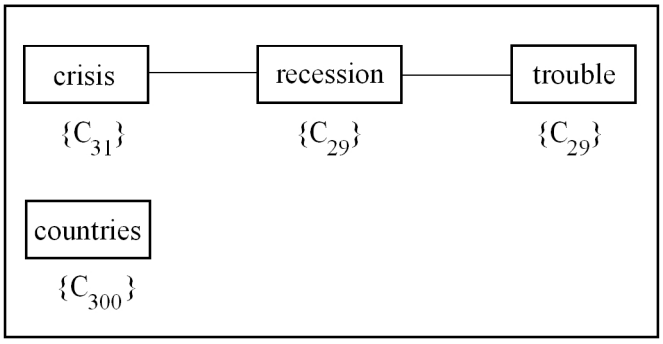

Figure 4.8: Creation of new chain.

Words read from the text are inserted into the chains, for which they maximize the internal semantic similarity of the chain. If no chain is found, then a new chain is created

with all the senses of the word. Like in Barzilay and Elhadad (1997) there are occurrences of a word belonging to more than one chain. This is not the case for Hirst and St-Onge (1997). However, our disambiguation process is done like Hirst and St-Onge (1997). That's why our algorithm is a mixture of both algorithms.

4.4 Score of a chain

Once chains are computed, the high-scoring ones must be picked as representing the important concepts of the original document. Not all the output chains represent topic concepts of the text. Therefore, one must first identify the strongest chains.

Like in Barzilay and Elhadad (1997), we define a chain score. However, our scoring function differs from their work and is defined in Equation (4.4).

$$score(chain) = \frac{\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \text{simLin}(w_i, w_j)}{\frac{(n-1)n}{2}} \tag{4.4}$$

As all chains will be scored, the ones with higher scores will be extracted. Of course, a threshold will have to be defined by the user. In the next chapter, we will show the five most important chains for each domain for the best parameters.

Chapter 5

Evaluating the Lexical Chainer

The evaluation of Lexical Chains is generally difficult. Even if they can be effectively used in many practical applications like Automatic Summarization, Topic Segmentation, and others, Lexical Chains are seldom desirable outputs in a real-world application, and it is unclear how to assess their quality independently of the underlying application in which they are used. For example, in Summarization, it is hard to determine whether a good or bad performance comes from the efficiency of the lexical chaining algorithm or from the appropriateness of using Lexical Chains in that kind of application. In this chapter, we evaluate Lexical Chains generated by our system from two methods broadly classified in two categories: quantitative and qualitative.

The data that was used for evaluation consists of a corpus extracted from DUC 2004 from four different domains: Economy, Politics, Sport and War. These texts are the same used in the construction of the knowledge base but, Lexical Chains could be created even for documents which are not *a priori* part of the collection.

5.1 Quantitative Results

5.1.1 Economy

These experimental results take into account the relatedness criterium and a threshold combining four different parameters and the average of all the semantic similarities related

to the particular knowledge base for this domain. The following results were obtained with $N = 10$ for the word-context vector length and 20 for the window context size to calculate the SCP.

The five texts of this domain have the following characteristics:

	# Words	#Distinct Words	#Distinct Nouns
Document 1	4606	1753	703
Document 2	8136	2830	1117
Document 3	4518	1712	664
Document 4	9083	2386	927
Document 5	6327	1956	759

Lexical Chains per document

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	30	48	69	95
Document 2	37	61	103	154
Document 3	33	58	88	136
Document 4	44	69	115	180
Document 5	46	70	129	205

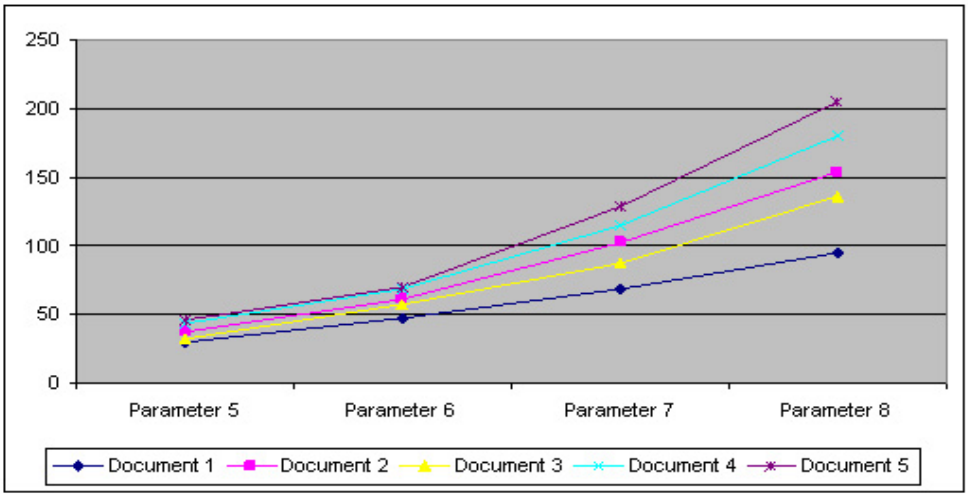

Figure 5.1: # Lexical Chains per Economy document.

Average strength of Lexical Chains

Per document the average strength of Lexical Chains is obtained by the following Equation (5.1)

$$AvgStrengthLC = \frac{\sum_{i=1}^{n} strengthLC_i}{n} \tag{5.1}$$

where n is the number of Lexical Chains in the document and $strengthLC_i$ the number of chain members in the Lexical Chain i.

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	125	77	47	30
Document 2	222	116	42	17
Document 3	95	58	26	14
Document 4	153	58	22	9
Document 5	162	44	17	8

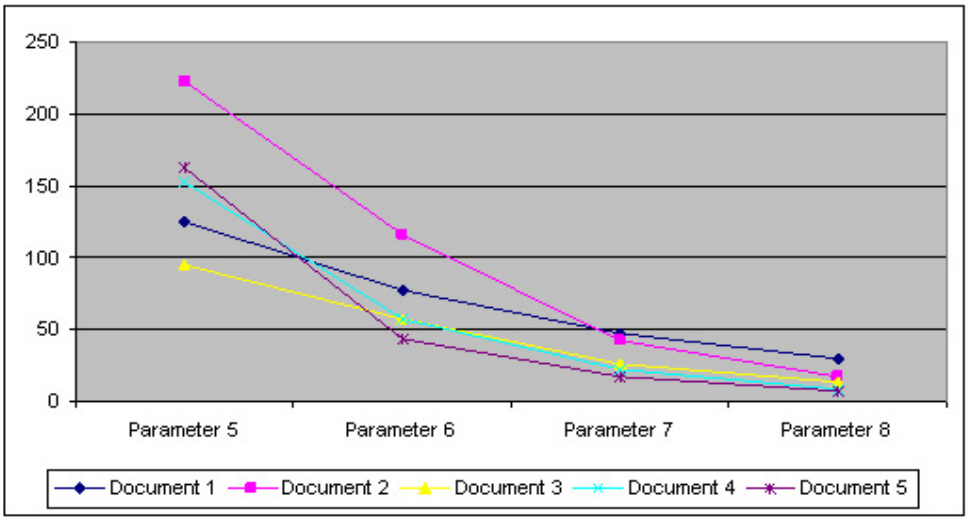

Figure 5.2: Average strength of Lexical Chains per Economy document.

Average number of clusters per Lexical Chain

Per document the average number of clusters per Lexical Chain is obtained by the following Equation (5.2)

$$AvgClustersLC = \frac{\sum_{i=1}^{n} clustersLC_i}{n} \tag{5.2}$$

where n is the number of Lexical Chains in the document and $clustersLC_i$ the number of different clusters in the Lexical Chain i.

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	29	17	11	6
Document 2	52	27	9	4
Document 3	25	13	6	3
Document 4	47	16	5	2
Document 5	55	15	5	2

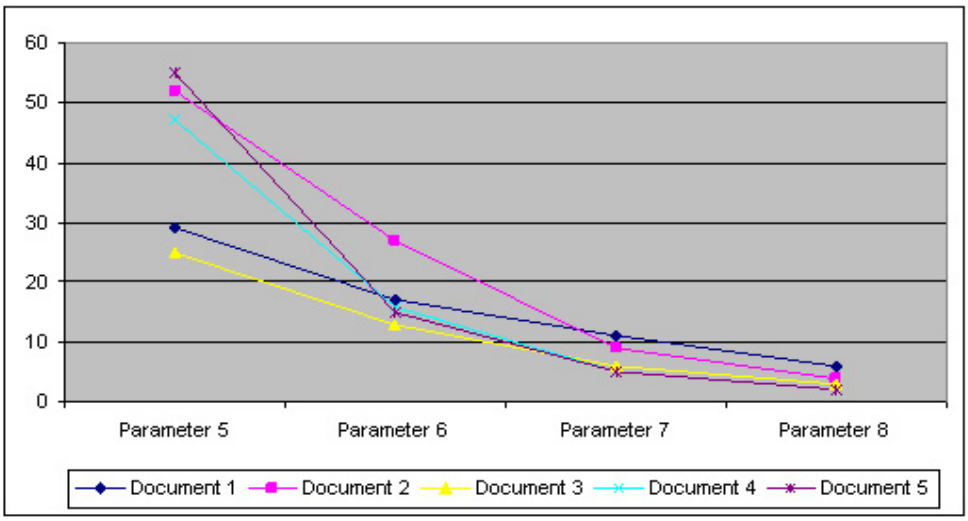

Figure 5.3: Average number of clusters per Lexical Chain per Economy document.

5.1.2 Politics

These experimental results take into account the relatedness criterium and a threshold combining four different parameters and the average of all the semantic similarities related to the particular knowledge base for this domain. The following results were obtained with $N = 10$ for the word-context vector length and 20 for the window context size to calculate the SCP.

The five texts of this domain have the following characteristics:

	# Words	#Distinct Words	#Distinct Nouns
Document 1	9362	2932	1112
Document 2	3356	1085	393
Document 3	3909	1281	515
Document 4	3233	1178	439
Document 5	8066	1932	720

Lexical Chains per document

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	34	52	79	110
Document 2	29	46	72	99
Document 3	30	48	73	104
Document 4	38	58	83	115
Document 5	39	59	100	144

Average strength of Lexical Chains

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	220	117	75	48
Document 2	37	21	9	6
Document 3	61	36	25	14
Document 4	42	22	12	8
Document 5	80	44	17	10

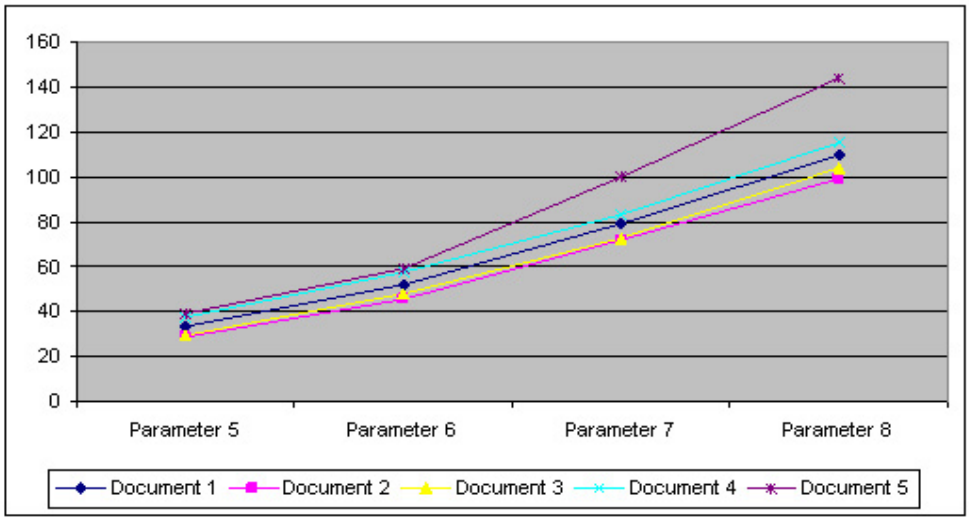

Figure 5.4: # Lexical Chains per Politics document.

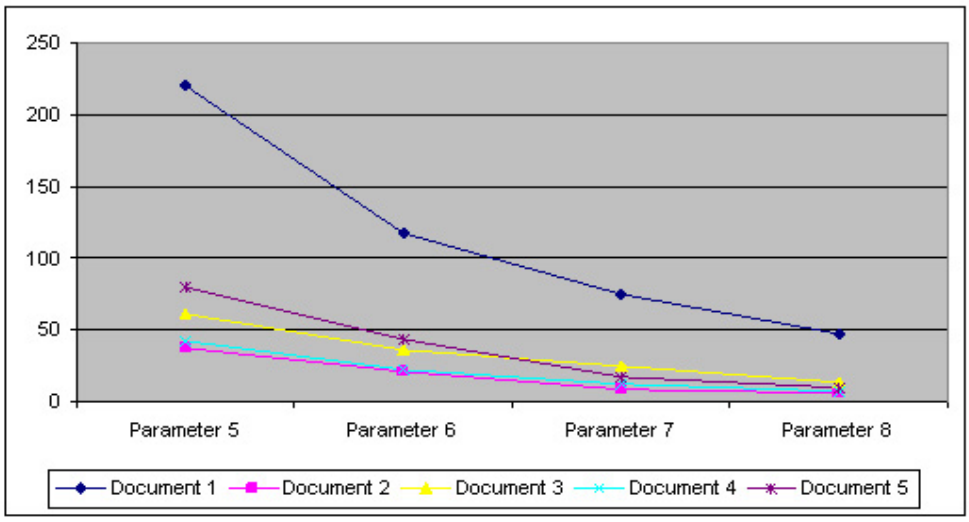

Figure 5.5: Average strength of Lexical Chains per Politics document.

Average number of clusters per Lexical Chain

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	36	19	11	7
Document 2	8	4	2	1
Document 3	15	8	5	3
Document 4	12	6	3	2
Document 5	19	11	4	2

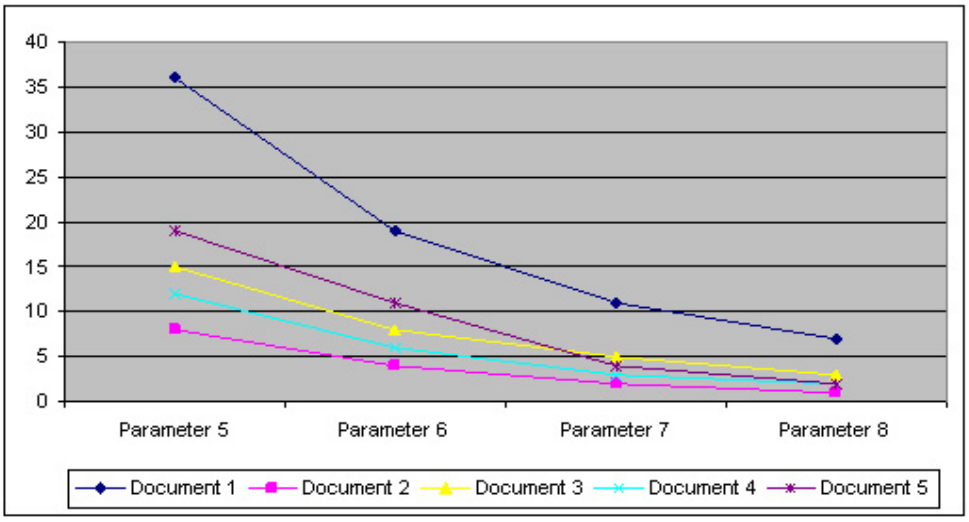

Figure 5.6: Average number of clusters per Lexical Chain per Politics document.

5.1.3 Sport

These experimental results take into account the relatedness criterium and a threshold combining four different parameters and the average of all the semantic similarities related to the particular knowledge base for this domain. The following results were obtained with $N = 10$ for the word-context vector length and 20 for the window context size to calculate the SCP.

The four texts of this domain have the following characteristics:

	# Words	#Distinct Words	#Distinct Nouns
Document 1	8133	1956	672
Document 2	3823	1630	708
Document 3	4594	953	324
Document 4	4530	1265	431

Lexical Chains per document

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	27	43	73	73
Document 2	31	52	81	83
Document 3	28	40	51	51
Document 4	29	53	83	87

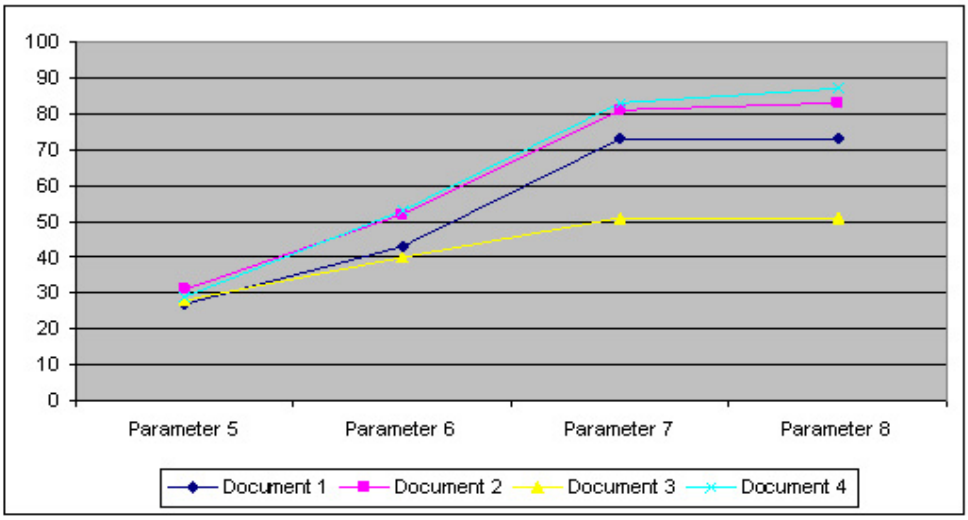

Figure 5.7: # Lexical Chains per Sport document.

Average strength of Lexical Chains

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	120	83	43	42
Document 2	185	36	17	17
Document 3	23	27	26	26
Document 4	36	25	20	19

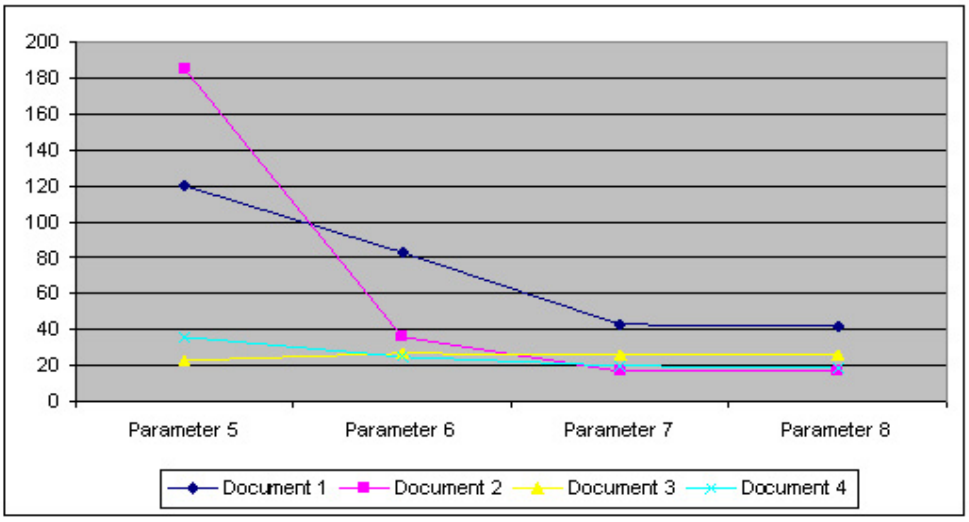

Figure 5.8: Average strength of Lexical Chains per Sport document.

Average number of clusters per Lexical Chain

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	19	13	7	7
Document 2	13	6	3	3
Document 3	3	4	4	4
Document 4	6	4	3	3

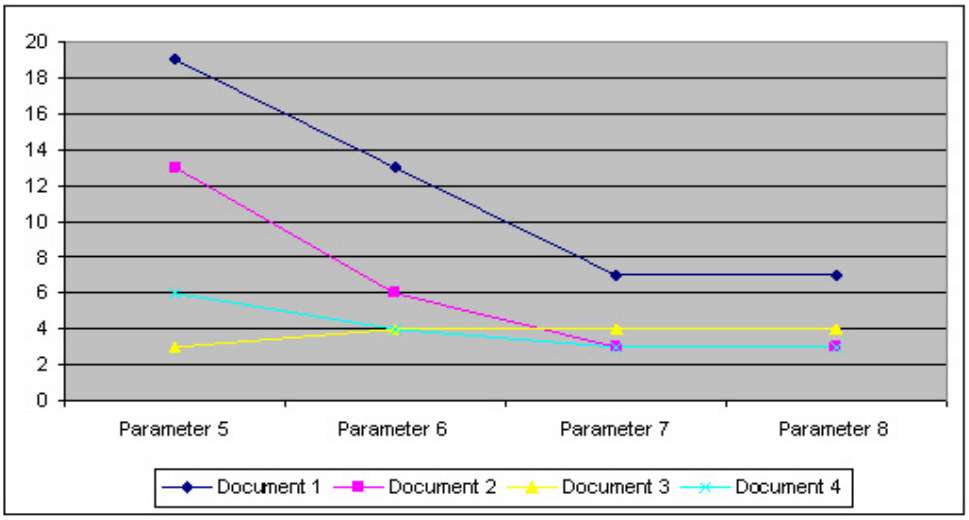

Figure 5.9: Average number of clusters per Lexical Chain per Sport document.

5.1.4 War

These experimental results take into account the relatedness criterium and a threshold combining four different parameters and the average of all the semantic similarities related to the particular knowledge base for this domain. The following results were obtained with $N = 10$ for the word-context vector length and 20 for the window context size to calculate the SCP.

The five texts of this domain have the following characteristics:

	# Words	#Distinct Words	#Distinct Nouns
Document 1	3950	1285	551
Document 2	6882	1541	580
Document 3	3237	1121	468
Document 4	5435	1837	725
Document 5	3215	881	345

Lexical Chains per document of War

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	30	39	59	64
Document 2	34	56	82	94
Document 3	35	57	84	96
Document 4	35	59	107	126
Document 5	33	48	77	84

Average strength of Lexical Chains

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	74	50	38	33
Document 2	76	45	21	18
Document 3	48	25	13	10
Document 4	105	47	21	19
Document 5	33	24	14	14

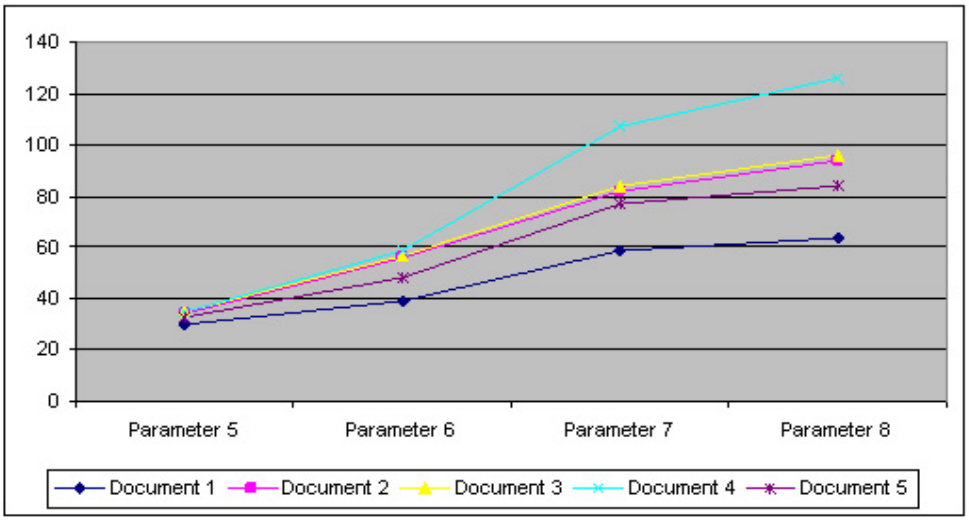

Figure 5.10: # Lexical Chains per War document.

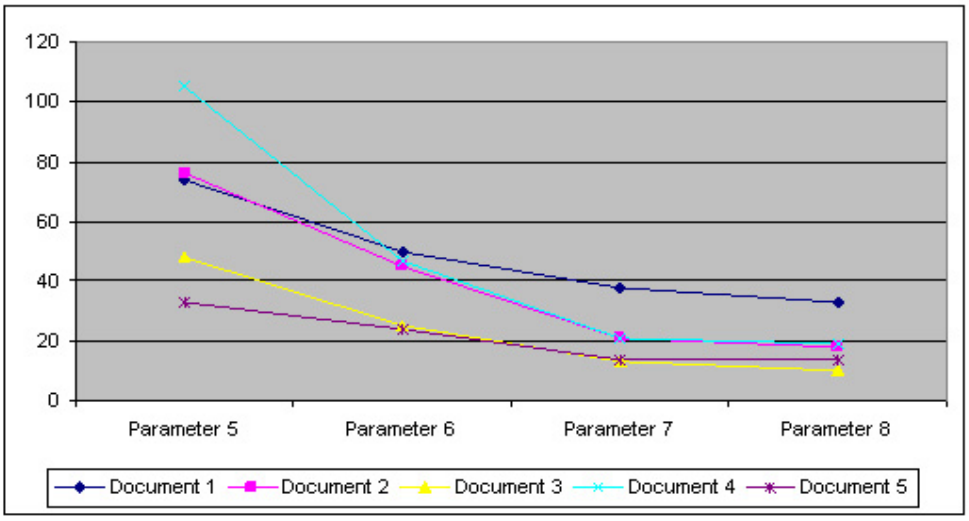

Figure 5.11: Average strength of Lexical Chains per War document.

Average number of clusters per Lexical Chain

	Parameter 5	Parameter 6	Parameter 7	Parameter 8
Document 1	12	9	6	6
Document 2	17	9	4	4
Document 3	10	5	3	2
Document 4	24	11	5	4
Document 5	6	4	2	2

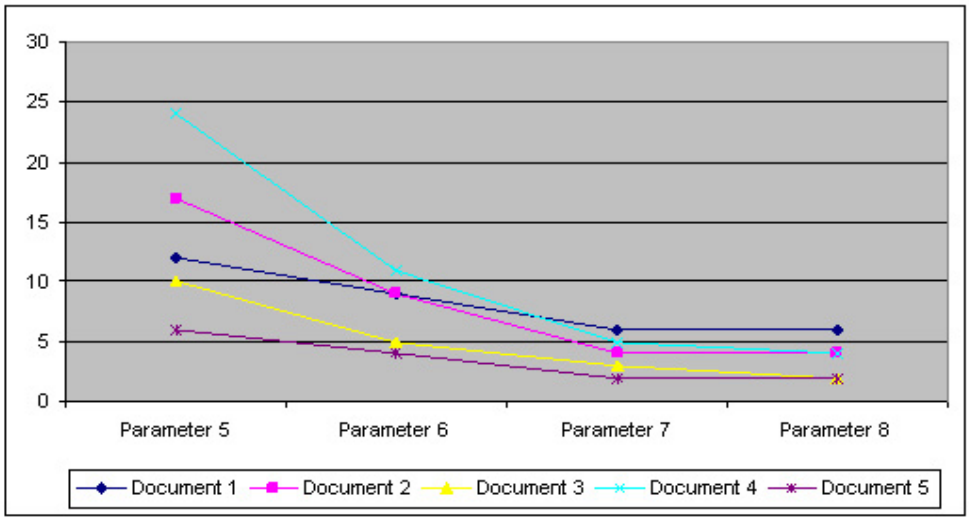

Figure 5.12: Average number of clusters per Lexical Chain per War document.

The evaluation presented here presents that the text properties are not crucial for our lexical chaining algorithm as there is no significant pattern that is common to all domains. So, we can conclude that our system well adapts to different genres/domains.
Here we present some conclusions that prove this assumption.

First conclusion, the number of Lexical Chains do not depend of the document size, type and noun distributions. In fact, we have small texts with more Lexical Chains than bigger ones. Consider for example, the Document 2 and Document 5 in the domain of Economy. The number of words and nominal units in Document 5 is less than the number of words and nominal units in Document 2. Although, we have more Lexical Chains in Document 5 than in Document 2.

Second conclusion, the strength of the Lexical Chains logically decreases with the parameter being more restrictive. However, the strength of Lexical Chains does not depend on the size of the document. For Document 2 and Document 4 in the domain of Economy we can support this conclusion. Document 4 is bigger in size but on the other side, has less nominal units than Document 2. But, we have more Lexical Chains in Document 4 than in Document 2 and these Lexical Chains are smaller than the Lexical Chains of Document 2.

Third conclusion, is that our algorithm does not gather words that belong to only one cluster and take advantage of the automatically built lexico-semantic knowledge base. This is illustrated by the average number of clusters per chain. However, it is obvious that by increasing the parameter the words in a chain tend to belong to only one cluster as it is the case for the best Lexical Chains with parameter equal to 8.

5.2 Qualitative Results

Once chains are computed, the high-scoring ones must be picked to represent the important concepts from the original document.

Here, we present the five high-scoring chains of each domain but only for the best threshold (the threshold combining parameter equal to 7 and 8 and the average of all the semantic similarities) and the Lexical Chains with more than 4 words.

5.2.1 Economy

Document 1

Parameter 7
- Lexical Chain 37 with a strength 6, 2 clusters and a score equal to 1.25: {countries, supply, analysts, economies, agreement, aims}
- Lexical Chain 28 with a strength 7, 1 cluster and a score equal to 1.24: {running, being, market, help, back, unrest, protesters}
- Lexical Chain 8 with a strength 18, 2 clusters and a score equal to 1.04: {Sunday, interview, investments, others, dollar, tailspin, yen, year's, ends, report, Deputy, corruption and sexual misconduct, warning, Monday's, edition, Time, magazine, aggressiveness}
- Lexical Chain 56 with a strength 21, 2 clusters and a score equal to 1.04: {issue, argument, liquidity, crunch, K.S, Jomo, professor, University, Malaya, timing, imposition, currencies, order, population, Things, Chandra, Muzaffar, science, question, leadership, Top}
- Lexical Chain 48 with a strength 11, 2 clusters and a score equal to 1.03: {investment, consumer, control, exchange, Analysts, disaster, review, progress, Framework, agreement, aims}

Parameter 8
- Lexical Chain 35 with a strength 11, 1 cluster and a score equal to 1.0: { trade, barriers, industries, products, services, fisheries, forestry, goods, telecommunications, accounts, half} - Lexical Chain 36 with a strength 7, 1 cluster and a score equal to 1.0: {running, being, market, help, back, unrest, protesters} - Lexical Chain 75 with a strength 16, 1 cluster and a score equal to 1.0: {flows, collapse, hedge fund, consequences, Long Term, Capital, Management, Greenwich, Conn, series, bets, securities, collapse, flight, fiasco, week's} - Lexical Chain 89 with a strength 7, 1 cluster and a score equal to 1.0: {risks, mention, Russia's, need, programs, worst hit, Fischer} - Lexical Chain 9 with a strength 14, 1 cluster and a score equal to 0.87: {Sunday, interview, yen, year's, ends, report, Deputy, corruption and sexual misconduct, warning, Monday's, edition, Time, magazine, aggressiveness}

Document 2

Parameter 7
- Lexical Chain 69 with a strength 26, 3 clusters and a score equal to 1.26: {governments, Sony, TDK, earnings, Pelham, Smithers, Ltd, survey, businesses, level, samplings, season, rubles, paychecks, cropsand, insult, Tolstoy, Chekhov, letter, labor, love, health, Baba, Anya, Omsk, regions} - Lexical Chain 40 with a strength 14, 2 clusters and a score equal to 1.24: {Boris, IMF, logic, willingness, fund's, Africa, story, demands, lengths, tripling, tensions, displeasure, gaps, bail} - Lexical Chain 15 with a strength 14, 1 cluster and a score equal to 1.0: {Russians, dollars, hedge, stupidity, Otto, Latsis, commentator, Noviye, Izvestiya, People, Yegor, Gaidar, Russia, flow} - Lexical Chain 53 with a strength 4, 1 cluster and a score equal to 1.0: {meltdown, depression, poverty, line} - Lexical Chain 99 with a strength 9, 2 clusters and a score equal to 1.0: {Vadim, Gustov, quarter, property, growth of production, effectiveness, renewal, privatization, ITAR Tass}

Parameter 8
- Lexical Chain 99 with a strength 5, 1 cluster and a score equal to 1.1: {crises, picture, dissension, Defense, Departments} - Lexical Chain 138 with a strength 18, 2 clusters and a score equal to 1.01: {recent days, press, rift, idea, sales, Yushchenko, consultations, hryvna, Valery, Pustovoitenko, ways, expectations, Ukrainians, clothing, household, deposits, authors, Government} - Lexical Chain 65 with a strength 9, 1 cluster and a score equal to 1.0: {flows, flight, collapse, Capital, Greenwich, Conn, securities, Long Term, Management} - Lexical Chain 70 with a strength 14, 1 cluster and a score equal to 1.0: {Moscow, children, twin, inability, businessmen, contrary, view, Treasury's, lap, dog, Ask, response, ventures, facts} - Lexical Chain 146 with a strength 9, 2 clusters and a score equal to 1.0: {Vadim, Gustov, quarter, property, growth of production, effectiveness, renewal, privatization, ITAR Tass}

Document 3

Parameter 7
- Lexical Chain 10 with a strength 6, 2 clusters and a score equal to 1.56: {steps, effort, days, corporations, agreement, ministers} - Lexical Chain 85 with a strength 8, 1 cluster and a score equal to 1.22: {pressures, currency's, dealings, People's, Administration, News, Agency, opening} - Lexical Chain 29 with a strength 10, 3 clusters and a score equal to 1.0: {countries, analysts, consolidation, development, reduction, economies, start, spring, agreement, ministers} - Lexical Chain 44 with a strength 9, 2 clusters and a score equal to 1.0: {survey, labor, level, governments, impact, reaction, slowdown, businesses, percentage} - Lexical Chain 87 with a strength 7, 1 cluster and a score equal to 1.0: {drive, ruins, dates, history, Finance Minister Dominique Strauss Kahn, Info, era}

Parameter 8
- Lexical Chain 125 with a strength 8, 1 cluster and a score equal to 1.22: {pressures, currency's, dealings, People's, Administration, News, Agency, opening} - Lexical Chain 56 with a strength 17, 2 clusters and a score equal to 1.01: {Duisenberg, improvement, euro zone's, event, Bank's, step, comprises, six member, Growth, financial markets, damage, years, job, term, Le Monde, Jean Claude, Trichet} - Lexical Chain 23 with a strength 7, 1 cluster and a score equal to 1.0: {timing, liquidity, issue, population, professor, University, question} - Lexical Chain 37 with a strength 14, 2 clusters and a score equal to 1.0: {Britain, Denmark, Sweden, Greece, concentration, confidence, new currency, thing, voters, Maastricht, treaty, opt out, supporters, referendum} - Lexical Chain 134 with a strength 7, 1 cluster and a score equal to 1.0: {drive, ruins, dates, history, Finance Minister Dominique Strauss Kahn, Info, era}

Document 4

Parameter 7
- Lexical Chain 1 with a strength 8, 1 cluster and a score equal to 1.0: {discussions, Trade, Commission, Petroleum's, purchase of Amoco, worth, scrutiny, problem} - Lexical Chain 110 with a strength 30, 3 clusters and a score equal to 0.99: {plant, billionaire, board, Beaumont, brick, Union leaders, industry's, humming, chemistry, Neches, River, Gulf, provider, livelihoods, spray, Spindletop, gusher, overtime, third generation, employee, nephew, dawn, justification, tasks, relief, favor, vacations, sickness, retirement, ethos} - Lexical Chain 83 with a strength 11, 2 clusters and a score equal to 0.95: {investors, Amoco,highs, Poor's, Nasdaq, indexes, anticipation, rush, equity, assurance, stripes} - Lexical Chain 78 with a strength 29, 2 clusters and a score equal to 0.94: {Chevron, Exxon Mobil combination, Ford, Motors, Macy's, Gimbels, Axinn, lawyer, Texaco, Getty, purview, vigor, Staples, Office, discounters, wholesalers, Boeing's, McDonnell, Douglas, backs, background, Shell Texaco, Eleanor, Law, expert, Terry, Calvani, commissioner, aftermath}

- Lexical Chain 2 with a strength 6, 2 clusters and a score equal to 0.92: {business, energy, services, telecommunications, industries, products}

Parameter 8

- Lexical Chain 1 with a strength 8, 1 cluster and a score equal to 1.0: {discussions, Trade, Commission, Petroleum's, purchase of Amoco, worth, scrutiny, problem}
- Lexical Chain 69 with a strength 4, 1 cluster and a score equal to 1.0: {services, telecommunications, industries, products}
- Lexical Chain 70 with a strength 6, 1 cluster and a score equal to 1.0: {technology, stocks, Dow, average, points, profits}
- Lexical Chain 72 with a strength 5, 1 cluster and a score equal to 1.0: {order, issue, question, professor, University}
- Lexical Chain 90 with a strength 14, 1 cluster and a score equal to 1.0: {mergers, stock market, recognition, April's, peak, volume, witness, Tom, Burnett, Merger, Insight, certainty, crude oil prices, care}

Document 5

Parameter 7

- Lexical Chain 88 with a strength 18, 4 clusters and a score equal to 1.1: {sign, chance, Rio, Janeiro, Grande, Sul, uphill, promise, hospitals, powerhouse, success, inhabitants, victory, pad, presidency, contingent, exit, legislature}
- Lexical Chain 50 with a strength 8, 1 cluster and a score equal to 1.0: {transactions, taxes, Stabilization, spate, fuel, income, fortunes, means}
- Lexical Chain 77 with a strength 5, 1 cluster and a score equal to 1.0: {proposal, factory, owners, Fund, Rubin's}
- Lexical Chain 126 with a strength 4, 1 cluster and a score equal to 1.0: {disaster, control, investment, review}
- Lexical Chain 12 with a strength 7, 2 clusters and a score equal to 0.99: {issue, order, University, population, question, timing, currencies}

Parameter 8
- Lexical Chain 13 with a strength 6, 1 cluster and a score equal to 1.0: {issue, order, University, population, question, timing} - Lexical Chain 42 with a strength 4, 1 cluster and a score equal to 1.0: {reforms, International Monetary Fund, resistance, credit} - Lexical Chain 52 with a strength 4, 1 cluster and a score equal to 1.0: {growth, head, unemployment, boost} - Lexical Chain 55 with a strength 8, 1 cluster and a score equal to 1.0: {transactions, taxes, Stabilization, spate, fuel, income, fortunes, means} - Lexical Chain 95 with a strength 6, 1 cluster and a score equal to 1.0: {products, trade, goods, industries, telecommunications, accounts}

5.2.2 Politics

Document 1

Parameter 7
- Lexical Chain 73 with a strength 4, 1 cluster and a score equal to 1.45: {society, cast, extremists, orientation} - Lexical Chain 24 with a strength 9, 3 clusters and a score equal to 1.4: {Judiciary Committee, removal, November, offenses, inquiry, hour, Christopher, Shays, Georgia} - Lexical Chain 30 with a strength 6, 2 clusters and a score equal to 1.27: {years, decades, drawing, act, chaos, increases} - Lexical Chain 34 with a strength 16, 2 clusters and a score equal to 1.14: {administration, fact, will, whole, series, comebacks, nation's, Richard, Nixon, respectability, angst, story, accuracy, culture, criminalization, indifference} - Lexical Chain 26 with a strength 13, 2 clusters and a score equal to 1.08: {Bob Livingston, time, votes, thing, being, David, Obey, Appropriations, Committee, interview, repeat, hell, souls}

Parameter 8
- Lexical Chain 12 with a strength 5, 1 cluster and a score equal to 1.0: {ovation, officer, announcement, fight, constitutional process}
- Lexical Chain 37 with a strength 5, 1 cluster and a score equal to 1.0: {members of Congress, weapons, Others, Pentagon, flourishes}
- Lexical Chain 48 with a strength 4, 1 cluster and a score equal to 1.0: {Iraq, attack, public opinion polls, Americans}
- Lexical Chain 73 with a strength 9, 1 cluster and a score equal to 1.0: {Tennessee, course, states, communications, economy, agriculture, commitments, politics, child}
- Lexical Chain 101 with a strength 12, 1 cluster and a score equal to 1.0: {drag, defiance, contempt, voters, Alan, Simpson, senator, Wyoming, span, movie, stock, Robert}

Document 2

Parameter 7
- Lexical Chain 37 with a strength 5, 1 cluster and a score equal to 1.0: {hard line, faction, Cossuta, minute, wing}
- Lexical Chain 34 with a strength 5, 2 clusters and a score equal to 0.96: {Senate, negotiations, end, interest, Washington}
- Lexical Chain 2 with a strength 7, 2 clusters and a score equal to 0.8: {party, week, leader, Friday, efforts, member, head}
- Lexical Chain 65 with a strength 12, 2 clusters and a score equal to 0.93: {Silvio, Liotta, Foreign Minister Lamberto, Dini, heads, names, Treasury, Carlo, Azeglio, Ciampi, technicians, period,}
- Lexical Chain 46 with a strength 19, 3 clusters and a score equal to 0.70: {opposition, majority, Communists, participation, coherence, instability, constitution, Sergio, diplomat, science, professor, credibility, thread, factions, leftists, fringe, fragile, beginning, intact}

Parameter 8
- Lexical Chain 1 with a strength 39, 1 cluster and a score equal to 1.0: {government, President, Luigi, Scalfaro, vote in Parliament, year, cuts, majority, risk, shrug, view, Bertinotti, choices, rescue, cent, invitations, common currency, early elections, Europe's, common currency the euro, outset, Albania, concessions, premier, hardline, stance, Politicians, finances, options, dozens, economist, patchwork, quilt, pact, doors, swing, technocrats, creation, launch} - Lexical Chain 13 with a strength 8, 1 cluster and a score equal to 1.0: {Refounding's, hard line leftists, centrist group, June, Francesco Cossiga, rank and file, NATO expansion, Cossiga's} - Lexical Chain 31 with a strength 4, 1 cluster and a score equal to 1.0: {cut, address, taxes, deficit} - Lexical Chain 44 with a strength 9, 1 cluster and a score equal to 1.0: {Armando, crisis, Cossutta, weekend, gymnastics, break, speculation, moderate wing, December} - Lexical Chain 61 with a strength 4, 1 cluster and a score equal to 1.0: {negotiations, end, interest, Washington}

Document 3

Parameter 7
- Lexical Chain 5 with a strength 4, 1 cluster and a score equal to 1.0: {report, leaders, lives, information} - Lexical Chain 33 with a strength 4, 1 cluster and a score equal to 1.0: {past, attention, defenders, investigations} - Lexical Chain 28 with a strength 19, 2 clusters and a score equal to 0.95: {investigators, hospital, ward, wounds, neck, description, fashion, suspects, raids, assault, rifles, door, further details, surgery, service, detective, Igor, Kozhevnikov, Ministry} - Lexical Chain 40 with a strength 4, 2 clusters and a score equal to 0.92: {security, times, weeks, fire}

- Lexical Chain 24 with a strength 31, 3 clusters and a score equal to 0.85: {enemies, Choice, stairwell, assailants, woman, attackers, entrance, car, guns, Friends, relatives, Mrs. Staravoitova, founder, movement, well thought, Sergei, Kozyrev, Association, Societies, supporter, Stalin's, council, criminals, Yegor, Gaidar, minister, ally, suggestions, measures, smile, commitment}

Parameter 8

- Lexical Chain 29 with a strength 4, 1 cluster and a score equal to 1.0: {enemies, Choice, smile, commitment}

- Lexical Chain 33 with a strength 15, 1 cluster and a score equal to 1.0: {investigators, hospital, neck, description, suspects, raids, assault, rifles, door, further details, surgery, detective, Igor, Kozhevnikov, Ministry}

- Lexical Chain 41 with a strength 4, 1 cluster and a score equal to 1.0: {past, attention, defenders, investigations}

- Lexical Chain 47 with a strength 19, 1 cluster and a score equal to 1.0: {stairwell, assailants, woman, attackers, entrance, car, guns, Friends, relatives, Mrs Staravoitova, founder, movement, well thought, Sergei, Kozyrev, Association, Societies, supporter, Stalin's}

- Lexical Chain 74 with a strength 13, 1 cluster and a score equal to 1.0: {city's, Marble, Hall, Ethnography Museum, casket, ribbons, Alexander Nevsky Monastery, composers, Tchaikovsky, Modest, Mussorgsky, author, Dostoevsky}

Document 4

Parameter 7
- Lexical Chain 55 with a strength 8, 2 clusters and a score equal to 1.17: {reform, date, Italy, consensus, system, Jan, constitution, line}
- Lexical Chain 16 with a strength 6, 2 clusters and a score equal to 0.95: {offer, million, report, tenure, reference, leaders}
- Lexical Chain 60 with a strength 8, 2 clusters and a score equal to 0.94: {policy making, officers, priorities, emerges, fundamentalism, Kurdish nationalism, apparatus, Ecevit's}
- Lexical Chain 54 with a strength 16, 2 clusters and a score equal to 0.79: {secular traditions, crackdown, Istanbul, literature, American run, school, courses, Harvard, University, journalist, People's, body, Ilter, Turan, perspective, talks}
- Lexical Chain 25 with a strength 8, 2 clusters and a score equal to 0.70: {hands, debate, way, duty, role, scandal, country, members}

Parameter 8
- Lexical Chain 9 with a strength , 1 cluster and a score equal to 1.0: {government, premier, year, majority, President}
- Lexical Chain 23 with a strength 5, 1 cluster and a score equal to 1.0: {million, report, tenure, reference, leaders}
- Lexical Chain 70 with a strength 14, 1 cluster and a score equal to 1.0: {Bulent Ecevit, reading, tastes, poetry, journals, York, Review, Books, works, T.S, Eliot, volumes, poems, Premier designate}
- Lexical Chain 77 with a strength 4, 1 cluster and a score equal to 1.0: {policy making, officers, priorities, emerges}
- Lexical Chain 64 with a strength 15, 1 cluster and a score equal to 0.88: {secular traditions, crackdown, Istanbul, literature, American run, school, courses, Harvard, University, journalist, People's, body, Ilter, Turan, perspective}

Document 5

Parameter 7
- Lexical Chain 31 with a strength 6, 2 clusters and a score equal to 1.24: {finance, home, host, official, week, others} - Lexical Chain 61 with a strength 6, 2 clusters and a score equal to 1.24: {Senate's, senators, two thirds, discussions, request, side} - Lexical Chain 71 with a strength 7, 1 cluster and a score equal to 1.23: {departure, Florida, appearance, afternoon, computers, chance, wrapping} - Lexical Chain 81 with a strength 18, 2 clusters and a score equal to 1.23: {hour, inquiry, March, Dick, Supreme Court, review, court, rulings, November, Carolyn, Maloney, player, realization, willingness, Christopher, Shays, need, Georgia} - Lexical Chain 90 with a strength 15, 2 clusters and a score equal to 1.02: {wouldn't, effect, populations, estimate, African-Americans, cities, blacks, Hispanics, areas, standard, means, method, living, boundaries, Monday's}

Parameter 8
- Lexical Chain 139 with a strength 10, 1 cluster and a score equal to 1.07: {type, good job, ransom, highway, robbery, caps, endgame, Gene, Taylor, D Miss} - Lexical Chain 97 with a strength 8, 1 cluster and a score equal to 1.03: {hour, inquiry, realization, player, Christopher, Shays, need, Georgia} - Lexical Chain 19 with a strength 5, 1 cluster and a score equal to 1.0: {members of Congress, schools, Others, weapons, Pentagon} - Lexical Chain 31 with a strength 9, 1 cluster and a score equal to 1.0: {Congress, dispute, proposal, office, day, airlines, zone, passengers, peanuts} - Lexical Chain 117 with a strength 11, 1 cluster and a score equal to 1.0: {African-Americans, cities, blacks, Hispanics, areas, standard, means, method, living, boundaries, Monday's}

5.2.3 Sport

Document 1

Parameter 7
- Lexical Chain 0 with a strength 6, 1 cluster and a score equal to 1.0: {prodigy, creativity, big city, profiteer, shot, play} - Lexical Chain 2 with a strength 15, 1 cluster and a score equal to 1.0: { Bird, years, Larry Bird exception, free agents, clause, Bulls, insistence, growth, elimination, amount, taxation, points, rights, example, marijuana} - Lexical Chain 14 with a strength 16, 1 cluster and a score equal to 1.0: {Patrick, loyalties, country, press, conferences, training camp, venues, arenas, gymnasiums, teammates, SUNY Purchase, N.Y, train, Oct, public, start} - Lexical Chain 59 with a strength 10, 1 cluster and a score equal to 1.0: {scale, management, middle class, surprise, Sports, entertainment, business, movies, efforts, rise} - Lexical Chain 61 with a strength 7, 1 cluster and a score equal to 1.0: {summer, clauses, lockouts, strikes, individual player contracts, support, labor law}

Parameter 8
- Lexical Chain 0 with a strength 6, 1 cluster and a score equal to 1.0: {prodigy, creativity, big city, profiteer, shot, play} - Lexical Chain 2 with a strength 15, 1 cluster and a score equal to 1.0: {Bird, years, Larry Bird exception, free agents, clause, Bulls, insistence, growth, elimination, amount, taxation, points, rights, example, marijuana} - Lexical Chain 14 with a strength 16, 1 cluster and a score equal to 1.0: {Patrick, loyalties, country, press, conferences, training camp, venues, arenas, gymnasiums, teammates, SUNY Purchase, N.Y, train, Oct, public, start} - Lexical Chain 42 with a strength 7, 1 cluster and a score equal to 1.0: {David Stern, commissioner, compromise, room, lawyers, intricacies, failure} - Lexical Chain 59 with a strength 10, 1 cluster and a score equal to 1.0: {scale, management, middle class, surprise, Sports, entertainment, business, movies, efforts, rise}

Document 2

Parameter 7
- Lexical Chain 4 with a strength 21, 2 clusters and a score equal to 1.0: {sports, Saudi, Arabia's, radio, politics, letter, minister's, office, authorities, Muslim, Ramadan, national centenary celebrations, reasons, suspicion, reason, jewelry, murders, anger, failure of Thai, nationals, mix,} - Lexical Chain 9 with a strength 16, 1 cluster and a score equal to 1.0: {deputy, police, beggars, campaign, streets, detention, hardship, Maj, Gen, Chanvut, Vajrabukka, immigration, control, gangsters, trouble, } - Lexical Chain 11 with a strength 8, 1 cluster and a score equal to 1.0: {Asian Games, Saturday, top, bin, Fahd, Prince Faisal, hosts, observers} - Lexical Chain 20 with a strength 17, 1 cluster and a score equal to 1.0: {award, critics, attempt, bail, property, developer's, Doubts, Thailand's, ability, stage, year and a half, times, decades, downturn, Asia's, version, past} - Lexical Chain 67 with a strength 11, 1 cluster and a score equal to 1.0: {snooker, table, men, breach, grownup, platform, neckties, Pakistani team, manager, Khan, drama}

Parameter 8
- Lexical Chain 77 with a strength 10, 1 cluster and a score equal to 1.06: {gold, medal, glimpse, screen, Security, measures, dogs, pinch, ground, Australia} - Lexical Chain 4 with a strength 21, 2 clusters and a score equal to 1.0: {sports, Saudi, Arabia's, radio, politics, letter, minister's, office, authorities, Muslim, Ramadan, national centenary celebrations, reasons, suspicion, reason, jewelry, murders, anger, failure of Thai, nationals, mix} - Lexical Chain 9 with a strength 16, 1 cluster and a score equal to 1.0: {deputy, police, beggars, campaign, streets, detention, hardship, Maj, Gen, Chanvut, Vajrabukka, immigration, control, gangsters, trouble} - Lexical Chain 11 with a strength 8, 1 cluster and a score equal to 1.0: {Asian Games, Saturday, top, bin, Fahd, Prince Faisal, hosts, observers}

- Lexical Chain 20 with a strength 17, 1 cluster and a score equal to 1.0: {award, critics, attempt, bail, property, developer's, Doubts, Thailand's, ability, stage, year and a half, times, decades, downturn, Asia's, version, past}

Document 3

Parameter 7

- Lexical Chain 0 with a strength 3, 1 cluster and a score equal to 1.0: {United States, couple, competition}
- Lexical Chain 6 with a strength 49, 3 clusters and a score equal to 1.0: {boats, Sunday night, sailor, Sword, Orion, veteran, cutter, Winston Churchill, Solo Globe, Challenger, navy, Race, supposition, instructions, responsibility, skipper, east, Melbourne, deck, kilometer, masts, bodies, races, GMT, Admiral's, Cups, Britain, Star, Class, Atlanta, Seattle, arms, fatality, sea, waves, dark, yacht's, Dad, Guy's, son, Mark, beer, talk, life, Richard, Winning, affair, canopy, death}
- Lexical Chain 9 with a strength 5, 1 cluster and a score equal to 1.0: {record, days, hours, minutes, rescue}
- Lexical Chain 16 with a strength 30, 3 clusters and a score equal to 1.0: {Snow, shape, north, easters, thunder, storm, change, knots, west, level, maxi's, search, Authority, seas, helicopter, night vision, equipment, feet, rescues, Campbell, suffering, hypothermia, safety, foot, sailors, colleagues, Hospital, deaths, bodies, fatality}
- Lexical Chain 19 with a strength 15, 2 clusters and a score equal to 1.0: {challenge, crew, Monday, VC, Offshore, Stand, Newcastle, mid morning, Eden, Rescuers, aircraft, unsure, whereabouts, killing, contact}

Parameter 8
- Lexical Chain 0 with a strength 3, 1 cluster and a score equal to 1.0: {United States, couple, competition} - Lexical Chain 6 with a strength 49, 3 clusters and a score equal to 1.0: {boats, Sunday night, sailor, Sword, Orion, veteran, cutter, Winston Churchill, Solo Globe, Challenger, navy, Race, supposition, instructions, responsibility, skipper, east, Melbourne, deck, kilometer, masts, bodies, races, GMT, Admiral's, Cups, Britain, Star, Class, Atlanta, Seattle, arms, fatality, sea, waves, dark, yacht's, Dad, Guy's, son, Mark, beer, talk, life, Richard, Winning, affair, canopy, death} - Lexical Chain 9 with a strength 5, 1 cluster and a score equal to 1.0: {record, days, hours, minutes, rescue} - Lexical Chain 16 with a strength 30, 3 clusters and a score equal to 1.0: {Snow, shape, north, easters, thunder, storm, change, knots, west, level, maxi's, search, Authority, seas, helicopter, night vision, equipment, feet, rescues, Campbell, suffering, hypothermia, safety, foot, sailors, colleagues, Hospital, deaths, bodies, fatality} - Lexical Chain 19 with a strength 15, 2 clusters and a score equal to 1.0: {challenge, crew, Monday, VC, Offshore, Stand, Newcastle, mid morning, Eden, Rescuers, aircraft, unsure, whereabouts, killing, contact}

Document 4

Parameter 7
- Lexical Chain 18 with a strength 17, 1 cluster and a score equal to 1.0: {Lausanne, Fiat, Gianni, Agnelli, vans, Italian, championships, claims, Howard, Peterson, delegate, ski, cars, federation, executives, fore, disclosure} - Lexical Chain 49 with a strength 5, 1 cluster and a score equal to 1.0: {year, million, agent, teams, statement} - Lexical Chain 64 with a strength 13, 1 cluster and a score equal to 0.9: {candidate, news conference, climate, boycott, Los Angeles, sprinter, Ben, Johnson's, expulsion, moments, member, Marc Hodler, graf} - Lexical Chain 70 with a strength 12, 2 clusters and a score equal to 0.85: {world, championships, claims, Howard, Peterson, delegate, ski, federation, cars, executives, fore, disclosure} - Lexical Chain 23 with a strength 16, 2 clusters and a score equal to 0.7: {Salt, Lake, possibility, championships, claims, Howard, Peterson, delegate, ski, federation, cars, fore, concern, executives, disclosure, kind}

Parameter 8
- Lexical Chain 18 with a strength 17, 1 cluster and a score equal to 1.0: {Lausanne, Fiat, Gianni, Agnelli, vans, Italian, championships, claims, Howard, Peterson, delegate, ski, cars, federation, executives, fore, disclosure} - Lexical Chain 51 with a strength 5, 1 cluster and a score equal to 1.0: {year, million, agent, teams, statement} - Lexical Chain 74 with a strength 8, 1 cluster and a score equal to 1.0: {Intermountain, care, Health, Care, Utah's, provider, apology, statements} - Lexical Chain 66 with a strength 13, 1 cluster and a score equal to 0.9: {candidate, news conference, climate, boycott, Los Angeles, sprinter, Ben, Johnson's, expulsion, moments, member, Marc Hodler, graf} - Lexical Chain 72 with a strength 12, 2 clusters and a score equal to 0.85: {world, championships, claims, Howard, Peterson, delegate, ski, federation, cars, executives, fore, disclosure}

5.2.4 War

Document 1

Parameter 7
- Lexical Chain 25 with a strength 14, 2 clusters and a score equal to 1.0: {lightning, advance, Africa's, nation, outskirts, capital Kinshasa, troops, Angola, Zimbabwe, Namibia, chunk, routes, Katanga, Eastern, Kasai, provinces, copper} - Lexical Chain 53 with a strength 15, 1 cluster and a score equal to 1.0: {Back, years, Ngeyo, farm, farmers, organization, breadbasket, quarter, century, businessman, hotels, tourist, memory, rivalry, rebellions} - Lexical Chain 56 with a strength 20, 1 cluster and a score equal to 1.0: {political, freedoms, Hutus, Mai-Mai, warriors, Hunde, Nande, militiamen, Rwanda, ideology, weapons, persecution, landowners, ranchers, anarchy, Safari, Ngezayo, farmer, hotel, owner, camps} - Lexical Chain 24 with a strength 10, 2 clusters and a score equal to 0.87: {fighting, people, leaders, diplomats, cause, president, Washington, U.S, units, weeks} - Lexical Chain 51 with a strength 10, 2 clusters and a score equal to 0.82: {West, buildings, sight, point, tourists, mountain, gorillas, shops, guest, disputes}

Parameter 8
- Lexical Chain 25 with a strength 14, 2 clusters and a score equal to 1.0: {lightning, advance, Africa's, nation, outskirts, capital Kinshasa, troops, Angola, Zimbabwe, Namibia, chunk, routes, Katanga, Eastern, Kasai, provinces, copper} - Lexical Chain 58 with a strength 15, 1 cluster and a score equal to 1.0: {Back, years, Ngeyo, farm, farmers, organization, breadbasket, quarter, century, businessman, hotels, tourist, memory, rivalry, rebellions} - Lexical Chain 61 with a strength 20, 1 cluster and a score equal to 1.0: {political, freedoms, Hutus, Mai-Mai, warriors, Hunde, Nande, militiamen, Rwanda, ideology, weapons, persecution, landowners, ranchers, anarchy, Safari, Ngezayo, farmer, hotel, owner, camps} - Lexical Chain 56 with a strength 10, 2 clusters and a score equal to 0.82: {West, buildings, sight, point, tourists, mountain, gorillas, shops, guest, disputes} - Lexical Chain 31 with a strength 6, 1 cluster and a score equal to 0.76: {top, Sunday, force, airstrikes, air, defense}

Document 2

Parameter 7
- Lexical Chain 26 with a strength 5, 2 clusters and a score equal to 1.32: {population, autonomy, artillery, enemy, rounds} - Lexical Chain 2 with a strength 17, 2 clusters and a score equal to 1.17: {offensive, observers, key, European Union, time, weekend, segments, brigades, observer, experts, separatist, face, monitors, swing, fields, shells, afternoon} - Lexical Chain 1 with a strength 13, 1 cluster and a score equal to 1.0: {threat, June, advantage, indecision, tank, guerrilla, bedrock, swaths, ban, month, operations, Junik, province's} - Lexical Chain 17 with a strength 7, 1 cluster and a score equal to 1.0: {rebels, control, town, government, miles, kilometers, troops} - Lexical Chain 35 with a strength 6, 1 cluster and a score equal to 1.0: {territory, village, civilians, Friday, abuses, others}

Parameter 8
- Lexical Chain 13 with a strength 12, 1 cluster and a score equal to 1.02: {European Union, segments, brigades, observer, experts, separatist, face, monitors, swing, fields, shells, afternoon} - Lexical Chain 1 with a strength 13, 1 cluster and a score equal to 1.0: {threat, June, tank, advantage, indecision, guerrilla, bedrock, swaths, ban, month, operations, Junik, province's} - Lexical Chain 19 with a strength 7, 1 cluster and a score equal to 1.0: {rebels, control, town, government, miles, kilometers, troops} - Lexical Chain 38 with a strength 6, 1 cluster and a score equal to 1.0: {territory, village, civilians, Friday, abuses, others} - Lexical Chain 54 with a strength 11, 1 cluster and a score equal to 1.0: {meeting, Secretary, General, Javier Solana, Commander, Wesley, Clark, level of fighting, capacity, resumption, route to Belgrade}

Document 3

Parameter 7
- Lexical Chain 3 with a strength 7, 1 cluster and a score equal to 1.0: {troops, rebels, kilometers, miles, town, government, control} - Lexical Chain 16 with a strength 6, 1 cluster and a score equal to 1.0: {capital, statement, resistance, sources, future, rifles} - Lexical Chain 22 with a strength 4, 2 clusters and a score equal to 0.92: {time, afternoon, separatist, weekend} - Lexical Chain 73 with a strength 16, 2 clusters and a score equal to 0.89: {interview, Press, Fears, Pakistani, Tempers, seaside, demonstrators, Activists, pledges, troop, rotations, staff, B.J, Habibie, batch, Year} - Lexical Chain 47 with a strength 15, 2 clusters and a score equal to 0.77: {mosques, strike, Muslims, church, offices, Nusa, Tenggara, Organizers, Monday's, residents, fires, rival, gangs, rocks, world's}

Parameter 8
- Lexical Chain 3 with a strength 7, 1 cluster and a score equal to 1.0: {troops, rebels, kilometers, miles, town, government, control} - Lexical Chain 18 with a strength 6, 1 cluster and a score equal to 1.0: {capital, statement, resistance, sources, future, rifles} - Lexical Chain 84 with a strength 16, 2 clusters and a score equal to 0.89: {interview, Press, Fears, Pakistani, Tempers, seaside, demonstrators, Activists, pledges, troop, rotations, staff, B.J, Habibie, batch, Year} - Lexical Chain 53 with a strength 15, 2 clusters and a score equal to 0.77: {mosques, strike, Muslims, church, offices, Nusa, Tenggara, Organizers, Monday's, residents, fires, rival, gangs, rocks, world's} - Lexical Chain 16 with a strength 6, 2 clusters and a score equal to 0.66: {people, leaders, unit, World, diplomats, situation}

Document 4

Parameter 7
- Lexical Chain 11 with a strength 6, 1 cluster and a score equal to 1.0: {parliament, threats, Organization, proposal, danger, option} - Lexical Chain 58 with a strength 8, 1 cluster and a score equal to 1.0: {peace process, visit's, differences, Yosef, Lapid, editions, Maariv, concern} - Lexical Chain 21 with a strength 9, 2 clusters and a score equal to 0.94: {process, land, clashes, roads, smoke, areas, elements, role, disagreement} - Lexical Chain 97 with a strength 15, 1 cluster and a score equal to 0.80: {negotiator, Saeb Erekat, childhood, friend, Samir, Abdel, Salam, studies, lesson, logic, cousin's, death, pray, rhythm, Memorandum} - Lexical Chain 46 with a strength 45, 2 clusters and a score equal to 0.75: {stones, bullets, gas, lands, house, America, occupation, debris, canisters, remnants, tires, flag, wind, tinny, wail, verses, relatives, man, university, student, roof, soldier's, Abu Dis, fears, Bank's, spasm, windows, TV, court, exhortations, intefadeh, worth, greeting, strains, anthem, friends, politics, Nasr, Amr, disbelief, brother, Taher, ceremony, survival, Palestinian run}

Parameter 8
- Lexical Chain 11 with a strength 6, 1 cluster and a score equal to 1.0: {parliament, threats, Organization, proposal, danger, option} - Lexical Chain 52 with a strength 10, 1 cluster and a score equal to 1.0: {footage, being, car, chunks, Palestinian, Yasser, Arafat's, court, exhortations, intefadeh} - Lexical Chain 65 with a strength 8, 1 cluster and a score equal to 1.0: {peace process, visit's, differences, Yosef, Lapid, editions, Maariv, concern} - Lexical Chain 113 with a strength 15, 1 cluster and a score equal to 0.80: {negotiator, Saeb Erekat, childhood, friend, Samir, Abdel, Salam, studies, lesson, logic, cousin's, death, pray, rhythm, Memorandum} - Lexical Chain 17 with a strength 7, 1 cluster and a score equal to 0.76: {officials, prime minister, Sunday, force, top, President, measures}

Document 5

Parameter 7
- Lexical Chain 4 with a strength 5, 1 cluster and a score equal to 1.0: {Friday, killing, others, village, territory} - Lexical Chain 17 with a strength 4, 1 cluster and a score equal to 1.0: {troops, kilometers, miles, government} - Lexical Chain 40 with a strength 16, 1 cluster and a score equal to 1.0: {England, career, Hickey's, sister, Deborah, tears, electrics, sort, stuff, family, parents, Thames, Ditton, back Christmas, Britain's} - Lexical Chain 56 with a strength 21, 2 clusters and a score equal to 0.62: {pickup, videotape, fellow, spy, Assailants, prosecutor, evening, work, band, prison, Tagirov's, Deputy, Turpal, suspect, Apti, Abitayev, security forces, person, turf, wars, Hundreds} - Lexical Chain 6 with a strength 12, 2 clusters and a score equal to 0.53: {news, agency, end, security, ITAR Tass, Ministry, forces, Igor, spokesman, report, Nations, U.N}

Parameter 8
- Lexical Chain 4 with a strength 5, 1 cluster and a score equal to 1.0: {Friday, killing, others, village, territory} - Lexical Chain 17 with a strength 4, 1 cluster and a score equal to 1.0: {troops, kilometers, miles, government} - Lexical Chain 45 with a strength 16, 1 cluster and a score equal to 1.0: {England, career, Hickey's, sister, Deborah, tears, electrics, sort, stuff, family, parents, Thames, Ditton, back Christmas, Britain's} - Lexical Chain 61 with a strength 21, 2 clusters and a score equal to 0.62: {pickup, videotape, fellow, spy, Assailants, prosecutor, evening, work, band, prison, Deputy, Turpal, Atgeriyev, suspect, Apti, Abitayev, security forces, person, turf, wars, Hundreds} - Lexical Chain 6 with a strength 12, 2 clusters and a score equal to 0.53: {news, agency, end, security, ITAR Tass, Ministry, forces, Igor, spokesman, report, Nations, U.N}

Chapter 6

Conclusion

In this thesis, we implemented a novel method for building Lexical Chains. Unlike the approaches which use WordNet as a linguistic resource, we examine lexical cohesive relationships that cannot be defined in terms of thesaural relationships, but are considered "intuitively" related due to their regular co-occurrence in text. We construct Lexical Chains using a lexico-semantic knowledge base automatically constructed as resource.

Our lexico-semantic knowledge base is a resource in which words with similar meaning are hierarchically grouped together into clusters. For this purpose, we applied the PoBOC (Cleuziou *et al*., 2004) algorithm to word-context vectors obtained by the application of the SCP measure (Silva *et al*., 1999) and the InfoSimBA (Dias and Alves, 2005) similarity measure. In order to determine if two concepts are semantically related we use our knowledge base and a measure of semantic similarity based on the ratio between the information content of the most informative common ancestor and the information content of both concepts. In particular, we use the similarity measure of Lin (1998). We use this semantic similarity measure to define a relatedness criterium in order to assign a given word to a given chain in the lexical chaining process. This assignment of a word to a chain is a compromise between the approaches of Hirst and St-Onge (1997) and Barzilay and Elhadad (1997). Our high-scoring chain process is similar to Barzilay and Elhadad's strong chain paradigm.

Our experimental evaluation shows that relevant Lexical Chains can be constructed with our lexical chaining algorithm, which behaves independently from domains to do-

mains and texts to texts.

6.1 Future work

We intend to pursue our research in the following directions:

- Compare our experimental results with results obtained with other algorithms and compare our algorithm but using WordNet as linguistic resource instead of our lexico-semantic knowledge base.
- Use Text Segmentation (Dias and Alves, 2005) in the lexical chaining algorithm. Text Segmentation can be defined as the automatic identification of boundaries between distinct textual units (segments) in a document. With respect to segmentation, an analysis of lexical cohesion can be used to indicate portions of text that represent single topical units or segments i.e. they contain a high number of semantically related words, which should benefit the lexical chainer.
- Consider different syntactic categories beyond nouns as candidate words to Lexical Chains like verbs and adjectives. The choice of this is based on the assumption that verbs indicate the real "action" of the context document.
- Compare results with other association measures beyond the Symmetric Conditional Probability (SCP) to build the word-context vectors.
- Enrich the lexico-semantic knowledge base with nominal units in the nodes of our hierarchical tree.
- Reduce the algorithm runtime with a parallel programming.
- Extract sentences to obtain a cohesive summary for a document using the computed Lexical Chains.
- Construct the Lexical Chains with a knowledge base which group all the knowledge bases already constructed for the four different domains.

Bibliography

[1] Barzilay, R. and Elhadad, M. (1997). Using Lexical Chains for Text Summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS-97)*, ACL. Madrid, Spain, pp. 10-18.

[2] Brants, T. (2000). TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*. Seattle, WA.

[3] Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, pp. 152-155, Trento, IT.

[4] Brown, P., Pietra, V., deSouza, P., Lai, J. and Mercer, R. (1992). Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4), pp. 467-479.

[5] Choueka, Y. (1988). Looking for needles in a haystack, or locating interesting collocational expressions in large textual databases. In *Proceedings of RIAO'88 Conference*, Cambridge, MA, pp. 609-623

[6] Cleuziou, G., Martin, L. and Vrain, C. (2004). PoBOC: an Overlapping Clustering Algorithm, Application to Rule-Based Classification and Textual Data. In *Proceedings of the 16th biennial European Conference on Artificial Intelligence (ECAI'04)*, Valencia, Spain, pp. 440-444.

[7] Daille, B. (1995). Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In *The balancing act combining symbolic and statistical approaches to language*. MIT Press.

[8] Dias, G. (2002). Extraction Automatique d'Associations Lexicales à Partir de Corpora. PhD Thesis, New University of Lisbon (Portugal) and University of Orléans (France).

[9] Dias, G. and Alves, E. (2005). Unsupervised Topic Segmentation Based on Word Co-occurrence and Multi-Word Units for Text Summarization. In *Proceedings of the ELECTRA Workshop associated to 28th Annual International ACM SIGIR Conference*, Salvador, Brazil, pp. 41-48.

[10] Dias, G., Guilloré, S. and Lopes, G. (1999). Language Independent Automatic Acquisition of Rigid Multiword Units from Unrestricted Text corpora. In *Proceedings of Traitement Automatique des Langues Naturelles*. Institut d'Etudes Scientifiques, Cargèse, France.

[11] Fellbaum, C.D. (1998). WordNet: An Electronic Lexical Database. MIT Press, New York.

[12] Galley, M. and McKeown, K. (2003). Improving Word Sense Disambiguation in Lexical Chaining. In *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, Acapulco, Mexico.

[13] Gil, A. and Dias, G. (2003). Efficient Mining of Textual Associations. In *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering Chengqing Zong (eds) Beijing*. China, IEEE Press. pp. 549-555.

[14] Habert, B. and Jacquemin, C. (1993). Noms composés, termes, dénominations complexes: Problématiques linguistiques et traitements automatiques. In Traitement Automatique des Langues. 34(2) pp. 5-41.

[15] Halliday, M.A.K. and Hasan, R. (1976). Cohesion in English. London: Longman.

[16] Harris, Z., Gottfried, M., Ryckman, T., Mattick, P., Daladier, A., Harris, T. N. and Harris, S. (1989). The form of Information in Science : Analysis of an immunology sublanguage. Dordrecht: Kluwer Adademic Publishers.

[17] Hearst, M. (1997). TextTiling: Segmenting Text into Multi-Paragraph Subtopic passages. *Computational Linguistics*, 23(1), pp. 33-64.

[18] Hirst, G. and St-Onge, D. (1997). Lexical Chains as representation of context for the detection and correction of malapropisms. In *WordNet: An electronic lexical database and some of its applications*. MIT Press.

[19] Hirst, G., DiMarco, C. and Hovy, E. (1997). Authoring and generating health-education documents that are tailored to the needs of the individual patient. In *Proceedings of the Sixth International Conference on User Modeling*. Sardinia, Italy. pp. 107-18.

[20] Hobbs, J. (1985). On the coherence and structure of discourse. Stanford University, California.

[21] Lin, D. (1998). An information-theoretic definition of similarity. In *15th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA.

[22] Mani, I. (2001). Automatic Summarization. John Benjamins Publishing Co, Amsterdam/ Philadelphia.

[23] Marcu, D. (1997). The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts. PhD Thesis, Department of Computer Science, University of Toronto. Canada.

[24] Maybury, M. and Merlino, A. (1997). Multimedia summaries of broadcast news. In *Multimedia Information Retrieval*. Maybury, M. editor.

[25] McKeown, K., Barzilay R., Chen J., Elson D., Evans D., Klavans J., Nenkova A., Schiffman B., and Sigelman S. (2003). Columbia's newsblaster: New features and future directions (demo). In *Proceedings of NAACL-HLT'03*.

[26] Miller, G. and Charles, W. (1991). Contextual correlates of semantic similarity. In *Language and Cognitive Processes*, 6(1), pp. 1-28.

[27] Miller, G. (1995). WordNet: An Lexical Database for English. *Communications of the Association for Computing Machinery (CACM)*, 38(11), pp. 39-41.

[28] Morris, J. and Hirst, G. (1991). Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistics*, 17(1), pp. 21-48.

[29] Rabiner, L.R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, 77(2), pp. 257-285.

[30] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI-95*. Montreal, Canada. pp. 448-453.

[31] Rohall, S., Gruen, D., Moody, P., Wattenberg, M., Stern, M., Kerr, B., Stachel, B., Dave, K., Armes, R., and Wilcox, E. (2004). Remail: a reinvented email prototype. In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pp. 791-792. ACM Press.

[32] Salton, G., Yang, C.S., and Yu, C.T. (1975). A theory of term importance in automatic text analysis. Amer. Soc. Inf. Sc 26, 1, pp. 33-44.

[33] Sampson, G. (1995). English for the computer. Oxford University Press.

[34] Samuelsson, C. (1993). Morphological tagging based entirely on Bayesian inference. In *9th nordic conference on computational linguistics NODALIDA-93*. Stockholm University, Stockholm, Sweden.

[35] Silber, G. and McCoy, K. (2002). Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization. *Computational Linguistics*, 28(4), pp. 487-496.

[36] Silva, J., Dias, G., Guilloré, S. and Lopes, G. (1999). Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units. In *Proceedings of 9th Portuguese Conference in Artificial Intelligence*. Springer-Verlag.

[37] Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1), pp. 143-177.

[38] Sparck-Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1), pp. 11-21.

[39] Waibel, A., Bett, M., Finke, R. and Stiefelhagen., M. (1998). Meeting browser: tracking and summarization meetings. In *Proceedings of the 1998 DARPA Broadcast News Workshop*.