AUTOMATIC TEXT SUMMARIZATION WITH MAXIMAL FREQUENT SEQUENCES

BIE

Biblioteca de Investigación Especializada

AUTOMATIC TEXT SUMMARIZATION WITH MAXIMAL FREQUENT SEQUENCES

Yulia Nikolaevna Ledeneva
René Arnulfo García Hernández

Colección Biblioteca de Investigación Especializada
Área del conocimiento: Humanidades/Ingeniería

Primera edición: diciembre de 2013

ISBN EÓN: 978-607-8289-50-9
ISBN UAEM: 978-607-422-505-1

D.R. © Universidad Autónoma del Estado de México
Instituto Literario núm. 100 Ote.
Colonia Centro
C.P. 50000. Toluca, México
http://www.uaemex.mx

D. R. © Ediciones y Gráficos Eón, S.A. de C.V.
Av. México-Coyoacán núm. 421
Colonia Xoco, Delegación Benito Juárez
C.P. 03330, México, Distrito Federal
Tels.: 56 04 12 04 y 56 88 91 12
administracion@edicioneseon.com.mx
www.edicioneseon.com.mx

El contenido total de este libro fue sometido a dictamen en el sistema de pares ciegos.

Impreso y hecho en México
Printed and made in Mexico

Índice

Abstract

In the last two decades, an exponential increase in the available electronic information causes a big necessity to quickly understand large volumes of information. It raises the importance of the development of automatic methods for detecting the most relevant content of a document in order to produce a shorter text. Automatic Text Summarization (ATS) is an active research area dedicated to generate abstractive and extractive summaries not only for a single document, but also for a collection of documents. Other necessity consists in finding method for ATS in a language and domain independent way.

In this book we consider extractive text summarization for single document task. We have identified that a typical extractive summarization method consists in four steps. First step is a term selection where one should decide what units will count as individual terms. The process of estimating the usefulness of the individual terms is called term weighting step. The next step denotes as sentence weighting where all the sentences receive some numerical measure according to the usefulness of its terms. Finally, the process of selecting the most relevant sentences calls sentence selection. Different extractive summarization methods can be characterized how they perform these steps.

In this book, in the term selection step, we describe how to detect multiword descriptions considering Maximal Frequent Sequences (MFSs), which bearing important meaning, while non-maximal frequent sequences (FSs), those that are parts of another FS, should not be considered. Our additional motivation was cost vs. benefit considerations: there are too many non-maximal FSs while their probability to bear important meaning is lower. In any case, MFSs represent all FSs in a compact way: all FSs can be obtained from all MFSs by bursting each MFS into a set of all its subsequences.

New methods based on graph algorithms, genetic algorithms, and clustering algorithms which facilitate the text summarization task are presented. We have tested different combinations of term selection, term weighting, sentence weighting and sentence selection options for language-and domain-independent extractive single-document text summarization on a news report collection. We analyzed several options based on MFSs, considering them with graph, genetic, and clustering algorithms. We obtained results superior to the existing state-of-the-art methods.

This book is addressed for students and scientists of the area of Computational Linguistics, and also who wants to know recent developments in the area of Automatic Text Generation of Summaries.

Resumen

En las últimas dos décadas un aumento exponencial de la información electrónica ha provocado una gran necesidad de entender rápidamente grandes volúmenes de información. En este libro se desarrollan los métodos automáticos para producir un resumen. Un resumen es un texto corto que transmite la información más importante de un documento o de una colección de documentos. Los resúmenes utilizados en este libro son extractivos: una selección de las oraciones más importantes del texto. Otros retos consisten en generar resúmenes de manera independiente de lenguaje y dominio.

Se describe la identificación de cuatro etapas para generación de resúmenes extractivos. La primera etapa es la selección de términos, en la que uno tiene que decidir qué unidades contarían como términos individuales. El proceso de estimación de la utilidad de los términos individuales se llama etapa de pesado de términos. El siguiente paso se denota como pesado de oraciones, donde todas las secuencias reciben alguna medida numérica de acuerdo con la utilidad de términos. Finalmente, el proceso de selección de las oraciones más importantes se llama selección de oraciones. Los diferentes métodos para generación de resúmenes extractivos pueden ser caracterizados como representan estas etapas.

En este libro se describe la etapa de selección de términos, en la que la detección de descripciones multipalabra se realiza considerando Secuencias Frecuentes Maximales (SFMs), las cuales adquieren un significado importante, mientras Secuencias Frecuentes (SF) no maximales, que son partes de otros SF, no deben de ser consideradas. En la motivación se consideró costo vs. beneficio: existen muchas SF no maximales, mientras que la probabilidad de adquirir un significado importante es baja. De todos modos, las SFMs representan todas las SFs en el modo compacto: todas las SFs podrían ser obtenidas a partir de todas las SFMs explotando cada SFM al conjunto de todas sus subsecuencias.

Se presentan los nuevos métodos basados en grafos, algoritmos de agrupamiento y algoritmos genéticos, los cuales facilitan la tarea de generación de resúmenes de textos. Se ha experimentado diferentes combinaciones de las opciones de selección de términos, pesado de términos, pesado de oraciones y selección de oraciones para generar los resúmenes extractivos de textos independientes de lenguaje y dominio para una colección de noticias. Se ha analizado algunas opciones basadas en descripciones multipalabra considerándolas en los métodos de grafos, algoritmos de agrupamiento y algoritmos genéticos. Se han obtenido los resultados superiores al de estado de arte.

Este libro está dirigido a los estudiantes y científicos del área de Lingüística Computacional, y también a quienes quieren saber sobre los recientes avances en las investigaciones de generación automática de resúmenes de textos.

Introduction

This section introduces some basic concepts in the first chapter. Then what we can learn in this book is described. The research developments presented in this book are summarized, and the methodology is presented. Finally, the organization of this book is included.

Introduction

A summary of a document is a (much) shorter text that conveys the most important information from the source document. There are a number of scenarios where automatic construction of such summaries is useful. For example, an information retrieval system could present an automatically built summary in its list of retrieval results, for the user to quickly decide which documents are interesting and worth opening for a closer look—this is what Google models to some degree with the snippets shown in its search results. Other examples include automatic construction of summaries of news articles or email messages to be sent to mobile devices as SMS; summarization of information for government officials, businessmen, researches, etc., and summarization of web pages to be shown on the screen of a mobile device, among many others.

Text summarization tasks can be classified into single-document and multi-document summarization. In single-document summarization, the summary of only one document is to be built, while in multi-document summarization the summary of a whole collection of documents (such as all today's news or all search results for a query) is built. In this book we present and experiment with single-document.

The summarization methods can be classified into abstractive and extractive summarization [Lin97]. An abstractive summary is an arbitrary text that describes the contexts of the source document. Abstractive summarization process consists of "understanding" the original text and "re-telling" it in fewer words. Namely, an abstractive summarization method uses linguistic methods to examine and interpret the text and then to find new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original document. While this may seem the best way to construct a summary (and this is how human beings do it), in real-life setting immaturity of the corresponding linguistic technology for text analysis and generation currently renders such methods practically infeasible.

An extractive summary, in contrast, is a selection of sentences (or phrases, paragraphs, etc.) from the original text, usually presented to the user in the same order—i.e., a copy of the source text with most sentences omitted. An extractive summarization method only decides, for each sentence, whether or not it will be included in the summary. The resulting summary reads rather awkward; however, simplicity of the underlying statistical techniques makes extractive summarization an attractive, robust, language-independent alternative to more "intelligent" abstractive methods. In this book, we consider extractive summarization.

A typical extractive summarization method consists in several steps, at each of them different options can be chosen. We will assume that the units of selection are sentences (these could be, say, phrases or paragraphs). Thus final goal of the extractive summarization process is *sentence selection*. One of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones; the process of assigning these usefulness weights is called *sentence weighting*. One of the ways to estimate the usefulness of a sentence is to sum up usefulness weights of individual terms of which the sentence consists; the process of estimating the individual terms is called *term weighting*. For this, one should decide what the terms are: for example, they can be words; deciding what objects will count as terms is the task of *term selection*. Different extractive summarization methods can be characterized by how they perform these tasks.

In this book we present new term selection, term weighting, sentence weighting, and sentence selection steps. We analyze several options for simple language-independent statistical term selection and corresponding term weighting, based on units larger than one word. In particular, we are looking for new terms denoted as multiword descriptions which could be good terms for the task of text summarization.

The book introduces basic concepts and definitions, summaries state-of-the-art of text summarization methods, describes proposed methods, and presents experimental settings and obtained results for different term selection, term weighting, sentence selection and sentence selection schemes. The conclusions are given.

1. What we can learn in this book

- Identification of general steps for an automatic extractive text summarization method.
- Description of new methods for generating text summarizes based on the extraction of maximal frequent sequences.
- Development of new methods for single-document summarization.
- New methods to deal with the task of generation of summaries in a language-independent way.
- New methods to deal with the task of generation of summaries in a domain-independent way.
- New methods for automatic generation of text summaries which are superior to the state-of-the-art methods.

2. Research objectives

1. New state-of-the art methods in various tasks of automatic generation of summaries of a single-document, with application to a different languages.
2. Development of the system for automatic generating of summaries with the superior quality to the state-of-the-art methods. The system will be useful for the users and also will serve as a framework and evaluation for the developed methods.
3. New methods to deal with the task of generation of summaries in a language- and domain-independent way.
4. New methods for generating text summarizes based on the discovery of multiword descriptions.

3. Methodology

For the development of new methods for automatic generation of text summaries, it is necessary to realize the following methodology:

1. Pre-processing stage:
    1.1 Preparation of corpus for the stage of experiments.

   1.2 Usage of lexical resources of the-state-of-the-art (WordNet, EuroWordNet, WordNet Similarity, GATE (General Architecture for Text Engineering), Natural Language Toolkit.
   1.3 Usage of measures for the evaluation of proposed methods: precision, recall, f-measure.
   1.4 Implementation of the methods and tools for the evaluation of proposed methods (ROUGE, SEE).
   1.5 Development in programming languages (Java, Perl, Builder C++).

2. Automatic generation of summaries of the-state-of the-art methods:
   2.1 Extraction of MFSs for single document.
   2.2 Generation of summaries for different languages.
   2.3 Tests, adjustments, documentation.

3. Automatic generation of summaries:
   3.1 Development of methods for automatic generation of text summaries which includes extraction of MFSs of single document.
   3.2 Implementation of the methods for automatic generation of summaries for English language.
   3.3 Tests, adjustments, documentation.

4. Documentation and presentations.

4. Organization of the book

This book is divided into four chapters, an introduction and conclusions. The first chapter summarizes state-of-the-art methods. In the second chapter theoretical framework is described. In Chapter III new methods for single-text summarization are presented. In Chapter IV, the experimental results are reported. And finally we present the conclusions and a future work.

Chapter I

State-of-the-art methods

Mainly, this chapter is dedicated of the detailed presentation of the-state-of-the-art. We begin this chapter with a presentation of natural language laboratory. Section I.1.1 describes the area of Computational Linguistics and its applications. In section I.2, an introduction to text summarization and evaluation measures used for summarization are given. Then, we present a detailed description of the state-of-the-art of extractive summarization. This section is ordered taking into account four steps of extractive summarization. A brief state-of-the-art for abstractive summarization and applications of text summarization are given. Finally, we conclude the chapter with the description of the research developments presented in this book.

I.1 Computational Linguistics

The Computational Linguistics (CL) area describes the modern models of how natural language processing systems function, explains how to compile the data for the necessity of the Natural Language Processing (NLP) systems, develops software for *n*-grams correction, resolves word sense disambiguation, constructs dictionaries and databases, retrieves information, translates automatically from one language to another, etc. [Bol04a]. As many other areas (for example, such as mechanical and engineering areas), CL has the necessity of intelligent language processing tools and automation of NLP tasks.

We mention some of the areas of NLP:

– Word Sense Disambiguation (WSD) [Gel03a, Man99]: solves what sense has a given word, generally based on its context. This task is very important because

of its successful resolution, depends the correctness of other applications such as Machine Translation, Question Answering, etcetera.
- Information Retrieval (IR) [Man07, Bae99]: consists of finding documents of an unstructured nature that satisfies an information need from within large collections of documents usually on local computer or in the internet. This area overtakes traditional database searching, becoming the dominant form of information access. Now hundreds of millions of people use IR systems every day when they use a web search engine or search their emails.
- Machine Translation [Gel03b, Bol04a]: is a machine-assisted system responsible for translation from one language to another. This application is very useful for business and scientific purposes by reason that the international collaboration grows exponentially.
- Question Answering (QA) [Ace07, Fer07]: is a complex task that combines techniques from NLP, IR and Machine Learning. The main aim of QA is to localize the correct answer to a question written in natural language in a non-structured collection of documents. Systems of QA look like a search engine where the input to the system is a question in natural language and the output is the answer to the question (not a list of entire documents like in IR).

I.1.1 Computational Linguistics in Mexico

More than 50 years have passed since first advances were published in the area of CL. From this time a lot of work has been done all over the world including Mexico. Especially, we would like to mention a work of Natural Language Laboratory where more than 300 papers during last 10 years were created mainly by three researches: Ph.D. Alexander Gelbukh, Ph.D. Igor Bolshakov, and Ph.D. Grigori Sidorov [Gel08]. Their works establish basic definitions and new research discoveries in different tasks of CL:

- Lexical resources [Gel06]
- Construction and compilation of dictionaries [Gel02; Gel03c; Gel04a]
- Database of collocations called CrossLexica [Bol01; Bol04b; Bol08]
- Syntactic analysis of the Spanish language [Gal07]
- Semantic errors and malapropism [Gel04b; Bol05]
- Word sense disambiguation [Gel03a; Ledo03]
- Automatic translation [Gel03b]
- Text mining [Mon01; Mon02]

Since the foundation of this Laboratory, CL has grown to become a major scientific domain in Mexico. For the past decade, educational and commercial CL systems have been successfully developed [Sid05] (see [Gel08] for more details). Interest of scientists of all over the world has also been growing, as we can see in special organized conferences [CICLing], and publishing plenty of works [Gel08]. Moreover, several Ph.D. students graduated from the Laboratory establish new Natural Language Laboratories [Ina08, Una08].

I.2 Text summarization

Early experimentation in the late 1950's and 1960's suggested that text summarization by computer was feasible though not straightforward. After a gap of some decades, progress in language processing, coupled with the growing presence of on-line text—in corpora and especially on the web—renewed interest in automated text summarization. So, the huge amount of available electronic documents in Internet has motivated the development of very good information retrieval systems. However, the information provided by such systems, like Google, only shows part of the text where the words of the request query appear. Therefore, the user has to decide if a document is interesting only with the extracted part of a text. Moreover, this part does not have any information if the retrieved document is interesting for the user, so it is necessary download and read each retrieved document until the user finds satisfactory information. It was unnecessary and time-consuming routine. A solution for such problems is to achieve an automatic text summarization extracting the essential sentences of the document.

The demand of the automatic generation of text summaries has appeared in other areas, for example, summaries of news articles; summaries of electronic mails and news to send them as SMS; summaries of information (for government officials, businessmen, researches, etc.); summaries of web pages to transmit them through telephone; in searching systems to receive the summaries of found documents and pages.

From one side, there is a single-document summarization which implies to communicate the principal information of one specific document, and from another side—a multi-document summarization which transmits the main ideas of a collection of documents. There are two options to achieve a summarization by computer: text abstraction and text extraction [Lin97]. Text abstraction examines a given text using linguistic methods which interpret a text and find new concepts to describe it. And then new text is generated which will be shorter with the same content of information. Text extraction means extract parts (words, sequences, sentences, paragraphs, etc.) of a given text based on statistic, linguistic or heuristic methods, and then join them to new text which will be shorter with the same content of information.

According to the classical point of view (see below how we introduce our point of view), there are three stages in automated text summarization [Hov03]. The first stage is performed by *topic identification* where almost all systems employ several independent modules. Each module assigns a score to each unit of input (word, sentence, or longer passage); then a combination module combines the scores for each unit to assign a single integrates score to it; finally, the system returns the *n* highest-scoring units, according to the summary length requested by the user. The performance of topic identification modules is usually measured using recall and precision scores (see section below).

The second stage denotes as the stage of *interpretation*. This stage distinguishes extract-type summarization systems from abstract-type systems. During the interpretation the topics identified as important are fused, represented in new terms, and expressed using a new formulation, using concepts or words not found in the original text. No system can perform interpretation without prior knowledge about the domain; by definition, it must interpret the input in term of something extraneous to the text. But acquisition deep enough prior domain knowledge is so difficult that summarizers to date have only attempted it in a small way. So, the disadvantage of this stage remains blocked by the problem of domain knowledge acquisition.

Summary *generation* is the third stage of text summarization. When the summary content has been created in internal notation, and thus requires the techniques of natural language generation, namely text planning, sentence planning, and sentence realization.

We identified four steps for composing a text summary:

- *Term selection*: during this step one should decide what units will count as terms, for example, they can be words, *n*-grams or phrases.
- *Term weighting*: this is a process of weighting (or estimating) individual terms.
- *Sentence weighting*: the process of assigning numerical measure of usefulness to the sentence. For example, one of the ways to estimate the usefulness of a sentence is to sum up usefulness weights of individual terms of which the sentence consists.
- *Sentence selection*: selects sentences (or other units selected as final parts of a summary). For example, one of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones.

I.2.1 Evaluation of summaries

Evaluation using ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [Lin03a] was proposed by Lin and Hovy [Lin04a, Lin04b, Lin04c]. This system calculates the quality of a summary generated automatically by comparing to the summary (or several summaries) created by humans. Specifically, it counts the number of overlapping different units such as word sequences, word pairs and n-grams between the computer-generated summary to be evaluated and the ideal summaries created by humans. ROUGE includes several automatic evaluation measures:

- ROUGE-N (n-grams co-occurrence):
  Is an n-gram recall between a candidate summary and a set of reference summaries, and calculates as follows:

$$ROUGE-N = \frac{\sum_{S\in\{\mathrm{Re}\,ferencesSummaries\}} \sum_{gram_n\in S} count_{match}(gram_n)}{\sum_{S\in\{\mathrm{Re}\,ferencesSummaries\}} \sum_{gram_n\in S} count\quad(gram_n)},$$

  where n is a length of the *n-gram*, $gram_n$ and $\text{count}_{\text{match}}$ ($gram_n$) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

- ROUGE-L (longest subsequence):
  A sequence $S = (s_1, s_2, \ldots, s_n)$ is a subsequence of another sequence $X = (x_1, x_2, \ldots, x_m)$, if there exists a strict increasing sequence $(i_1, i_2, \ldots, i_k)$ of indices of X such that all $j = 1, 2, \ldots, k$, than $x_{ij} = s_j$ [Cor89]. Given two sequences X and Y, the longest common subsequence (LCS) of X and Y is a common subsequence with maximum length. When LCS is applied in summarization evaluation, a summary sentence viewed as a sequence of words. Intuitively, the longer the LCS of two summary sentences is, the more similar the two summaries X of length m and Y of length n, assuming X is a reference summary sentence and Y is a candidate summary sentence.

- ROUGE-W (weighted longest subsequence):
  Given two sequences X and Y, LCS is called weighted if a length is calculated using a weighted function. For more details about weighted function see in [Lin03a].

– ROUGE-S (skip-bigram co-occurrence):
  Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. Skip-bigram co-occurrence statistics measure the overlap of skip-bigrams between a candidate summary and a set of reference summaries.

It is showed in [Lin03b] that these types of measures can be applied for evaluating the quality of summaries generated automatically achieving 95% of correlation of human judgments.

For each of the measures (ROUGE-N, ROUGE-L, ROUGE-W, etc.), ROUGE returns Recall, Precision and F-measure scores as follows:

Precision (P): reflects how many of the system's extracted sentences were good

$$P = \frac{\#(correct)}{\#(correct+wrong)}$$

Recall (R): reflects how many good sentences the system missed

$$R = \frac{\#(correct)}{\#(correct+missed)}$$

F-measure (F):

$$F = \frac{2PR}{P+R},$$

where *correct* is the number of sentences extracted by the system and the human, *wrong* is the number of sentences extracted by the system but not by the human, and *missed* is the number of sentences extracted by the human but not by the system.

I.3 Extractive text summarization

I.3.1 Term selection

Most works discussed below are based on *words* as terms; however, is not the only possible option. Liu et al. [Liu06a] use pairs of syntactically connected words (*basic elements*) as atomic features (terms). Such pairs (which can be thought of as arcs in the syntactic dependency tree of the sentence) have been shown to be more precise semantic units than words [Kos04]. However, while we believe that trying text units larger than a word is a good idea, extracting the basic elements from the text requires dependency syntactic parsing, which is language-dependent. Simpler statistical methods (cf. the use of *n-grams* as terms in [Vil06]) may prove to be more robust and language-independent.

Some approaches of text summaries match semantic units such as *elementary discourse units* [Mar01, Sor03], *factoids* [Teu04a, Teu04b], *information nuggets* [Voo04], *basic elements* [Liu06a], etc. A big disadvantage of these semantic units is that the detection of these units is realized manually. For example, *information nuggets* are atomic pieces of interesting information about the target identified by human annotators as vital (required) or non-vital (acceptable but not required) for the understanding of the content of a summary.

Factoids are semantic units which represent the meaning of a sentence. For instance, the sentence "The police have arrested a white Dutch man" by the union of the following factoids: "A suspect was arrested", "The police did the arresting", "The suspect is white", "The suspect is Dutch", "The suspect is male". Factoids are defined empirically based on the data in the set of summaries (usually are some manually made summaries taken from [Duc]). Factoid definition starts with the comparison of the information contained in two summaries, and factoids get added or split as incrementally other summaries are considered. If two pieces of information occur together in all summaries and within the same sentence, they are treated as one factoid, because differentiation into more than one factoid would not help us in distinguishing the summaries. Factoids are labeled with descriptions in natural language; initially, these are close in wording to the factoid's occurrence in the first summaries, though the annotator tries to identify and treat equally paraphrases of the factoid information when they occur in other summaries. If (together with various statements in other summaries) one summary contains "was killed" and another "was shot dead", we identify the factoids: "There was an attack", "The victim died", "A gun was used". The first summary contains only the first two factoids, whereas the second contains all three. That way, the semantic similarity between related sentences can be expressed. When factoids are identified in the collection of summaries, most factoids turned out to be independent of each other. But when dealing with naturally occurring documents many difficult cases appear, e.g. ambiguous expressions, slight differences in numbers and meaning, and inference.

The text is segmented in *Elementary Discourse Units* (EDUs) or non-overlapping segments, generally taken as clauses or clauses like units of a rhetorical relation that holds between two adjacent spans of text [Mar01, Car03]. The boundaries of EDUs are determined using grammatical, lexical, and syntactic information of the whole sentence.

Other possible option proposed by Nenkova in [Nen06] is *Semantic Content Units* (SCUs). The definition of the content unit is somewhat fluid, it can be a single word but it is never bigger than a sentence clause. The most important evidence of their presence in a text is the information expressed in two or more summaries, or in other words, is the frequency of the content unit in a text. Other evidence is that these frequent content units can have different wording (but the same semantic meaning) what brings difficulties for language-independent solution.

The concept of *lexical chains* was first introduced by Morris and Hirst. Basically, lexical chains exploit the cohesion among an arbitrary number of related words [Mor91]. Then, lexical chains were computed in a source document by grouping (chaining) sets of words that are semantically related (i.e., have a sense flow) [Bar99, Sil02]. Identities, synonyms, and hypernym/hyponyms are the relations among words that might cause them to be grouped into the same lexical chain. Specifically, words may be grouped when:

– Two noun instances are identical, and are used in the same sense. (The <u>house</u> on the hill is large. The <u>house</u> is made of wood.)
– Two noun instances are used in the same sense (i.e., are synonyms). (The <u>car</u> is fast. My <u>automobile</u> is faster.)
– The senses of two noun instances have a hypernym/hyponym relation between them. (John owns a <u>car</u>. It is a <u>Toyota</u>.)
– The senses of two noun instances are siblings in the hypernym/hyponym tree. (The <u>truck</u> is fast. The <u>car</u> is faster.)

In computing lexical chains, the noun instances were grouped according to the above relations, but each noun instance must belong to exactly one lexical chain. There are several difficulties in determining which lexical chain a particular word instance should join. For example, a particular noun instance may correspond to several different word senses and thus the system must determine which sense to use (e.g. should a particular instance of "house" be interpreted as sense 1: dwelling, or sense 2: legislature). In addition, even if the word sense of an instance can be determined, it may be possible to group that instance into several different lexical chains because it may be related to words in different chains. For example, the word's sense may be identical to that of a word instance in one grouping while having a hypernym/hyponym relationship with that of a word instance in another. What must happen is that the words must be grouped in such a way that the overall grouping is optimal in that it creates the longest/strongest lexical chains. It was observed that words are grouped into a single chain when they are "about" the same underlying concept. That fact confirms the usage of lexical chains in text summarization [Bru01, Zho05, Li07].

Keyphrases, also known as *keywords*, are linguistic units, usually longer than words but shorter than a full sentence. There are several kinds of keyphrases ranging from statistical motivated keyphrases (sequences of words) to more linguistically motivated ones (that are defined in according to a grammar). In keyphrases extraction task, keyphrases are selected from the body of the input document, without a predefined list. Following this approach, a document is treated as a set of candidate phrases and the task is to classify each candidate phrase as either a keyphrase or nonkeyphrase [Dav07]. When authors assign keyphrases without a controlled vocabulary (free text keywords or free index terms), about 70% to 80% of their keyphrases typically appear somewhere in the body of their documents [Dav07]. This suggests the possibility of using author-assigned free-text keyphrases to train a keyphrases extraction system.

D'Avanzo [Dav07] extracts *syntactic patterns* using two ways. The first way focuses on extracting uni-grams and bi-grams (for instance, noun, and sequences of adjective and noun, etc.) to describe a precise and well defined entity. The second way considers longer sequences of part of speech, often containing verbal forms (for instance, noun plus verb plus adjective plus noun) to describe concise events/situations. Once all the uni-grams, bi-grams, tri-grams, and four-grams are extracted from the linguistic pre-processor, they are filtered with the patterns defined above. The result of this process is a set of patterns that may represent the current document.

For multi-document summarization, *passages* are retrieved using a language model [Yin07]. The goal of language modeling is to predict the probability of natural word sequences, or in other words, to put high probability on word sequences that actually occur and low probability on word sequences that never occur. The simplest and most successful basis for language modeling is the *n*-gram model.

I.3.2 Term weighting

Of the works devoted to term-based methods, most concentrate on term weighting. Terms identified in the previous step are scored in order to select the most appropriate terms as representative of the original text.

The use of frequency as a feature in text summarization has been proven useful. Term frequency was first used in extractive text summarization in the late 1950's [Luh57]. Subsequent research using frequency methods focused on the use of frequency as one feature among many for identifying important terms

[Edm69]. Most recently, the SumBasic algorithm uses term frequency as part of a context-sensitive approach to identifying important sentences while reducing information redundancy [Nen05b].

The proposed scoring by D'Avanzo [Dav07] is based on a combination of *tf* x *idf* and first occurrence, i.e. the distance of the candidate term (or specifically key phrases are used as terms) from the beginning of the document in which it appears. However, since candidate phrases do not appear frequently enough in the collection, it has been decided to estimate the values of the *tf* x *idf* using the head of the candidate phrase, instead of the whole term. Every phrase has a single word as head. The head is the main verb in the case of verb phrases, and a noun (last noun before any post-modifiers) in noun phrases. As learning algorithm, it has been used an SVM. The classifier was trained on a corpus with the available keyphrases. From the document collection we extracted all nouns and verbs. Each of them was marked as a positive example of a relevant keyphrase for a certain document if it was present in the assessor's judgment of that document; otherwise it was marked as a negative example. Then the two features (i.e. *tf* x *idf* and first occurrence) were calculated for each word. The classifier was trained using this material and a ranked word list was returned. The system automatically looks in the candidate phrases for those phrases containing these words. The top candidate phrases matching the word output of the classifier are kept. The model obtained is reused in the subsequent steps. When a new document or corpus is ready we use the pre-processor module to prepare the candidate phrases. The model we got in the training is then used to score the phrases obtained. In this case the preprocessing part is the same. So, using the model we got in the training, we extract nouns and verbs from documents, and then we keep the candidate phrases containing them. The system from [Dav07] uses two parameters for controlling its work: one is the maximum number of words allowed in a keyphrase and the second is the maximum number of keyphrases to be extracted from a document.

Nenkova et al. [Nen04, Pas05, Nen06] annotate special terms using the pyramid scheme—a procedure specifically designed for comparative analysis of the content of several texts. The idea of this scheme is to calculate presence of each term in all documents of the collection. The more documents have the term, the more important is this term, and consequently will be included in the summary.

Wei et al. [Wei06] derive relevance of a term from an ontology constructed with formal concept analysis. Song et al. [Son04] basically weight a word basing on the number of lexical connections, such as semantic associations expressed in a thesaurus, that the word has with its neighboring words; along with this, more frequent words are weighted higher. Mihalcea [Mih06] presents a similar idea in the form of a neat, clear graph-based formalism: the words that have closer relationships with a greater number of “important” words become more important themselves, the importance being determined in a recursive way similar to the PageRank algorithm used by Google to weight webpages.

The latter idea can be applied directly to sentence weighting without term weighting: a sentence is important if it is related to many important sentences, where relatedness can be understood as, say, overlap of the lexical contents of the sentences [Mih06]. The two methods presented in [Mih06] are those that currently give the best results and with which we compare our suggested method.

I.3.3 Sentence weighting

Ideally, a text summarization system should “understand” (analyze) the text and express its main contents by generating the text of the summary. For example, Cristea et al. [Cri05] perform sentence weighting according to their proximity to the central idea of the text, which is determined by analysis of the discourse structure.

However, the techniques that try to analyze the structure of the text involve too sophisticated and expensive linguistic processing. In contrast, most of the methods discussed in the literature nowadays represent the text and its sentences as a bag of simple features, using statistical processing without any attempts to “understand” the text.

A very old and very simple sentence weighting heuristic does not involve any terms at all: it assigns highest weight to the first sentences of the text. Texts of some genres—such as news reports or scientific papers—are specifically designed for this heuristic: e.g., any scientific paper contains a ready summary at the beginning. This gives a baseline [DUC] that proves to be very hard to beat on such texts. However, comparing term-based methods with such position-based baseline is not fair in the sense that this baseline only works on text of specific genres (say, it will not work on official documents, email messages, webpages, or literary novels) and uses information (the position of the sentence) not available to term-based methods. It is worth noting that in Document Understanding Conference (DUC) competitions [DUC] only five systems performed above this baseline, which does not demerit the other systems because this baseline is genre-specific. Though the method proposed in this work outperforms very slightly this baseline, such a comparison is unfair.

Another of the possible approaches is *relative utility* [Rad03]. In this approach, all sentences in the input are ranked on a scale from 0 to 10 as to their suitability for inclusion in a summary. In addition, sentences that contain similar information are explicitly marked, so that in the metric evaluation one could penalize for redundancy and reward equally informational equivalent sentences. The ranking of sentences from the entire input allows for a lot of flexibility, because summaries of any size or compression rate can be evaluated. At the same time, the method is applicable only to extractive systems that select sentences directly from the input and do not attempt any reformulation or regeneration of the original journalist-written sentence. The relative utility approach is very similar in essence to the evaluation used by [Mar00], who asked multiple independent subjects to rank the importance of information units. The main difference is that earlier research directly concentrated on subsentential units rather than sentences.

Verma [Ver07] utilizes ontology knowledge for weighting sentences using statistical data of sentences and also parsing and syntax analysis. The disadvantage of this proposal is that was made for only one particular domain.

I.3.4 Sentence selection

Supervised learning methods consider sentence selection as a classification task: they train a classifier using a collection of documents supplied with existing summaries. As features of a sentence such methods can consider text units (in which case we can speak of term selection) or other non-lexical characteristics.

Different lexical and non-lexical features have been used in [Kup95, Chu04, Net04]. Most of these features are "heuristically motivated", since they tend to emulate the manual creation of extracts. In a work of Kupiec [Kup95], the following features were proposed: *sentence position*, *sentence length*, *the presence of key-phrases* and *overlap with the title of the document*. More recent works [Chu04, Net04] extend these features incorporating information about the occurrence of proper names and the presence of anaphors. The "heuristically motivated" features allow extract very precise summaries. However, they have a very big disadvantage of being highly linked to a specific domain. This condition implies that the change for one domain to another, it may be necessary to redefine or even eliminate some features. For instance, keyphrases, which are particular for each domain, require being modified, while the overlap with the title, which has no sense in all topics, may be eliminated.

In order to increase the domain (and language) independence of machine learning summaries, Villatoro [Vil06] eliminates all kind of "heuristically motivated" attributes and substitute them by word-based features. In particular, he uses word sequences (*n*-grams) as terms. Although the first attempt to use *n*-grams is exceeded the results of other methods, it has some disadvantages. One is that they are always sequences of a fixed size, which was previously defined by the user. The big part of the problem in such techniques lies in defining the size of the sequence to be extracted, which usually depends on the analysis of the text.

Chali and Kolla [Cha04, Kol05] presented work on multi-document summarization using lexical chains for sentence selection. In [Cha04], the scoring mechanism only considers the number of occurrences of words within a sentence and within a segment (segment in their work is comparable to single documents within a collection for one topic). No additional information like *n*-grams is used. The obtained results report that sentence scoring based on simple lexical chain counts is not performing enough. As a continuation, the scoring was changed in [Fil07] adding different scoring. One added score was the number of chains passing through a sentence (score-chain) and the other score was based on *n*-grams (scorebigram, scoretrigram). Each occurrence of a chain and bigram increased the score by 1; of a trigram increased the score by 2. The overall score was calculated for each sentence and then used to rank sentences for the final extraction. Using the number of chains passing through a sentence gives higher scores to longer sentences as they are likely to have several chains passing through. But longer sentences also have a higher likelihood to contain more information—especially, if several chains pass through them. The score based on n-grams finds sentences that also have a high score based on identity. It also emphasizes sentences where words from the topic not only occur somewhere in the sentences, but also where words occur in the same order as in the topic sentences.

The approach from [Sek02] is based on weighting approach calculated in the following way. First, the *tf* x *idf* values of all nouns in the document except some stop words are computed. According to each document, the sum of all the *tf* x *idf* values of nouns in the document is computed. The importance value of a sentence is computed by the sum of *tf* x *idf* values of sentences containing nouns divided by the sum of all *tf* x *idf* values in the document. Second, if a sentence contains phrases in the heading, the number of phrases is divided by the total phrases in the heading. That value is then multiplied by the constant 0.1, and adds to the sentence weights. Third, the line number of the sentence in the document divided by the number of all lines in the document corresponds to the position value in the document from 0 to 1. And thus the resultant importance value of each sentence is obtained.

Extractive approaches to text summarization usually follow a model of scoring sentences based on a set of features. The highest scoring sentences are then extracted to form a summary. When using frequency as the only feature, unit items are counted and then each sentence is given a score based on the frequency count of each unit item in the sentence. A key problem in generating summaries is reducing redundancy. Each new sentence in the summary should add new information rather than repeating already included information. Using the highest frequency terms will likely result in the same information repeatedly being selected, with the chance that some additional information is included. In the SumBasic [Nen05b] frequency approach, a probability distribution model is first generated, and as each term is used to select sentences, the term probabilities are reduced so that lower probability terms have a better chance of selecting sentences with new information content. This approach is called context sensitive since the summarizer considers sentences already in the summary before selecting a new sentence to add to the summary. This is also related to the idea of finding Maximal Marginal Relevance (MMR), where marginal relevance is defined as finding relevant sentences which contain minimal similarity to previously selected sentences [Car98].

The frequency distribution algorithm—FreqDist—uses a context sensitive approach to scoring sentences based on a frequency distribution model rather than a probability distribution model [Ree07]. The rationale of the frequency distribution approach is that the frequency distribution of terms or concepts in the source text ought to appear in the generated summary as closely as possible to the source text. That is, the frequency distribution models of the source text and its corresponding summary should be as similar as possible. There are two stages in the algorithm: Initialization and Summary Generation: in the initialization stage, the unit items (terms or concepts) of the source text are counted to form a frequency distribution model of the text, and a pool of sentences from the source text is created, called the sentence pool. A summary frequency distribution model is created from the unit items found in the source text, and its frequency counts are initially set to zero to indicate an empty summary. In the summary generation stage new sentences are evaluated and then selected for inclusion in the summary. Identifying the next sentence to be added to the summary is accomplished by finding the sentence which most closely aligns the frequency distribution of the summary generated so far to the frequency distribution of the original source text. A candidate summary is first initialized to the summary generated so far. For each sentence in the sentence pool, a sentence is added to the candidate summary to see how much it contributes to the candidate summary. To determine the sentence's contribution, the candidate summary frequency distribution is compared for similarity to the source text's frequency distribution. The comparison generates a similarity score assigned to the sentence as the sentence's score. After all sentences from the sentence pool have been evaluated for their contribution to the candidate summary, the highest scoring sentence is added to the summary and removed from the sentence pool. The sentence selection process is iterative, and repeats until the desired length of the summary is reached.

However, the majority of current methods are purely heuristic: they do not use any learning but directly state the procedure used for term selection, term weighting, and/or sentence weighting (given that sentence selection in most cases consists in selecting the best-weighted sentences).

I.4 Abstractive text summarization

Abstractive summarization approaches use information extraction, ontological information, information fusion, and compression. Automatically generated abstracts (abstractive summaries) move the summarization field from the use of purely extractive methods to the generation of abstracts that contain sentences not found in any of the input documents and can synthesize information across sources. An abstract contains at least some sentences (or phrases) that do not exist in the original document. Of course, true abstraction involves taking the process one step further. Abstraction involves recognizing that a set of extracted passages together constitutes something new, that is not explicitly mentioned in the source, and then replacing it in the summary with the (ideally more concise) new concept(s). The requirement that the new material not be in the text explicitly means that the system must have access to external information of some kind, such as an ontology or a knowledge base, and be able to perform combinatory inference.

Different methods are developed in abstractive summarization. For example, techniques of sentence fusion [Dau04, Bar03, Bar05], information fusion [Bar99], sentence compression [Van04, Mad07], etcetera.

I.5 Applications of text summarization

We can find different systems made for the summarization of the following applications:

- Legal texts [Far04, Hac04]
- Emails [Cor04, Shr04, Wan04]
- Web pages [Dia06]
- Web documents using mobile devices [Ott06]
- Figures and graphics [Fut04, Car04, Car06]
- News [Eva05, Mck03, Nen05a]

Finally, we mention some applications where we can use summarization:

1. Selecting scientific papers about the topic we are working on, including single-document summarization for composing the state-of-the-art of a topic.
2. Before we buy a book, generally we read a brief description of it.
3. As students, we prefer to ask the professor for a class's content until we sign in his class.
4. Short description of TV, radio, entertainment programs.
5. Mainly, navigating and looking for the information in internet.

I.6 Research problem

The big number of researches in CL caused the development of more efficient algorithms. This influenced that the usage of text summarization has grown considerably in many of the scientific and business applications in this days. In the great majority of the cases, text summarization methods generate good summaries based on the language-and-domain-dependent techniques, nevertheless these summaries are made generally with a low quality in the content compared to the ability shown by a not trained person. In order to solve this problem, text summarization methods made possible that the user had tools to summarize texts, however a data interchange in many languages and domains increases the necessity of new methods in complex design. The implementation of such types of automatic text summarization methods has become more difficult and eventually impossible due to the natural language complexity.

The limitations caused by the complexity of natural language suggested new methodologies for the design of text summarization methods through the use of language-and-domain-dependent methods. Although such methods denote to be the best option to reduce text dimensions; some of these methods include some specific problems such that much content is not relevant for the general understanding of complete texts. The design of these methods includes the composition of summaries which in the state-of-the-art methods depends on the extraction of the units without considering their importance as descriptors of a text.

In this book we answer the following research question: how to automatically detect the most important parts of a text for composing a summary in a language-and-domain-independent way?

The purpose of the present book is to describe new methods for producing a text summary extracting the terms which describe the most important information of the text. The proposed methods will generate summaries in a language-and-domain-independent way. These methods will include different steps, adjusting each of them for the improvement of the total results. The corpus with manually elaborated summaries is used to exemplify and to verify the effectiveness of the methods.

Chapter II

Theoretical framework

The objective of this chapter is to provide a theoretical framework of this book. We begin with a description of text pre-processing. Section II.2 is dedicated to text representation models. Then, graph algorithms including graph representation of text and graph ranking algorithms are presented in Section II.3. Finally, genetic and clustering algorithms are presented in sections II.4 and II.5 respectively.

II.1 Text pre-processing

The pre-processing step is perhaps the most important in the area of computational linguistics, since the quality of the obtained summary depends on how efficient is the representation of a text. In this book, some experiments will contain the pre-processing stage. Generally, this stage will include only two steps: eliminating stopwords and applying stemming.

II.1.1 Stop-words

When a pre-processing of a text is realized, an intermediate representation of it is obtained. One of the pre-processing stages consists in eliminating stop-words or empty words from the text. There is a set of empty words in every language, common to all domains which are easily identified, for example, articles, prepositions, conjunctions, etc. Also they can be verbs, adjectives and adverbs.

The words that are too frequent in the documents in a particular collection are not good descriptors. In fact, it is considered that a word which appears in at least 80% of the documents of a particular collection is useless for purpose of retrieval. These words are considered empty and normally are removed to avoid being considered as potential.

Later, we will realize a process of extraction of stop-words in the documents with the aim of reducing the content of the text to more specific expressions (we call them multiword descriptions), containing only the words that are useful and meaningful for the generation of automatic summaries.

II.1.2 Stemming

Stemming technique consists in obtaining the root of words, so that the text processing is conducted on the roots and not on the original words. This technique allows us to relate more terms in the document. It is supposed that two words with the same root represent the same concept. Basically, the process of stemming of the words is realized for reducing a word to a minimum common portion of called stem. The stem is the portion of the word which is left after the removal of its affixes, prefixes and suffixes. Once implemented stemming, the document will contain only the roots of the words. This will simplify the representations of the documents using the models mentioned above and clustering methods.

The first stemming algorithm was developed for the English language, and then was adapted for the Spanish language. The algorithm Porter [Por80] is the most commonly used for the English language. Also there are algorithms for other languages such as French, Dutch, Greek and Latin. In general, these algorithms are based on a simple set of rules that cut off words to obtain a common root [Bae99].

II.2 Text representation models

Text representation models are techniques based on the extraction of terms of a text or document which consist in choosing terms that will be extracted, and then turned into terms. The difference between models is the type of terms that are extracted from a document. In this book, three models are considered: bag of words (proposed by Salton in 1975), *n*-grams and MFSs.

II.2.1 Bag of words

The representation with bag of words consists in obtaining all different words which appear in a text. Subsequently, the document is shown as a vector, where each position will contain a term, and each term will correspond to a word in the document.

II.2.2 *N*-grams

The *n*-gram model follows the same principle that the model based on bag of words, likewise the text is represented as a vector of terms. The difference is that the size of *n*-gram is previously defined, i.e. *n* is the number of consecutive elements that contain the term. These elements can be words or characters. For example, if *n* equals 2, the defined term will contain 2 words or characters, namely bigrams.

Observe that in the *n*-gram model, extracted elements not completely preserve the order they appear in the text. In addition, we find another disadvantage common to both models: high dimensionality. Clearly, even with a small document, you have a considerable amount of different characteristics to evaluate, which means an enormous expense of resources to handle such amount of information. Trying to solve the problems of sequential order and dimensionality of models, MFSs has been proposed to use as a model representation of the text.

II.2.3 MFSs

Frequent Sequence (FS) is a sequence of words or characters that appear in a text for the repeated manner. A sequence is called a Maximal Frequent Sequence (MFSs) if it is not contained in another FS. MFSs model determines the number of times the FS will be repeated in the text to be considered frequent. This number is called the threshold. For example, for the document presented in Figure II.1, boolean model based on MFSs is shown, taking as a threshold equal to 2. This means that each sequence must be appeared at least 2 times in the document to be frequent. Table II.1 shows the representation of boolean MFSs model.

As shown in Table II.1, the number of terms for MFSs model is reduced considerably comparing to bag of words and *n*-grams models. Tables II.2, II.3, II.4, and II.5 show models based on MFSs using 4 different weightings.

Figure II.1
Example of 5 sentences from an arbitrary text

… *El gobierno de Egipto protege las pirámides…*
… *Las pirámides de Egipto son un patrimonio cultural…*
… *Las pirámides fueron construidas por los faraones…*
… *Las pirámides de Egipto fueron tumbas para los faraones de Egipto...*
… *Un buen gobierno protege su patrimonio cultural...*

Table II.1
Boolean representation of MFSs model for the Figure II.1

MFSs	1	2	3	4	5
gobierno	1	0	0	1	0
las pirámides de Egipto	0	1	0	0	1
patrimonio cultural	0	1	0	1	0
los faraones	0	0	1	0	1

Table II.2
Representation of MFSs model with Boolean weighting

MFSs	1	2	3	4	5
gobierno	1	0	0	1	0
las pirámides de Egipto	0	1	0	0	1
patrimonio cultural	0	1	0	1	0
los faraones	0	0	1	0	1

Table II.3
Representation of MFSs model with *tf* weighting

MFSs	1	2	3	4	5
gobierno	1	0	0	1	0
las pirámides de Egipto	0	1	0	0	1
patrimonio cultural	0	1	0	1	0
los faraones	0	0	1	0	1

Table II.4
Representation of MFSs model with *idf* weighting

MFSs	1	2	3	4	5
gobierno	0.397	0	0	0.397	0
las pirámides de Egipto	0	0.397	0	0	0.397
patrimonio cultural	0	0.397	0	0.397	0
los faraones	0	0	0.397	0	0.397

Table II.5
Representation of MFSs model with *tf-idf* weighting

MFSs	1	2	3	4	5
gobierno	0.397	0	0	0.397	0
las pirámides de Egipto	0	0.397	0	0	0.397
patrimonio cultural	0	0.397	0	0.397	0
los faraones	0	0	0.397	0	0.397

As can be seen, in some cases the tables are identical, it can be explained by simplicity of the example, the values do not always change with respect to each sentence from one type weighting to another.

II.2.4 Term weighting

Within vector space models, there is another way of representing a document which is denoted as term weighting. Term weighting consists in assigning a weight for each term which reflects the importance of the term in the document. Below in this section four different term weighting options are described.

Boolean Weighting. It is the easiest way to weigh a term. It has 1, if it appears in the document and 0 in another case.

$$P_i(t_j) = \begin{cases} 0, & if \quad appeared \\ 1, & other \quad case \end{cases} \tag{2.1}$$

where $p_i(j)$ is term frequency j in the document i.

Term Frequency (tf) was proposed in [Luh57]. This weighting takes into account that a term that frequently occurs in a document can better reflect the contents of the document than a term that occurs less frequent. Therefore, the weighting tf assigns a greater weight to terms with greater frequency and consists in evaluating the number of times when the word occurs in the document.

$$P_i(t_j) = f_{ij} \tag{2.2}$$

where f_{ij} is term frequency j in the document i.

Inverse Document Frequency (*idf*) was proposed in [Sal88]. Taking into account the observation that a very frequent term appeared in several documents is less useful than a term that is appeared less frequent, because evaluates the distribution of terms in the document. The inverse frequency of the document is defined as

$$P_i(t_j) = \log \left(\frac{N}{nj}\right) \qquad (2.3)$$

where f_{ij} is term frequency j in the document i; N is the number of the documents in the collection; n_j is the number of documents where the term j appears.

tf-idf weigthing. It is common that term frequency (*tf*) and inverse term frequency (*idf*) of the document are used together in order to determine the weight of each term in the vector space model [Sal88]. This combination is known as weighting *tf-idf* and consists of multiplying the frequency of the term by the inverse frequency of the documents where appears this term.

$$P_i(t_j) = f_{ij} \text{ x } \log\left(\frac{N}{n_j}\right) \qquad (2.4)$$

Note that in this method, if we are working with a single document, we will take N as the number of sentences and n_j as the number of sentences where the term appears.

II.3 Graph algorithms

Many language processing applications can be modelled by means of a graph. These data structures have the ability to encode in a natural way the meaning and structure of a cohesive text, and follow closely the associative or semantic memory representations.

TextRank [Mih06] has been successfully applied to three natural language processing tasks: document summarization, word sense disambiguation, and keyword extraction, with competitive results to those of state-of-the-art systems. The strength of the model lies in the global representation of the context and its ability to model how the co-occurrence between features might propagate across the context and affect other distant features. The description of application of random-walks to text processing, as done in the TextRank system, is given below.

II.3.1 Graph representation of text

To enable the application of graph-based ranking algorithms to natural language texts, a graph that represents the text is built, and interconnects words or other text entities with meaningful relations. The graphs constructed in this way are centred around the target text, but can be extended with external graphs, such as off-the-shelf semantic or associative networks, or other similar structures automatically derived from large corpora.

Graph Nodes: Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, word senses, entire sentences, entire documents, or others. Note that the graph-nodes do not have to belong to the same category.

Graph Edges: Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, measures of text cohesiveness, contextual overlap, membership of a word in a sentence, and others.

Algorithm: Regardless of the type and characteristics of the elements added to the graph, the application of the ranking algorithms to natural language texts consists of the following main steps:

- Identify text units that best define the task at hand, and add them as vertices in the graph.
- Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
- Apply a graph-based ranking algorithm to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

II.3.2 Graph ranking algorithms

The basic idea implemented by a random-walk algorithm is that of "voting" or "recommendation." When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex.

Moreover, the vertex casting a vote determines how important the vote itself is, and this information is also taken into account by the ranking algorithm. While there are several random-walk algorithms that have been proposed in the past, we focus on only one such algorithm, namely PageRank [Bri98], as it was previously found successful in a number of applications, including Web link analysis, social networks, citation analysis, and more recently in several text processing applications.

Given a graph $G = (V, E)$, let $In(V_i)$ be the set of vertices that point to vertex V_i (predecessors), and $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The PageRank score associated with the vertex V_i is defined using a recursive function that integrates the scores of its predecessors:

$$S(V_I) = (1-d) + d^* \sum_{Vj \in In(V_i)} \frac{S(V_j)}{|Out(V_j)|} \tag{2.5}$$

where d is a parameter that is set between 0 and 1.

The score of each vertex is recalculated upon each iteration based on the new weights that the neighboring vertices have accumulated. The algorithm terminates when the convergence point is reached for all the vertices, meaning that the error rate for each vertex falls below a pre-defined threshold.

This vertex scoring scheme is based on a random-walk model, where a walker takes random steps on the graph, with the walk being modelled as a Markov process. Under certain conditions (the graph is aperiodic and irreducible), the model is guaranteed to converge to a stationary distribution of probabilities associated with the vertices in the graph. Intuitively, the stationary probability associated with a vertex represents the probability of finding the walker at that vertex during the random-walk, and thus it represents the importance of the vertex within the graph.

HITS (Hyperlinked Induced Topic Search) [Kle99] is an iterative algorithm that was designed for ranking Web pages according to their degree of "authority". The HITS algorithm makes a distinction between "authorities" (pages with a large number of incoming links) and "hubs" (pages with a large number of outgoing links). For each vertex, HITS produces two sets of scores: an "authority" score, and a "hub" score:

$$HITS_A(V_i) = \sum_{V_j \in In(V_i)} HITS_H(V_j) \tag{2.6}$$

$$HITS_H(V_i) = \sum_{V_j \in In(V_i)} HITS_A(V_j) \tag{2.7}$$

PageRank [Bri98] is perhaps one of the most popular ranking algorithms, and was designed as a method for Web link analysis. Unlike other ranking algorithms, PageRank integrates the impact of both incoming and outgoing links into one single model, and therefore it produces only one set of scores:

$$PR(Vi) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{Out(V_j)} \tag{2.8}$$

where d is a parameter that can be set between 0 and 1. In matrix notation, the PageRank vector of stationary probabilities is the principal eigenvector for the matrix A_{row}, which is obtained from the adjacency matrix A representing the graph, with all rows normalized to sum to 1: $P = A^T{}_{row}\ P$.

A ranking process starts by assigning arbitrary values to each node in the graph, followed by several iterations until convergence below a given threshold is achieved. Convergence is achieved when the error rate for any vertex in the graph falls below a given threshold, where the error rate of a vertex V_i is approximated with the difference between the scores computed at two successive iterations: $S^{k+1}(V_i) - S^k(V_i)$ (usually after 25-35 iteration steps). After running the algorithm, a score is associated with each vertex, which represents the "importance" (rank) of the vertex within the graph. Note that for such iterative algorithms, the final value obtained for each vertex is not affected by the choice of the initial value only the number of iterations to convergence may be different.

Undirected Graphs: Although traditionally applied on directed graphs, algorithms for node activation or ranking can be also applied to undirected graphs. In such graphs, convergence is usually achieved after a larger number of iterations, and the final ranking can differ significantly compared to the ranking obtained on directed graphs.

Weighted Graphs: When the graphs are built from natural language texts, they may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices. Consequently, we introduce new formulae for graph-based ranking that take into account edge weights when computing the score associated with a vertex in the graph, e.g.

$$PR^{w}(V_i)=(1\text{-}d)+d*\sum_{V_j\in In(V_i)} w_{ij}\frac{PR^{W}(V_j)}{\sum_{V_k\in Out(V_j)} w_{jk}} \tag{2.9}$$

II.4 Genetic algorithms

A Genetic Algorithm (GA) uses the principles of evolution, natural selection, and genetics from natural biological systems in a computer algorithm to simulate evolution [Gol89]. Essentially, the GA is an optimization technique that performs a parallel, stochastic, but directed search to evolve the fittest population.

GAs encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. GAs are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad. The more common applications of GAs are the solution of optimization problems, where efficient and reliable results have been shown. That is the reason why we will use these algorithms to find parameters for the rule base reduction methods.

In the early 1970's, John Holland introduced the concept of genetic algorithms. His aim was to make computers do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits. Each artificial "chromosome" consists of a number of "genes" and each gene is represented by 0 or 1:

1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	1

Nature has an ability to adapt and learn without being told what to do. In other words, nature finds good chromosomes blindly. GAs do the same. Two mechanisms link a GA to the problem it is solving: encoding and evaluation. The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome.

II.4.1 Basic genetic algorithm

The basic genetic algorithm consists of following steps [Neg02]:

Step 1: Represent the problem variable domain as a chromosome of a fixed length; choose the size of a chromosome population N, the crossover probability P_c, and the mutation probability P_m.
Step 2: Define a fitness function to measure the performance or fitness of an individual chromosome in the problem domain. The fitness function establishes the basic for selecting chromosome that will be mated during reproduction.
Step 3: Generate an initial population of chromosome of size N: $x_1, x_2, ..., x_N$.
Step 4: Calculate the fitness of each individual chromosome: $f(x_1)$, $\mathrm{f}(x_2)$, ... $\mathrm{f}(x_N)$.
Step 5: Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness.
Step 6: Create a pair of offspring chromosomes by applying the genetic operators—crossover and mutation.
Step 7: Place the created offspring chromosomes in the new population.
Step 8: Repeat Step 5 until the size of the new chromosome population becomes equal to the size of the initial population N.
Step 9: Replace the initial (parent) chromosome population with the new (offspring) population.
Step 10: Go to Step 4, and repeat the process until the termination criterion is satisfied.

GA represents an iterative process. Each iteration is called a generation. A typical number of generations for a simple GA can range from 50 to over 500. The entire set of generations is called a run. Because GAs use a stochastic search method, the fitness of a population may remain stable for a number of generations before a superior chromosome appears. A common practice is to terminate a GA after a specified number of generations and then examine.

II.4.2 Representation, population, and fitness function

Representation

Before applying a GA we first must encode the parameters of the problem to be optimized. GAs do not deal directly with the parameters, they work with codes that represent the parameters. Thus, the representation of the problem is the first important issue in the design of genetic algorithms, i.e., how to represent the problem parameters.

Different representation schemes might cause different performances in GAs [Cha99, Hau04, Mel99]. There are two common representation methods that we can use: *floating point* and *bit string*. The preferred method is the binary string because the majority of genetic operators are suitable for this type of representation, and also, this representation has a better impact in the performance of genetic algorithms. In binary representation of GAs each parameter to optimize is encoded using a binary string of a fixed length, so we need to find a codification function that maps a real parameter value into an integer in the interval [0, 2] where/is the length of the binary string. To construct such a function, we usually first decide the range of each parameter value based on background knowledge of the problem whose parameters we want to optimize. Based on the range and desired precision of the optimal value for each parameter we can calculate the length of the binary string required. The role of the codification function and its inverse (decodification function) is encoding and decoding a space of values (possible solutions) for a parameter, such that we can pass from real parameter values to a binary string that can be used by GAs.

Population

Genetic algorithms operate with a population of possible solutions, not only one, so, at the beginning, a GA requires an initial population of individuals. The size of the initial population can be fixed, or depending of the algorithm, this can be adaptive. There are three ways of forming the initial population: *randomly*, *deterministic* and *by help of other methods*. The first methods generate solutions randomly. The second initializes the population with specified chromosomes, for instance, only chromosomes of 0's, 1's, and so on [Una05]. Also knowledge of the problem can be used and obtain solutions that satisfy certain requirements. Finally, the initial population can be also initialized with individuals proportioned by other optimization techniques.

Fitness evaluation function

The fitness of an individual in genetic algorithms is the value returned by the fitness evaluation function that measures the fitness or quality of chromosomes to solve a problem. Obviously, the fitness of chromosomes less fit to solve a problem are more punished than the fitness of fitter chromosomes.

The fitness evaluation function acts as an interface between the genetic algorithm and the optimization problem. First, the chromosome must be decoded, and then evaluated by the fitness function which returns a value indicating the fitness of chromosomes to solve the problem. Fitness evaluation function plays an important role in GA because it provides information about how good a solution performs to solve the problem. This information guides the search of a genetic algorithm, and more accurately, the fitness evaluation function results to determine the likelihood that a possible solution is selected to produce new solutions in the next generation.

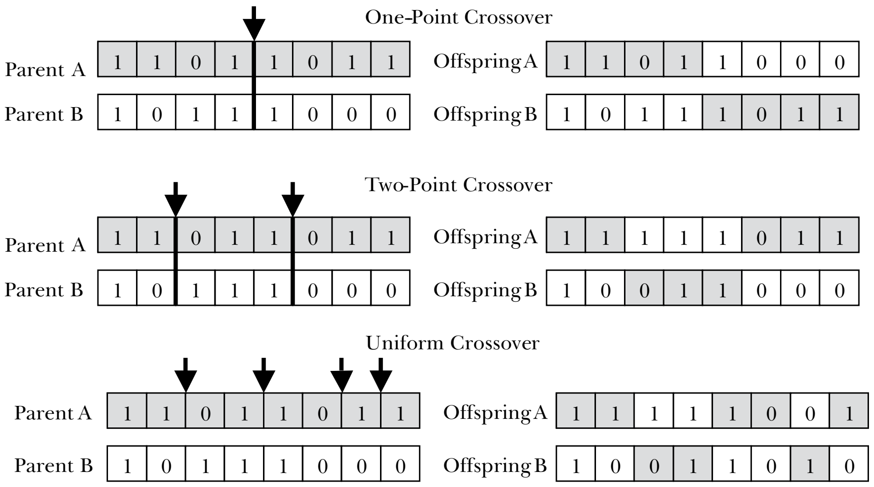
II.4.3 Genetic operators

Crossover Operator

Crossover is a genetic operator that combines two chromosomes (parents) to produce one or two chromosomes (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. First, the crossover operator randomly chooses a crossover point where two parent chromosomes "break", and then exchanges the chromosome parts after that point with a user-definable crossover probability. As a result, two new offspring are created. If a pair of chromosomes does not cross over, then the chromosome cloning takes place, and the offspring are created as exact copies of each parent [Neg02].

The most common forms of crossover are one-point, two-point, *n*-point, and uniform crossover showed in Figure II.2.

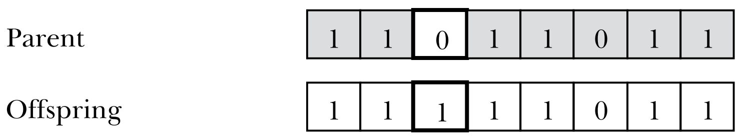
Figure II.2
Crossover operators



Mutation operator

Mutation represents a change in the gene (Figure II.3). Its role is to provide and guarantee that the search algorithm is not trapped on a local optimum. The mutation operator flips a randomly selected gene in a chromosome. The mutation operator uses a mutation probability p_m previously set by the user, which is quite small in nature, and it is kept low for GAs, typically in the range 0.001 and 0.01. According to this probability, the bit value is changed from 0 to 1 or vice versa. This way, an offspring is produced from a single parent [Neg02].

Figure II.3
Mutation operator



The following three operators compose the CHC (Cross-generational elitist selection, Heterogeneous recombination by "incest prevention", and Cataclysmic mutation) algorithm which has the idea that recombination should be the dominant search operator.

Elitist selection and incest prevention

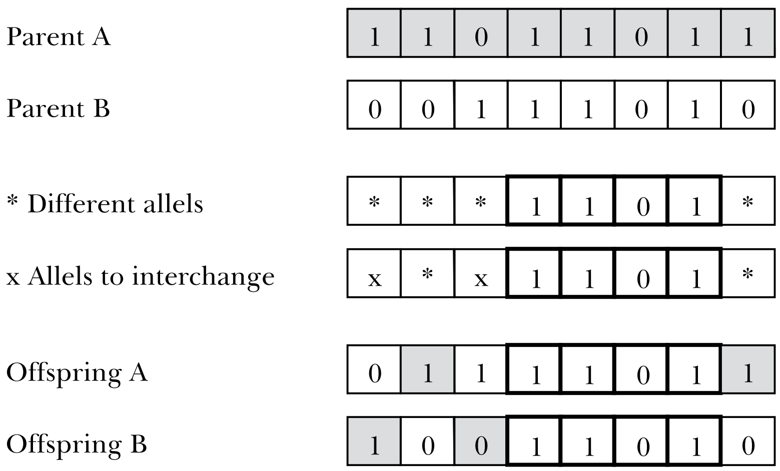
After recombination, the *N* best unique individuals are drawn from the parent population and the offspring population to create the next generation. This also implies that duplicate individuals are removed from population. This form of selection is also referred to as *truncation selection* [Esh91].

After the truncation selection, pairs of individuals are randomly formed with the new parent population to apply the recombination, forming *N/2* pairs of individuals. However, the CHC algorithm also employs a heterogeneous recombination restriction as a method of "*incest prevention*". This is accomplished by matting only those pairs of chromosomes which differ from each other by some number of bits, i.e., a *matting threshold*. The initial threshold is set at *L/4*, where *L* is the length of the string. If any of the *N/2* recombination could not be applied, i.e., if a generation occurs in which no offspring are inserted into the new children population, then the threshold is reduced by one. This means that the chromosomes of the population have become very similar.

Half uniform crossover (HUX) operator

In this operator bits are randomly and independently exchanged, but exactly half of the bits that differ between parents are swapped, see Figure II.4. The HUX operator [Esh91] ensures that the offspring are equidistant between the two parents. This serves as a diversity preserving mechanism.

Figure II.4
Half uniform crossover



Cataclysmic mutation

No mutation is applied during the regular search phase of the CHC algorithm [Esh91]. When no offspring can be inserted into the population of a succeeding generation, and the mating threshold has reached a value of zero, CHC introduces new diversity into the population via a form of restart. Cataclysmic mutation uses the best individual in the population as a template to re-initialize the population. The new population includes a copy of the best individual; the remainder of the population is generated by applying a simple mutation relatively high, for instance

35%, of the best individual. The new threshold value will be the product of the chromosome's length (L) and the mutation's percentage (%) used to generate the new population. There are many other ways to refresh the population, for example, to rescue the best k individuals and generating the remainder randomly, or to rescue the best k individuals and use these as templates to generate the remaining of the population, and so on.

II.5 Clustering algorithms

The clustering algorithms form groups of objects in order to archive the greatest possible similarity between objects of a group, and at the same time to keep the dissimilarity of the objects of other groups. More formally, the clustering problem [Jai99] is dividing a given set $\{x_1, ..., x_N\}$ of N data points into several non-overlapping homogenous groups. Each such group or cluster should contain similar data items and data items from different groups should not be similar. We refer to a clustering in k groups as a k-clustering.

Many different approaches to the clustering problem have been developed. Some operate on data represented by their coordinates in a feature space and others operate on a matrix of pairwise similarities between data points. To give a briefly overview of the different types of methods, we list them in three groups [Ver04]:

1. Hierarchical clustering methods. These produce a hierarchy of clusters for the data. The first level of the hierarchy contains all data and at each subsequent level of the hierarchy, one of the clusters of the previous level is split in two. The last level contains all data in individual clusters. The hierarchy is based on pairwise similarities between data points and is constructed either top-down or bottom-up.
2. Partitional clustering methods. These produce a single clustering with a fixed and (often) specified number of clusters. Most partitional clustering algorithms do not operate on the basis of pairwise similarities, but with data represented in some feature space. Typically, these methods start with an initial k-clustering and apply an iterative algorithm to improve upon the initial clustering according to some criterion. Most partitional clustering methods make, sometimes implicitly, assumptions on the distribution of data within each cluster. The partitional algorithms CURE, k-means [Har79] is used in this method.
3. Spectral clustering methods. These operate on a matrix with pairwise similarities between the data points. The optimal clustering is defined as the clustering that minimizes the 'normalized cut' criterion that depends on the size of the

clusters and the total sum of the similarities between points that are assigned to different clusters. Unfortunately, finding the clustering that minimizes the normalized cut is an NP-complete problem). However, a relaxation of this optimization problem can be efficiently solved, and the solution is given by an eigenvector of the normalized similarity matrix. The solution of the relaxed problem is then further processed to find an approximate solution for the original problem. The term spectral clustering' refers to the normalized similarity matrix which can be used to assess the number of clusters in the data. Spectral methods are used both to find hierarchical clustering and k-clustering for a given k. We treat them separately since their working is quite different from the other approaches.

II.5.1 Algorithm *k*-means

The k-means algorithm [Har79] clusters n objects based on attributes into k partitions, $k < n$. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data. It assumes that the object attributes form a vector space. The objective it tries to achieve is to minimize total intra-cluster variance, or the squared error function

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{2.10}$$

where there are k clusters S_i, $i = 1, 2, ..., k$, and μ_i is the centroid or mean point of all the points $x_j \in S_i$.

Chapter III

New methods for automatic single text summarization

The objective of this chapter is to present the new methods for generation of summaries for single document. In the section III.1 we present the first method for the generation of text summaries for a single document. Different term selection, term weighting, sentence weighting and sentence selection methods are described. Another interesting results to generate summaries are described using the graph algorithms (section III.2), which considerably improves the results for the generation of text summaries. Finally, some general ideas of genetic and clustering algorithm are presented.

III.1 Definitions

III.1.1 Sequences of *n*-grams

An *n*-gram is a sequence of *n* words. We say that an *n*-gram occurs if these words appear in the text in the same order immediately one after another. For example, a 4-gram (*n*-gram of length 4) *words appear in the text occurs* once in the previous sentence, while *appear immediately after another* does not (these words do not appear on adjusting positions), neither does *the text appear in* (order is different).

The definition of *n*-gram depends on what one considers words. For example, one can consider capitalized (*Mr. Smith*) and non-capitalized (*a smith*) words as the same word or as different words; one can consider words with the same morphological stem (*ask, asked, asking*), the same root (*derive, derivation*), or the same meaning (*occur, appear*) as the same word; one can omit the stop-words (*the, in*) when counting word positions, etc. Say, one can consider that in our example sentence above there occur the *n*-grams *we say* (capitalization ignored), *word appear* (plural ignored), *appear text* (*in the* ignored). This can affect counting the *n*-grams: if one considers occur and appear as equivalent and ignores the stop-words, then in our example sentence the bigram *appear text* occurs twice.

III.1.2 Frequent sequences

We call an *n*-gram frequent (more accurately, β-frequent) if it occurs at least β times in the text, where β is a predefined threshold. Frequent *n*-grams—we will also call them Frequent Sequences (FSs)—often bear important semantic meaning: they can be multiword expressions (named entities: *The United States of America*, idioms: *kick the basket*) or otherwise refer to some idea important for the text (*the President's speech*, *to protest against the war*).

In frequent sequential pattern mining, a sequence is extracted if it is repeated frequently in a collection of documents. For example, the following frequent sequences were obtained from five sentences (see Figure III.1) with β = 2 showed in Figure III.2.

Figure III.1
Example of 5 sentences from an arbitrary text

… *El gobierno de Egipto protege las pirámides…*
… *Las pirámides de Egipto son un patrimonio cultural…*
… *Las pirámides fueron construidas por los faraones…*
… *Las pirámides de Egipto fueron tumbas para los faraones de Egipto...*
… *Un buen gobierno protege su patrimonio cultural...*

The discovery of frequent sequences, as has been commonly used in sequential pattern mining, would not be very useful in analyzing texts because the number of frequent sequences that appear in a given text is very big. In Figure III.2, frequent sequences are shown for β = 2. In this example, we obtained 19 frequent sequences.

Figure III.2
Frequent sequences extracted from the example of Figure III.1 with β = 2

1. "gobierno"
2. "de"
3. "Egipto"
4. "protege"

5. *"las"*
6. *"pirámides"*
7. *"un"*
8. *"patrimonio"*
9. *"cultural"*
10. *"fueron"*
11. *"los"*
12. *"faraones"*
13. *"de Egipto"*
14. *"las pirámides"*
15. *"pirámides de"*
16. *"patrimonio cultural"*
17. *"los faraones"*
18. *"las pirámides de Egipto"*
19. *" patrimonio cultural"*

III.1.3 Maximal Frequent Sequences

One way to reduce all frequent sequences is to take into account only those sequences that are not only frequent subsequences but in addition are maximal. FSs that are not parts of any other FS are called Maximal Frequent Sequences (MFSs) [Gar04, Gar06]. For the example of Figure III.1, we obtained sequences but only four of them are maximal. In this sense, one of the properties of MFSs is that all those subsequences that can be formed MFS are also frequent. In other words, this represents that MFS contains any frequent subsequence, thus bearing a compact representation.

Figure III.3
MFSs for the example of Figure III.1 with β = 2, GAP = 0

1. *" gobierno"*
2. *" las pirámides de Egipto"*
3. *" patrimonio cultural"*
4. *" los faraones"*

Although MFSs "gobierno", "las pirámides de Egipto", "patrimonio cultural", "los faraones" (see Figure III.3) have consistency, in practice it may not happen, because there is no restriction in the separation in the text between the words that form a sequence. The restriction in the separation is called GAP restriction. Thus, for small values of GAP we can find more understandable patterns and the context of words maintains. For example, in Figure III.4, MFSs with a threshold β = 3 and a GAP equal to 0 are presented.

Figure III.4
MFSs for the example of Figure III.1 with β= 3, GAP = 0

1. "las pirámides"
2. "de Egipto"

In the case of the discussed example, we obtain the same MFSs as in Figure III.4 with β = 4 and GAP = 0. For example, with β = 2 and GAP = 2, we obtain the following MFSs showed in Figure III.5.

Figure III.5
MFSs for the example of Figure III.1 with β= 2, GAP = 2

1. "gobierno protege"

The MFS from Figure III.5 was obtained from sequences 1 and 4 in the way showed in Figure III.6 with the words which are underlined. As we can see, there is a separation between words (two words as a given restriction).

Figure III.6
Sequences utilized for obtaining MFS from Figure III.5

… *El <u>gobierno</u> **de Egipto** <u>protege</u> las pirámides…*
… *Las pirámides de Egipto son un patrimonio cultural…*
… *Las pirámides fueron construidas por los faraones…*
… *Las pirámides de Egipto fueron tumbas para los faraones de Egipto...*
… *Un buen* <u>gobierno protege</u> *su patrimonio cultural...*

One of our hypotheses was that only MFSs should be considered as bearing important meaning, while non-maximal FSs (those that are parts of another FS) should not be considered. Our additional motivation was cost vs. benefit considerations: there are too many non-maximal FSs while their probability to bear important meaning is lower. In any case, MFSs represent all FSs in a compact way: all FSs can be obtained from all MFSs by bursting each MFS into a set of all its subsequences. Garcia [Gar04] proposed an efficient algorithm to find all MFSs in a text, which we also used to efficiently obtain and store all FSs of the document.

The notions of FSs and MFSs are closely related to that of repeating bigrams; see Section III.1.2. This set is conceptually simpler, but for computational implementation MFSs could be more compact.

We should note that MFSs can loss a property to be maximal if the threshold increases. For example, we have the set of MFSs with threshold β. Then with β + 1, we can have more MFSs but the set of derived words will be the subset of words of MFSs with threshold β. Despite we have more MFSs with β + 1, MFSs will be shorter (have a less length).

III.1.4 Multiword descriptions

Our hypothesis is that FSs can express ideas both important and specific for the document. This can be argued in terms of *tf-idf* (term frequency-inverse document frequency, a notion well-known in information retrieval [Bae99]): on the one hand, the idea expressed by an FS is important for the document if it repeatedly returns to it (high term frequency); on the other hand, the corresponding idea should be specific for this document, otherwise there would exist in the language a single word or at least an abbreviation to express it (high inverse document frequency).

An *n*-gram can be a part of another, longer *n*-gram. All *n*-grams contained in an FS are also FSs. However, with the arguments given above one can derive that such smaller *n*-grams may not bear any important meaning by their own: e.g., *The United States of America* is a compound named entity, while *The United* or *States of America* are not. Exceptions like *The United States* should not affect much our reasoning since they tend to be synonymous to the longer expression, and the author of the document would choose one or another way to refer to the entity, so they should not appear frequently both in the same document.

III.2 New method using maximal frequent sequences

We describe the general scheme of the proposed algorithm which consists of the next four steps.

III.2.1 Term selection

When we say the terms, we refer to the features which we use in this step. The terms are words, *n*-grams, or MFS's extracted from a document. The details of

MFSs are described below. Also we extract terms derived from MFSs such as words and n-grams. Namely, we propose the following variants of term selection:

- M: the set of all *MFSs*, i.e., an n-gram $m \in M$ if it is an MFS with some threshold β (recall that MFSs are of 2 words or longer and $\beta \geq 2$). In the example from Figure III.7, M = {*is the most beautiful, the most beautiful*}. Also, we denote by M_2 the set of all MFSs with $\beta = 2$.
- B: repeating bigrams, i.e., bigrams with frequency at least 2. It is easy to show that it is the same set as the set of all bigrams from MFSs: a bigram $b \in B$, if there exists an MFS $m \in M$ such that $b \subseteq m$. What is more, considering in the latter definition M_2 instead of M also gives the same set. In our example, B = {*is the, the most, most beautiful*}.
- W: single words (unigrams) from elements of B or, which is the same, of M. Namely, a word $w \in W$ if there exists a bigram $b \in B$ such that $w \in b$; it is easy to show that $w \in W$ if there exists an MFS $m \in M$ such that $w \in m$. Again, considering M_2 in the latter definition also gives the same set. In our example, B = {*is, the, most, beautiful*}.
- N: all n-grams from MFSs, i.e., an n-gram $n \in N$ if there exists an MFS $m \in M$ such that $n \subseteq m$ (including single words, i.e., 1-grams). Again, considering in the latter definition M_2 also gives the same set, which allows for efficient calculation of the set N in practice. In our example, N = {*is, the, most, beautiful, is the, the most, most beautiful, is the most, the most beautiful, is the most beautiful*}. Note that $W \subset N$, $M \subset N$.
- $N \setminus W$, $N \setminus M_2$, $N \setminus (W \cup M_2)$: same as N but not including 1-grams, the whole MFS, or both; here M_2 is the set of MFSs with $\beta = 2$. In our example, $N \setminus (W \cup M_2)$ = {*is the, the most, most beautiful, is the most, the most beautiful*}.

Figure III.7
Example of 4 sentences from an arbitrary text

1. … Mona Lisa is the most beautiful picture of Leonardo da Vinci…
2. … Eiffel tower is the most beautiful tower…
3. … St. Petersburg is the most beautiful city of Russia…
4. … The most beautiful church is not located in Europe…

We give different definitions of the sets B and W to show that they are naturally derived from the notion of MFS and at the same time can be efficiently calculated.

III.2.2 Term weighting

We propose a scheme for weighting MFS which take into account T_i frequency of MFS, length of MFS, and frequency of derived terms from MFS. The terms T_i can be weighted in different manners, and have a weight t_i. This general scheme is defined as $p_i(t_j) = X \cdot Y$, where $p_i(t_j)$-term weighting j in the documents i, X and Y can be determined as frequency of MFS, length of MFS, and frequency of derived terms from MFS. This term weighting scheme permits to detect which of the characteristics of MFS helps better to summarize a text. Specifically, the following term weighting schemes are proposed:

- *f*: frequency of the term in MFSs, i.e., the number of times the term occurs in the text within some MFS. In our example, *f*(*is*) = 3 since it occurs 3 times in the text within the MFS is the most beautiful. If the term itself is an MFS, then this is just the frequency of this term in the text (e.g., for *M*, *f* is the same as term weight in Experiment 1; for *W* and *N* it is not). Under certain realistic conditions (MFSs do not intersect in the text, words do not repeat within one MFS) *f* is the number of times the term occurs in the text as part of a repeating bigram. In our example, *f*(*is*) = 3 since it occurs 3 times in a repeating bigram *is the* (and one time in a non-repeating context *church is not*).
- *l*: the maximum length of an MFS containing the term. In our example, *l*(*is*) = 4 since it is contained in a 4-word MFS *is the most beautiful*.
- 1: the same weight for all terms.

III.2.3 Sentence weighting

For this stage, we calculate the sum of the weights of the terms contained in the sentence. When a sentence S_i has weight s_j = sum w_{ij}, contribution of T_i in S_j is $w_{ij} = f_{ij} \cdot t_i$, where f is a presence of T_i in D_j, t is an importance of T_i. Here f is binary.

III.2.4 Sentence selection

This procedure completes a summary adding the densest sentences or choosing the position of a sentence in a text until the summary is limited by the number of words. As the first option, we chose the sentences which have more weighting score. This type of methods is domain-independent and can be applied for a variety of texts. As the second option, the type of methods is position dependent and can be applied only for special topics. These two options are resumed as follows:

- best: sentences with greater weight were selected until the desired size of the summary (100 words) is reached. This is the most standard method.
- *k*best+first: *k* best sentences were selected, and then the first sentences of the text weight were selected until the desired size of the summary is reached. This was motivated by the very hard-to-beat baseline mentioned in section IV.1: only the very best sentences according to our weighting scheme might prove to be above this baseline.

III.3 New method using graph algorithms

III.3.1 Term selection

The main contribution of this method is the proposal of using multiword descriptions as nodes of a graph.

- Term selection: *M, W.*

III.3.2 Term weighting

Term weighting: frequency of the term in MFSs (*f*); the maximum length of an MFS containing the term (*l*); the same weight for all terms (1).

III.3.3 Sentence weighting

Sentence weighting: using PageRank.

III.3.4 Sentence selection

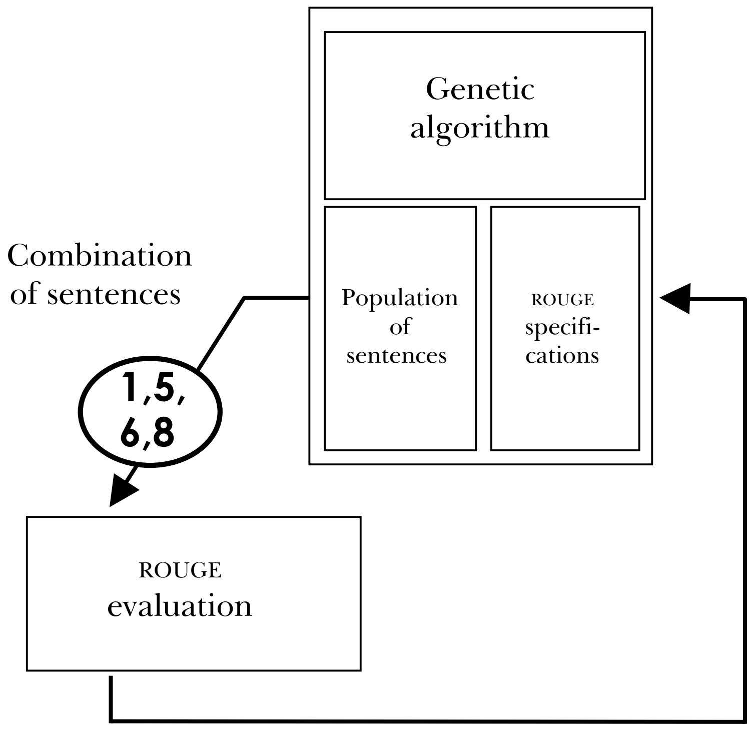
The sentences with greater weight were selected until the desired size of the summary is reached. We consider *best* and *kbest+first* options.

III.4 New method for topline using genetic algorithms

In this section, we present new method to find topline. The best result obtained for the given collection we call topline. Our method can be applied to find topline not only for summarization corpus (for example, such as DUC-2001 till DUC-2007), but also for other tasks of natural language processing. The scheme of the proposed method is showed in Figure II.1. Here we have the detailed algorithm of proposed genetic algorithm. The objective of the proposed genetic algorithm is not only to find topline, but also to find the best combination of the sentences.

The genetic algorithm maintains a population of chromosomes each of which represents a combination of candidate sentences. This genetic algorithm uses data from the system ROUGE to evaluate the fitness of each sentence in the population. It does this evaluation at each time step by simulating with each combination of the sentences and forming a fitness function based on the ROUGE evaluation which characterizes the desired performance. Using this fitness evaluation, the genetic algorithm propagates the number of sentences into the next generation via the combination of genetic operations proposed below. The combination of the sentences that is the fittest one in the population is used to compose a summary.

Figure III.8
Scheme of the proposed genetic algorithm



The proposed procedure of estimating the combination of sentences by GA is summarized as follows (see Figure II.4):

1. Determine the number of sentences of the given text.
2. Construct an initial population.
3. Encode each chromosome in the population.
4. Evaluate the fitness value for each chromosome.
5. Reproduce chromosomes according to the fitness value calculated in Step 4.
6. Create offspring and replace parent chromosomes by the offspring through crossover and mutation.
7. Go to 3 until the maximum number of iterations is met.

Representation

To represent the combination of sentences, chromosomes of length N·B is used, where N is calculated as the numbers of sentences of the original text and B the number of bits which we use to encode the number of sentence.

Population

The initial population is formed randomly. Its size is fixed and equal to 35 individuals.

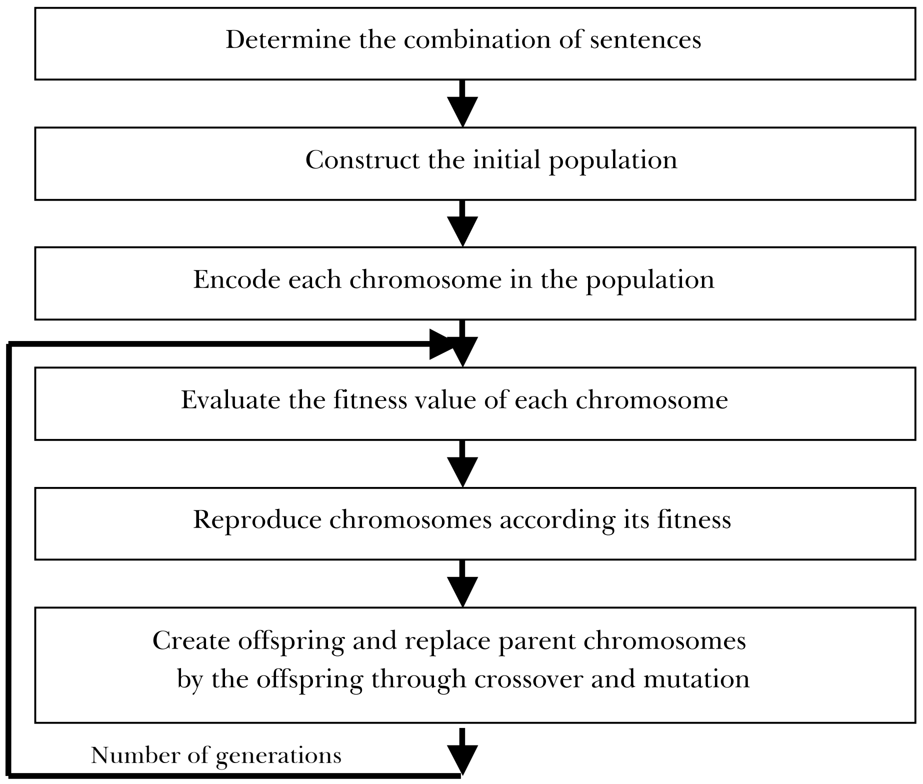
Reproduction

When the evaluation is done, we continue with the reproduction stage. In this step, some genetic operators were evaluated in an attempt to find the appropriate one. We consider in applying the strategy which more approximate to an equilibrium between diversity and convergence. Such strategy or algorithm (for example, CHC algorithm) involves the usage of HUX reproduction operator. So the new population is obtained by applying the HUX operator which ensures that offspring are equidistant between the two parents. This serves as a diversity preserving mechanism.

Fitness function

We propose the fitness function so that it measures for each combination of sentences its *F*-measure score using ROUGE evaluation system. And the combination of sentences, which obtain the best *F*-measure score, will be the best summary of a text.

Figure III.9
Proposed genetic algorithm


Determine the combination of sentences
Construct the initial population
Encode each chromosome in the population
Evaluate the fitness value of each chromosome
Reproduce chromosomes according its fitness
Create offspring and replace parent chromosomes by the offspring through crossover and mutation
Number of generations

III.5 New method using clustering algorithms

In this section, we describe how to apply a clustering algorithm with the objective to find more similar groups of sentences. From each group only one sentence will be chosen assuming that this sentence is the most representative. The resulted summary will be composed by the sentences extracted from each group.

For first experiments, we consider the clustering algorithm *k*-means [Har79], which algorithm is presented as follows:

1. Stage of pre-processing: eliminate stopwords, apply stemming.
2. Stage of clustering of the sentences:
   - 2.1 Elaboration of vector model starting with sequences of the words of a document.
   - 2.2 Determine the number of groups.
   - 2.3 Assign initial seeds for clustering.
   - 2.4 Generation of cluster of sentences.
3. Stage of generation of sentences:
   - 3.1 Search of the sentences more representative of each group.
   - 3.2 Composition of the summary.

Chapter IV

Experimental results for automatic single text summarization

We test new methods for composing document summaries on corpus DUC-2002. This is standard summarization collection in the English language using to compare the results of different text summarization methods. In this chapter, various term selection, term weighting, sentence weighting, and sentence selection schemes are tested. They result from the new methods described in Chapter III (single-document summarization methods). For each experiment the corresponding proposed method was applied and the composed summaries were evaluated. The obtained results are presented and discussed.

IV.1 Experimental settings

We have conducted several experiments to verify our hypotheses formulated in the previous chapters.

IV.1.1 Algorithm

In each experiment, we followed the standard sequence of steps:

- Term selection: decide which features are to be used to describe the sentences;
- Term weighting: decide how the importance of each feature is to be calculated;
- Sentence weighting: decide how the importance of the features is to be combined into the importance measure of the sentence;
- Sentence selection: decide which sentences are selected for the summary.

The specific settings for each step varied between the experiments and are explained below for each experiment.

IV.1.2 Test data set

We used the DUC-2002 collection provided [Duc]. In particular, we used the data set of 567 news articles of different length and with different topics. Each document in the DUC collection is supplied with a set of human-generated summaries provided by two different experts. While each expert was asked to generate summaries of different length, we used only the 100-word variants.

IV.1.3 Evaluation procedure

We used the ROUGE evaluation toolkit [Lin03a]. This system is found highly correlated with human judgments [Lin03b]. It compares the summaries generated by the program with the human-generated (gold standard) summaries. For comparison, it uses different statistics such as *n*-grams co-occurrences, longest common subsequence, weighted subsequence, skip-bigrams co-occurrence, etc. (see description in section I.1.2). Our evaluation was done using *n*-gram (1, 1) setting of ROUGE, which was found to have the highest correlation with human judgments, namely, at a confidence level of 95%.

IV.1.4 Baseline

We denote *Baseline*: *first* the baseline, which selects the first sentences of the text until the desired size of the summary, is reached [Duc]. The baseline configuration selects the first sentences of the text until the desired size is reached [Duc]. This configuration gives very good results on the kind of the texts (news reports) that we experimented with, but would not give so good results on other types of texts. Thus we proposed another baseline (we believe this to be a more realistic baseline for the types of texts other than news reports), denoted *Baseline*: *random*, which selects random sentences; the results presented below are averaged by 10 runs (for results see Table III.4).

IV.2 Experimental methodology

We test new methods using different configurations of term selection, term weighting, sentence weighting, and sentence selection. We propose the following experimental methodology:

- Experiment 1: Different term selection options are tested.
- Experiment 2: Term selection using multiword description extracted for each sentence separately, term weighting and sentence selection.
- Experiment 3: Term selection using multiword description extracted for a collection of sentences (in other words, for a whole document), term weighting and sentence selection.
- Experiment 4: Term selection, term weighting and sentence selection using different thresholds.
- Experiment 5: Term selection, term weighting and sentence selection using the phase of pre-processing.
- Experiment 6: Term selection, term weighting and sentence selection using DimaspCn.
- Experiment 7: Term selection, term weighting and sentence selection using graph algorithm.
- Experiment 8: Term selection, term weighting and sentence selection using genetic algorithm.
- Experiment 9: Term selection, term weighting and sentence selection using clustering algorithm.

IV.3 Experimental results

IV.3.1 Term selection

Experiment 1

For term selection, we compared MFSs with more traditional features such as single words and *n*-grams.

Optionally, stop-words were eliminated at the pre-processing stage; in this case our bigrams (or MFSs) could span more words in the original text, as explained in Chapter II.

For term weighting, the frequency of the term was used; for sentence weighting, the sum of the weights of the terms contained in the sentence was used; for sentence selection, the sentences with greater weight were selected until the desired size of the summary (100 words) is reached.

Discussion

As a kind of statistical significance check, we randomly divided our test data in half and ran this (and most of the other) experiments separately on each subset. These experiments confirmed the qualitative observations reported.

As Table IV.1 shows, MFSs are a promising choice for term selection. This motivated our further experiments with term selection schemes derived from them, as well as with term weighting options for them.

Table IV.1 shows the results. The measures recall or precision can be used for comparison, since the size of all summaries is the same (100 words).

Table IV.1
Recall on 100-words summaries for different term selection options

Term	*With stop-words*	*Without stop-words*
W: words from B or M	0.39421	0.41371
B: repeating bigrams	0.40810	0.42173
M: all MFSs	0.43066	0.44085

IV.3.2 Term selection (extracted for sentence), term weighting, and sentence selection

Experiment 2

Inspired by the above results, we further experimented with MFSs and other term selection options derived from them. In addition to *M*, we consider an option *W* from Chapter II.

The results are shown in Table IV.2. We conducted our experiments in three phases. From Table IV.1 we knew that term selection scheme *M* with stop-words removed gave the best results with other parameters fixed (term weighting, sentence weighting, and sentence selection). So we started from modifying these parameters for the same term selection scheme; see the upper third part of Table IV.2. The first line of the table represents the best result from Table IV.1. The best results are highlighted in boldface.

In each experiment, we consider the following configuration of the main algorithm:

– Pre-processing: Optionally, stop-words were eliminated at the pre-processing stage.
– Term selection: Each original text is represented separately by each sentence. MFSs are extracted from each sentence separately. Resulted multiword descriptions extracted from each sentence are different from multiword descriptions

extracted from a complete document. Specifically, the representation of a text is different, with the consequence that resulted patterns are different.

– Term weighting: frequency of the term in MFSs (*f*); the maximum length of an MFS containing the term (*l*); the same weight for all terms (1).
– Sentence weighting: the sum of the weights of the terms contained in the sentence was used.
– Sentence selection: the sentences with greater weight were selected until the desired size of the summary is reached *(best)*; *k* best sentences were selected, and then the first sentences of the text weight were selected until the desired size of the summary is reached *(kbest+first)*.

Discussion

Then we tried other term selection options, such as *W*, with the term weighting option 1 and the options related to *f*, which showed good performance in the first experiment. The results are shown in the middle third of Table IV.2. Term selection *W* gave better result than *M*. Finally, with the best combinations obtained from the first two experiments, we tried different sentence selection variants; see the last third of Table IV.2.

Table IV.2
Results of the experiment where multiword descriptions are extracted from each sentence

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Term			*Recall*	*Precision*	*F-measure*
M	$l \times f$	best	0.43734	0.45402	0.44519
	1		0.43881	0.45415	0.44600
	l		0.43824	0.45487	0.44606
	f		**0.44034**	**0.45581**	**0.44759**
	$l \times l$		0.42839	0.44633	0.43685
	$l \times\times f$		0.42588	0.44360	0.43423
W	f	best	**0.44483**	**0.45829**	**0.45134**
	1		0.38367	0.40290	0.39291
W	f	1best+first	**<u>0.46523</u>**	**<u>0.48219</u>**	**<u>0.47344</u>**
		2best+first	0.46214	0.47739	0.46952
M	l	1best+first	0.46306	0.48052	0.47150
	f	1best+first	0.46448	0.48185	0.47288
	1	1best+first	0.46423	0.48143	0.47255

One can observe that any *kbest+first* sentence selection option outperformed any combination that used the standard sentence selection scheme, with smaller *k* always giving better results—that is, only the slightest correction to the baseline improved it. The best result was obtained with single words derived from MFSs, with their weighting by the frequency of the corresponding MFS.

IV.3.3 Term selection (extracted for document), term weighting, and sentence selection

Experiment 3

In each experiment, we consider the following configuration of the main algorithm:

- Pre-processing: Optionally, stop-words were eliminated at the pre-processing stage.
- Term selection: Each original text is represented as a collection of sentences. MFSs are extracted from a complete document. In this experiment, in addition to M and W from experiment 2, we considered an option N and generalization of the sets N, $N \setminus W$, $N \setminus M_2$, $N \setminus (W \cup M_2)$.
- Term weighting: frequency of the term in MFSs (f); the maximum length of an MFS containing the term (l); the same weight for all terms (1).
- Sentence weighting: the sum of the weights of the terms contained in the sentence was used.
- Sentence selection: the sentences with greater weight were selected until the desired size of the summary is reached *(best)*; k best sentences were selected, and then the first sentences of the text weight were selected until the desired size of the summary is reached *(kbest+first)*.

Then we tried other term selection options, such as W and N, with the term weighting option 1 and the options related to f, which showed good performance in the first experiment. The results are shown in the middle third of Table IV.3. Term selection W gave a slightly better result than M. The results for N are equal with f and 1 as weighting. Other combinations based on N did not give good results; see Table IV.4 (stop-words excluded, best sentence selection). Finally, we tried different sentence selection variants; see the last third of Table IV.3.

Table IV. 3
Results for different term selection options

Term selection		*Term weighting*	*Sentence selection*	*Results*		
Term	*Stop-words*			*Recall*	*Precision*	*F-measure*
M	excluded	f	best	0.44085	0.45564	0.44796
		1		**0.44128**	**0.45609**	**0.44840**
		l		0.43977	0.45587	0.44752
		l^2		0.42995	0.44766	0.43847
		$l \times f$		0.43812	0.45411	0.44581
	included			0.43353	0.44737	0.44022
W	included	f	best	0.44582	0.45820	0.45181
	excluded			**0.44609**	**0.45953**	**0.45259**
		1		0.38364	0.40277	0.39284
		f^2		0.43892	0.45265	0.44556
N		f or 1		0.43711	0.45099	0.44383
W	excluded	f	1best+first	<u>**0.46576**</u>	<u>**0.48278**</u>	<u>**0.47399**</u>
			2best+first	0.46158	0.47682	0.46895
M		1	1best+first	0.46354	0.48072	0.47185
			2best+first	0.46028	0.47567	0.46772
		l	1best+first	0.46381	0.48124	0.47223
			2best+first	0.45790	0.47430	0.46583

Table IV.4
Results for variants of the set N (options: excluded, best)

Term	*Term weighting*	*Recall*	*Precision*	*F-measure*
N	f or 1	**0.43711**	**0.45099**	**0.44383**
	l	0.42911	0.44324	0.43594
$N \setminus W$	1	0.42009	0.43693	0.42823
	f	0.41849	0.43532	0.42662
$N \setminus M_2$	1	0.42315	0.43806	0.43035
$N \setminus (W \cup M_2)$		0.41084	0.42759	0.41893

Comparison with experiment 1

One can observe that the results for experiment 2 are better than for experiment 3. Therefore, we consider for the comparison the results of experiments 1 and 3:

- State of the art: The author of [Mih04, Mih06] provided us with her data, which were evaluated in the same conditions as proposed methods. Specifically, *DirectedBackward* version of *TextRank* [Mih04] was evaluated. We also list the results of the original *TextRank* with implementation of *PageRank* with *DirectedBackward* version of *TextRank* but with some additional data processing to remove noisy data [Mih06] and the modified *TextRank* with a biased version of *PageRank* [Has07]. See details of the preprocessing in [Mih04, Mih06, Has07].
- Baseline: we use *Baseline*: *first* and *Baseline*: *random* (see Section IV.1).
- Our proposal: We compare these methods with the best results obtained with the best and 1*best+first* sentence selection scheme, as shown in Table IV.3. In both cases our best results were obtained with the options *W* without stopwords for term selection and *f* for term weighting.

For fair comparison, we separated the methods by the type of information they used in addition to the weighting derived from terms:

- None (text is considered as a bag of sentences, sentence as a bag of terms, terms as strings),
- Order of sentences (say, first sentences are treated specially),
- Sophisticated pre-processing to obtain the terms.

We believe that in the future combination of these types of additional information can give even better results. The comparison is given in Table IV.5.

Table IV.5
Comparison of results of experiment 3 with other methods

Additional info used	*Method*	*Recall*	*Precision*	*F-measure*
None	Baseline: *random*	0.37892	0.39816	0.38817
	TextRank: [Mih04]	**0.45220**	0.43487	0.44320
	Proposed: *W*, *f*, *best*	0.44609	**0.45953**	**0.45259**
Order of sentences	Baseline: *first*	0.46407	0.48240	0.47294
	Proposed: *W*, *f*, 1*best+first*	**0.46576**	**0.48278**	**0.47399**
Pre-processing	TextRank: [Mih06]	0.46582	0.48382	0.47450
	TextRank: [Has07]	<u>**0.47207**</u>	<u>**0.48990**</u>	<u>**0.48068**</u>

We could not apply our method with the pre-processing option because we did not have access to the specific details of the pre-processing procedure used in [Mih06] and [Has07] (see experiment 5 for detail of pre-processing). However, in the other two categories our method outperformed the others. Possibly with the same type of pre-processing our method would outperform the others in the last category.

Discussion

We observed that words from repeating bigrams are good terms, and so are MFSs (we can speculate that MFSs are still better semantic units but splitting them into single words gives a more flexible and less sparse comparison). For term weighting, we observed that a good weighting scheme is the number of occurrences of the term in the text as part of a repeating bigram. With these settings, we obtained the results superior to the existing state-of-the-art methods.

Most of the state-of-the-art methods perform worse than the baseline method that takes into account a special ordering of sentences in news reports, which contain a nearly ready abstract in their first sentences. However, our methods can select one sentence better than this baseline method (while already the second-best sentence selected by our method proves to be worse than the baseline). This gives a hybrid method (one sentence our and then back-off to the baseline) superior to both the baseline and other state-of-the-art methods.

In this experiment we did not apply pre-processing that was shown to be beneficial for other methods, so our results are below those of other methods when they do apply it, though above them when they do not. The latter makes us believe that when we apply pre-processing we will obtain results superior to all existing methods. This will be one of the experiments described below.

On the other hand, our experiments show that very different options (some of them rather absurd) only slightly affect the overall result, at least on the collection we used for our experiments. This can probably be explained by the nature of the texts in this collection (short news reports) and maybe by the behaviour of the ROUGE evaluation scheme: the completely random selection baseline is rather high (so nearly any method would give at least similar results) while what seems to be almost top-line—selecting the first sentences of the text—is quite low and quite near to the random baseline. This makes us rather pessimistic about much further progress in the results unless another data collection is used and probably better evaluation schemes are developed.

IV.3.4 Term selection, term weighting, and sentence selection with different thresholds

Experiment 4

For this experiment, we use the configuration of the algorithm of experiment 3. Then we tested that configuration with β = 2, 3, 4 (Table IV.3). We tested the proposed schemes with β = 2 (see Table IV.6), β = 3 (see Table IV.6), and β = 4 (see Table IV.8). The comparison results are shown below in Tables IV.9 - IV.11.

Table IV.6
Results for experiment 4 with β = 2

Term selection		*Term weighting*	*Sentence selection*	*Results*		
Term	*Stop-words*			*Recall*	*Precision*	*F-measure*
M	excluded	$l \times f$	best	0.43731	0.45347	0.44508
		1		**0.43749**	**0.45182**	**0.44438**
		l		0.43731	0.45347	0.44508
		l^2		0.42781	0.44566	0.43640
W	excluded	f	best	**0.44659**	**0.45968**	**0.45293**
		1		0.38367	0.40290	0.39291
		f^2		0.44114	0.45512	0.44790
W	excluded	f	1best+first	**0.46536**	**0.48230**	**0.47355**
			2best+first	0.46296	0.47769	0.47009
M		1	1best+first	0.45674	0.47551	0.46582
		l	1best+first	0.46342	0.48069	0.47177
			2best+first	0.45701	0.47320	0.46484

Table IV.7
Results for experiment 4 with β = 3

Term selection		*Term weighting*	*Sentence selection*	*Results*		
Term	*Stop-words*			*Recall*	*Precision*	*F-measure*
M	excluded	$l \times f$	best	0.43470	0.45120	0.44247
		1		**0.43701**	**0.45310**	**0.44459**
		l		0.43470	0.45120	0.44247
		l^2		0.42686	0.44463	0.43525

Continues...

Term selection		*Term weighting*	*Sentence selection*	*Results*		
Term	*Stop-words*			*Recall*	*Precision*	*F-measure*
W	excluded	*f*	best	**0.44397**	**0.45773**	**0.45062**
		1		0.38367	0.40290	0.39291
		f^2		0.43797	0.45220	0.44485
W	excluded	*f*	1best+first	**0.46622**	**0.48407**	**0.47486**
			2best+first	0.46223	0.47806	0.46989
M		1	1best+first	0.45674	0.47551	0.46582
		l	1best+first	0.46631	0.48392	0.47483
			2best+first	0.46007	0.47638	0.46796

Table IV.8
Results for experiment 5 with $\beta = 4$

Term selection		*Term weighting*	*Sentence selection*	*Results*		
Term	*Stop-words*			*Recall*	*Precision*	*F-measure*
M	excluded	$l \times f$	best	0.43013	0.44680	0.43812
		1		**0.43266**	**0.44861**	**0.44025**
		l		0.43013	0.44680	0.43812
		l^2		0.42354	0.44084	0.43183
W	excluded	*f*	best	**0.44631**	**0.46505**	**0.45536**
		1		0.38367	0.40290	0.39291
		f^2		0.43712	0.45138	0.44402
W	excluded	*f*	1best+first	**0.46788**	**0.48537**	**0.47634**
			2best+first	0.46397	0.47985	0.47165
M		1	1best+first	0.45674	0.47551	0.46582
		l	1best+first	0.46568	0.48373	0.47441
			2best+first	0.45977	0.47604	0.46734

Comparison with experiment 3

In this experiment, we obtained better results with proposed schemes using different thresholds. Here, we compare best results of the actual experiment (see Tables IV.9-IV.11). We detect that the best configuration for MFSs as selected terms was obtained with combination of threshold ($\beta = 2, 3, 4$). Also, we detect that for the terms derived from MFSs, the best threshold is $\beta = 2$. The results of the configuration with combination of sentences with $\beta = 4$ is the best obtained result.

Table IV.9
Comparison of results using different thresholds (terms are MFS)

Method	*Recall*	*Precision*	*F-measure*
M where $\beta = 2, 3, 4$	**0.44128**	**0.45609**	**0.44840**
M where $\beta = 2$	0.43749	0.45182	0.44438
M where $\beta = 3$	0.43701	0.45310	0.44459
M where $\beta = 4$	0.43266	0.44861	0.44025

Table IV.10
Comparison of results using different thresholds (terms derived from MFS)

Method	*Recall*	*Precision*	*F-measure*
M where $\beta = 2, 3, 4$	0.44582	0.45820	0.45181
M where $\beta = 2$	**0.44659**	**0.45968**	**0.45293**
M where $\beta = 3$	0.44397	0.45773	0.45062
M where $\beta = 4$	0.44090	0.45509	0.44776

Table IV.11
Comparison of results using different thresholds (combination of sentences)

Method	*Recall*	*Precision*	*F-measure*
M where $\beta = 2, 3, 4$	0.46576	0.48278	0.47399
M where $\beta = 2$	0.46536	0.48230	0.47355
M where $\beta = 3$	0.46622	0.48407	0.47486
M where $\beta = 4$	**0.46788**	**0.48537**	**0.47634**

Discussion

There are only five better systems [Mih06] than baseline with little differences of the results. In previous experiment, we obtained better results than baseline. For the collection of DUC2002 the results of baseline configuration is very high because the majority of texts consisted of news descriptions and in such type of texts is common that the first sentences describe briefly the given news. In other words, some of the first sentences are abstract or summary of a given file. In other types of texts the configuration of baseline will not work at all. So, it is fair to compare with the state-of-the-art methods like Random Walks [Mih06]. The author of this work provided the data of its summaries which were evaluated in the same conditions as proposed methods. Specifically, *DirectedBackward* version of *TextRank* was evaluated (see Table IV.12, *TextRank*). Finally, the best of the proposed methods is included.

Table IV.12
Coparison of results of experiments 2 and 3 with other methods

Additional info used	*Method*	*Recall*	*Precision*	*F-measure*
None	Baseline: *random*	0.37892	0.39816	0.38817
	TextRank: [Mih04]	**0.45220**	0.43487	0.44320
	Proposed: *Z*, *best*	0.44659	**0.45968**	**0.45293**
Order of sentences	Baseline: *first*	0.46407	0.48240	0.47294
	Proposed: *Z*, 1*best*+*first*	**<u>0.46788</u>**	**<u>0.48537</u>**	**<u>0.47634</u>**

We tested new methods for the automatic generation of text summaries for a single document based on the discovery of MFSs, specifically we tested different combinations of term selection, term weighting, sentence weighting and sentence selection schemes with different thresholds. With first experiment, we observed that MFSs are good terms and help us to obtain good results comparing with words and *n*-grams. In the second experiment, we tested the proposed schemes with different thresholds. We conclude that words derived from MFSs are the best terms with $\beta = 2$ and MFSs are good terms with $\beta = 2, 3, 4$.

IV.3.5 Pre-processing

Experiment 5

The results of experiment are presented in Table IV.13. The best results are highlighted with bold type. We detect that the weighting scheme of frequency of words derived from MFSs gives the best sentence for a summary, and together with sentences obtained with baseline configuration, the best summary is obtained. For the first part of this experiment, we extract MFSs excluding stop-words.

For the second part of this experiment, we change pre-processing configuration: MFS are stemmed and stop-words are excluded from MFSs. See results in Table IV.14.

Table IV.13
Results for configuration of experiment 2 using pre-processing (stop-words excluded)

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Terms			*Recall*	*Precision*	*F-measure*
M	$l \times f$	best	0.42689	0.43347	0.43005
	1		0.44193	0.44426	0.44298
	l		0.42263	0.42961	0.42599
	f		**0.44678**	**0.44849**	**0.44752**
W	f	best	**0.45504**	**0.45626**	**0.45553**
	1		0.39657	0.39834	0.39733
W	f	1best+first	0.46416	0.48090	0.47226
		2best+first	0.46033	0.47532	0.46759
M	1	1best+first	**<u>0.46266</u>**	**<u>0.47979</u>**	**<u>0.47094</u>**
	f	1best+first	0.44605	0.44771	0.44676

Table IV.14
Results for configuration of experiment 2 using pre-processing (stemming and stop-words excluded)

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Terms			*Recall*	*Precision*	*F-measure*
M	$l \times f$	best	0.42538	0.43151	0.42831
	1		0.44315	0.44517	0.44405
	l		0.41837	0.42496	0.42153
	f		**0.44538**	**0.44681**	**0.44598**
W	f	best	**0.45576**	**0.45679**	**0.45615**
	1		0.39657	0.39834	0.39733
W	f	1best+first	0.46413	0.48081	0.47220
		2best+first	0.46259	0.47721	0.46966
M	1	1best+first	**<u>0.46456</u>**	**<u>0.48169</u>**	**<u>0.47285</u>**
	f	1best+first	0.46432	0.48139	0.47258

For the third part of this experiment, MFS are stemmed and stop-words are included. The results are shown in Table IV.15.

Table IV.15
Results for configuration of experiment 2 using pre-processing (stemming but stop-words kept)

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Terms			*Recall*	*Precision*	*F-measure*
M	$l \times f$	best	0.43386	0.43673	0.43494
	1		0.43971	0.44234	0.44067
	l		0.43380	0.43664	0.43487
	f		**0.43867**	**0.44100**	**0.43949**
W	f	best	**0.44609**	**0.44632**	**0.44608**
	1		0.39657	0.39834	0.39733
W	f	1best+first	0.46486	0.48189	0.47310
		2best+first	0.46293	0.47831	0.47037
M	1	1best+first	0.46461	0.48182	0.47293
	f	1best+first	**<u>0.46508</u>**	**<u>0.48233</u>**	**<u>0.47343</u>**

Comparison with experiment 3

We compare with the state-of-the-art methods like TextRank [Mih06]. Specifically, DirectedBackward version of TextRank was evaluated in the same conditions as proposed methods (see Table IV.16, TextRank) and the same version of TextRank with pre-processing (see Table IV.16, TextRank *w*/pre-processing). And also we compare results presented in experiment 3 (see Table IV.16, MFS *w/o* pre-processing). Finally, the best version of each experiment is included (see MFS *w*/ pre-processing 1, 2, and 3).

We can see that pre-processing does not affect positively obtaining terms for extractive summarization, at least not in the case of MFSs.

Discussion

We modified our automatic single-document text summarization method based on MFSs as terms by including pre-processing stage. We found, however, that pre-processing does not affect positively the summaries obtained with our method. This is good news and bad news. Bad because we did not find better terms, and our summaries did not improve. Good because we confirmed that classic plain MFSs (sequences of wordforms and not stems or only significant words), which are calculated in a totally language-independent manner, are good terms for this task.

Table IV.16
Comparison of results of pre-processing with other methods

Method	*Recall*	*Precision*	*F-measure*
TextRank	0.45220	0.43487	0.44320
TextRank *w*/pre-processing	0.46582	0.48382	0.47450
MFS *w/o* pre-processing	**0.46576**	**0.48278**	**0.47399**
MFS *w*/pre-processing 1	0.46266	0.47979	0.47094
MFS *w*/pre-processing 2	0.46456	0.48169	0.47285
MFS *w*/pre-processing 3	0.46508	0.48233	0.47343

On the other hand, since we showed that our pre-processing almost does not either affect the results negatively, one can exclude stop-words and word endings from processing and still obtain almost the same quality of extractive summarization. Excluding stop-words significantly reduces the risk of exponential explosion of the size of the data structures used to mine for MFSs and for their application in our method, as well as the number of the terms (MFSs or *n*-grams) dealt with.

IV.3.6 Graph algorithm

Experiment 6

The main contribution of this method in term selection step is the proposal of using MFS as nodes of a graph, and in sentence weighting—using PageRank. You can see more detail about graph ranking algorithm in Section II.3. In each experiment, we consider the following configuration of the graph algorithm:

Vertices. We propose to use MFSs as vertices of a graph.

Edges. Relations that connect MFSs are term weighting relations such as frequency of MFSs in a text, length of MFS, and its presence.

Algorithm. We use a graph-based ranking algorithm PageRank (a text version is called *TextRank*) to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

In each experiment, we consider the following configuration of the proposed method:

- Term selection: *M, W.*
- Term weighting: frequency of the term in MFSs (*f*); the maximum length of an MFS containing the term (*l*); the same weight for all terms (1).
- Sentence weighting: using PageRank.
- Sentence selection: the sentences with greater weight were selected until the desired size of the summary is reached.

For this task, the goal is to rank MFSs, and therefore a vertex is added to the graph for each MFS in the text. To draw edges between vertices, we are defining a term weighting relation, where "term weighting" can be defined in various ways. In the experiments presented below, we use a term weighting described in section III. Such a relation between two sentences can be seen as a process of recommendation: a sentence that addresses certain concepts in a text, gives the reader a recommendation to refer to other sentences in the text that address the same or similar concepts. The resulting graph is highly connected, with a weight associated with each edge, and thus we use again the weighted version of the graph algorithms.

The results are shown in Tables IV.17 and IV.18. The size of summaries is 100 words. F-measure is used for comparison. The best results are highlighted in bold-face.

In Table IV.17 normalization is used. More specifically, when we calculate the weight of a sentence, it is divided between the number of words of a given sentence.

We conducted our experiments in three phases. From the results of other methods, we knew that term selection scheme *M* with stop-words removed gave the best results with other parameters fixed (term weighting, sentence weighting, and sentence selection). So we started from modifying these parameters for the same term selection scheme; see the upper part of Table IV.17.

Then we tried other term selection options, such as *W*, with the term weighting option 1 and the options related to *f*, which showed the best performance in the first experiment. The results are shown in the middle third of Table IV.17. Term selection *W* gave a much better result than *M*. We discarded to report term selection for *N* because the obtained results were not better. Other combinations based on *M* and *W* did not give good results compared with other method where this option gave the best results, see the below part of Table IV.17.

Finally, we discarded the normalization for sentence weighting and we could obtain better results for *M* and *W* term selection where *M* is a slightly better than *W* (see Table IV.18). One can observe that any *kbest+first* sentence selection option did not outperform the standard sentence selection scheme. The best result

Table IV.17
Results of the graph algorithm (normalization is used)

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Terms			*Recall*	*Precision*	*F-measure*
M	f	best	0.48009	0.47757	0.47865
	f^2		0.48056	0.47801	0.47910
	1		0.46668	0.48337	0.47474
	l		0.48025	0.47773	0.47881
	l^2		**0.48058**	**0.47812**	**0.47917**
	$f \times l$		0.48060	0.47810	0.47916
	$f \times\times l$		0.48079	0.47831	0.47937
W	f	best	**<u>0.48659</u>**	**<u>0.48324</u>**	**<u>0.48473</u>**
	1		0.47682	0.47604	0.47626
	f^2		0.48705	0.48235	0.48451
W	f	1best+first	0.47603	0.47518	0.47543
		2best+first	0.47718	0.47621	0.47652
M	l	1best+first	0.47783	0.47699	0.47724
		2best+first	0.48212	0.48088	0.48132
	f	1best+first	0.47797	0.47712	0.47737
		2best+first	**0.48211**	**0.48093**	**0.48134**

Table IV.18
Results of the graph algorithm

Term selection	*Term weighting*	*Sentence selection*	*Results*		
Terms			*Recall*	*Precision*	*F-measure*
M	f	best	0.48803	0.48533	0.48626
	f^2		0.48746	0.48482	0.48572
	1		0.47484	0.49180	0.48283
	l		**<u>0.48823</u>**	**<u>0.48577</u>**	**<u>0.48658</u>**
	l^2		0.48741	0.48518	0.48587
	$f \times l$		0.48796	0.48529	0.48620
	$f \times\times l$		0.48716	0.48497	0.48564
W	f	best	**0.48821**	**0.48424**	**0.48604**
	1		0.47529	0.47483	0.47489
	f^2		0.48784	0.48322	0.48534
W	f	1best+first	0.47694	0.47612	0.47635
		2best+first	0.47870	0.47761	0.47798
M	l	1best+first	0.47711	0.47623	0.47650
		2best+first	0.48064	0.47923	0.47976
	f	1best+first	0.47738	0.47649	0.47676
		2best+first	**0.48148**	**0.48016**	**0.48065**

was obtained with MFSs, with their weighting by the length of the corresponding MFS.

IV.3.7 Topline using genetic algorithm

Experiment 7

First, we order all documents based on the number of sentences. First documents in the list are short document with small number of sentences, and in the end of list, the documents are long with a lot of sentences. We calculate topline trying all combinations of sentences in one document. The best combination of sentences, which has the highest score of F-measure, is chosen as topline result for a given document.

The average result of topline is 0.62971 (for 350 documents), see Table IV.19. It is not possible to find the best combination of sentences for all documents because of dimensionality explosion for long documents. That is why we propose to use genetic algorithm to find the result of topline. The average result of topline using GA is 0.5931 (for 568 documents), see Table IV.20. The final topline result is 0.5960 (see Table IV.21).

Table IV.19
Topline results trying all combination of sentences

Number of sentences	*F-measure*
1-49	0.67297
50-99	0.65268
100-149	0.63767
150-199	0.62785
200-249	0.61697
250-299	0.59601
between 300-400	0.60715
total	**0.62971**

Table IV.20
Topline results ussing proposed GA

Number of sentences	*F-measure*
1-49	0.6720
50-99	0.6514
100-149	0.6346
150-199	0.6218
200-249	0.6095
250-299	0.5821
300-349	0.5824
350-399	0.5841
400-449	0.5578
450-499	0.5553
500-549	0.5408
550-568	0.5250
total	**0.5931**

Table IV.21
Final topline results considering all combination of sentences (0-299) and proposed GA (300-368)

Number of sentences	*F-measure*
1-49	0.67297
50-99	0.65268
100-149	0.63767
150-199	0.62785
200-249	0.61697
250-299	0.59601
300-349	0.5824
350-399	0.5841
400-449	0.5578
450-499	0.5553
500-549	0.5408
550-568	0.5250
total	**0.5960**

IV.3.8 Clustering algorithm

Experiment 8

In each experiment (see Table IV.22-IV.25), we consider the following configuration of the proposed algorithm:

– Pre-processing: eliminate stop-words, then apply Porter stemming [Por80];
– Term selection: decide which size of *n*-grams as features are to be used to describe the sentences;
– Term weighting: decide how the importance of each feature is to be calculated, it can be BOOL, TF, IDF or TFIDF;
– Sentence clustering: decide the initial seeds for the *k*-means algorithm, in this case Baseline sentences;
– Sentence selection: after *k*-means finishes, select the closest sentence (the most representative) to each centroid for composing the summary.

Table IV.22
Recall for different sizes of *n*-grams and its weights

Term selection	*Term weighting*			
	BOOL	*TF*	*IDF*	*TFIDF*
1-grams	0.47517	0.47686	0.47632	0.47545
2-grams	0.47705	0.47694	0.47779	0.47777
3-grams	0.47940	0.47940	0.47932	0.47932
4-grams	0.47891	0.47891	0.47916	0.47913
5-grams	0.47942	0.47942	0.47910	0.47910
6-grams	0.47989	0.47979	0.48020	0.48020
7-grams	0.47976	0.47992	0.47964	0.47993
8-grams	**<u>0.48113</u>**	0.48072	**0.48075**	0.48055
9-grams	**0.48084**	**0.48084**	0.48020	**<u>0.48109</u>**
10-grams	0.48058	**<u>0.48103</u>**	**<u>0.48155</u>**	**0.48101**
11-grams	0.48004	0.47903	0.47856	0.47856

Table IV.23
Precision for different sizes of *n*-grams and its weights

Term selection	*Term weighting*			
	BOOL	*TF*	*IDF*	*TFIDF*
1-grams	0.47039	0.47219	0.47168	0.47078
2-grams	0.47211	0.47204	0.47284	0.47284
3-grams	0.47452	0.47454	0.47441	0.47441
4-grams	0.47410	0.47410	0.47432	0.47429
5-grams	0.47462	0.47462	0.47432	0.47432
6-grams	0.47495	0.47497	0.47530	0.47530
7-grams	0.47493	0.47510	0.47487	0.47512
8-grams	0.47633	0.47606	**0.47588**	0.47587
9-grams	0.47632	<u>**0.47632**</u>	0.47553	<u>**0.47654**</u>
10-grams	0.47575	**0.47609**	<u>**0.47684**</u>	**0.47634**
11-grams	0.47529	0.47409	0.47370	0.47370

Table IV.24
F-measure for different sizes of *n*-grams and its weights

Term selection	*Term weighting*			
	BOOL	*TF*	*IDF*	*TFIDF*
1-grams	0.47264	0.47439	0.47387	0.47298
2-grams	0.47445	0.47436	0.47519	0.47517
3-grams	0.47683	0.47684	0.47673	0.47673
4-grams	0.47638	0.47638	0.47661	0.47658
5-grams	0.47689	0.47689	0.47658	0.47658
6-grams	0.47729	0.47725	0.47762	0.47762
7-grams	0.47721	0.47738	0.47713	0.47739
8-grams	<u>**0.47860**</u>	0.47826	**0.47818**	0.47808
9-grams	**0.47845**	<u>**0.47845**</u>	0.47773	<u>**0.47868**</u>
10-grams	0.47803	**0.47842**	<u>**0.47906**</u>	**0.47854**
11-grams	0.47753	0.47642	0.47599	0.47599

Table IV.25
F-measure for different configurations of the proposed clustering algorithm

Model		*Weighting*			
		BOOL	*TF*	*IDF*	*TFIDF*
Initial baseline centroids for *k*-means	Bag of words	0.47264	0.47439	0.47387	0.47298
	n-grams with $n = 2$ (bigrams)	0.47445	0.47436	0.47519	0.47517
	n-grams with $n = 3$ (trigrams)	**0.47683**	**<u>0.47684</u>**	0.47673	0.47673
	n-grams with $n = 4$ (tetragrams)	0.47638	0.47638	0.47661	0.47658
	n-grams with $n = 5$ (pentagrams)	**<u>0.47689</u>**	**<u>0.47689</u>**	0.47658	0.47658
	MFSs $\beta = 2$, GAP $= 0$	0.47022	0.46862	0.47050	0.46985
Initial random centroids for *k*-means	Bag of words	0.44374	0.44037	0.43949	0.43996
	n-grams with $n = 2$ (bigrams)	0.43814	0.43824	0.43927	0.43953
	n-grams with $n = 3$ (trigrams)	0.44054	0.43519	**<u>0.44644</u>**	0.44127
	n-grams with $n = 4$ (tetragrams)	0.43511	0.43510	0.44027	0.43142
	n-grams with $n = 5$ (pentagrams)	0.43617	0.43892	**0.44343**	0.43604
	MFSs $\beta = 2$, GAP $= 0$	**<u>0.44522</u>**	0.43979	0.43953	0.44056

Discussion

In this experiment, we tested an extractive automatic text summarization approach by sentence extraction using an unsupervised learning algorithm. In particular, the *k*-means algorithm for creating groups of similar sentences was used. Then, from the groups of sentences, the most representative sentence was selected for composing the summary. Normally, the definition of the number of groups to form and the initial seeds of the groups are considered as disadvantages of *k*-means. However, these parameters are used to take advantage of Baseline sentences in order to improve the quality of the summaries. The proposed method, in contrast to supervised methods, does not need large amount of golden samples for training. Therefore, our proposed method is language-and domain-independent.

According to experimental results we demonstrate that the proposed method obtains more favourable results than others state-of-the-art methods; ranking our proposed method in second place, very close to the first place. In addition, our proposed method outperforms the Baseline (first) heuristic for F-measure results, except for 1-gram and BOOL weighting.

Conclusions

This section concludes this book. We include the comparison of the experimental results described in this book with the state-of-the-art methods. Finally, the contributions of this book are included.

Conclusions

In this book, new methods were presented for automatic single text extractive summarization. The development of these methods for automatic generation of text summarization allows us to contribute in an efficient way to the area of natural language processing.

Each of the proposed methods includes the description of term selection, term weighting, sentence weighting, and sentence selection steps. For each experiment, the configuration of the proposed methods was described and the corresponding results were given. Also, the discussion of the experimental results and comparison between different experiments presented in this book and the state-of-the-art were explicitly specified.

The following new methods for automatic single text extractive summarization were described in this book:

- New methods for automatic generation of text summarizes based on the discovery of maximal frequent sequences.
- New methods for automatic generation of text summaries which are superior to the state-of-the-art methods.
- New methods to deal with the task of generation of summaries in a language-independent way.

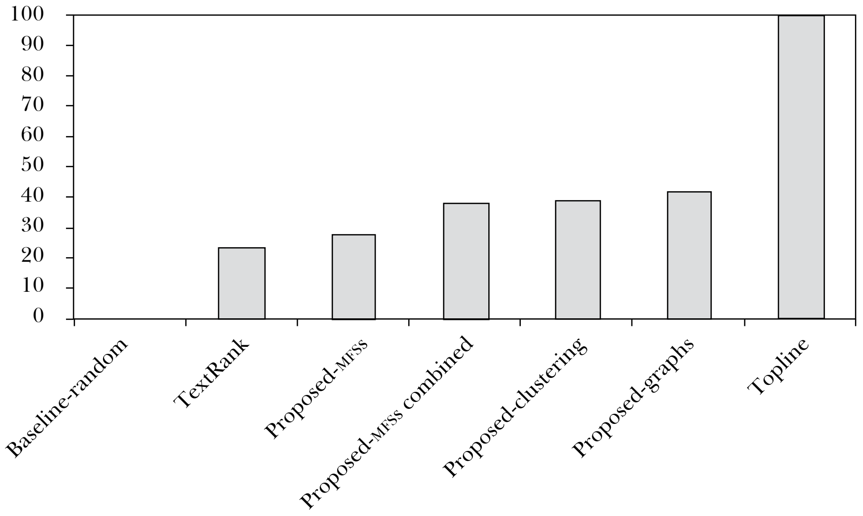
- New methods to deal with the task of generation of summaries in a domain-independent way.
- New methods to deal with the task of generation of summaries using graph, clustering, and genetic algorithm.

The proposed method was tested on corpus DUC-2002. This is standard summarization collection in the English language proposed in text summarization conference which facilitates the comparison of the results obtained by researchers of the area of text summarization.

The necessity of having better methods motivated this work; especially we look for the most important parts of the text that can be automatically extracted. In particularly, we use maximal frequent sequences in order to extract these parts of text.

The comparison results are shown in Figure 1. Topline was calculated in order to find the result of evaluation for the best summary. In the same manner, the result of evaluation for the worst summary can be obtained using the proposed method. However, random selection of sentences for composing a summary is considered as the worst summary. Also, considering that F-measure do not adequately qualifies summaries; we can recalculate the results as shown in Figure 1.

Figure 1
Comparison results for different proposed methods for single-document summarization



List of new methods described in this book

In this section, we present the list of new methods described in this book, as follows:

- Development of new methods for automatic single extractive text summarization.
- New methods for generating text summarizes based on the discovery of maximal frequent sequences.
- New methods for automatic generation of text summaries which are superior to the state-of-the-art methods.
- New methods to deal with the task of automatic generation of summaries in a language-independent way.
- New methods to deal with the task of automatic generation of summaries in a domain-independent way.
- New methods for automatic text summarization using graph, clustering, and genetic algorithms.

Also some additional contributions are:

- Identification of general steps for an automatic extractive text summarization method.
- New representation for displaying the results of single extractive text summarization.

References

[Agr95] Agrawal R., Srikant R. "Mining Sequential Patterns", In Proc. of the International Conference on Data Engineering, 1995.

[Ace07] Aceves-Peréz R., Montes y Gómez M., Villaseñor Pineda L. Enhancing Cross-Language Question Answering by Combining Multiple Question Translations. Lecture Notes in Computer Science, Springer-Verlag, Vol. 4394, pp. 485-493, 2007.

[Bae99] Baeza-Yates R. Modern Information Retrieval. Addison Wesley Longman Publishing Co. Inc., 1999.

[Bar99] Barzilay R., Elhadad M. Using Lexical Chains for Text Summarization. In: Inderjeet Mani, Mark T. Maybury (Eds.), Advances in Automatic Text Summarization, Cambridge/MA, London/England: MIT Press, pp. 111-121, 1999.

[Bar03] Barzilay R. Information Fusion for Multi Document Summarization. Ph.D. Thesis. Columbia University, 2003.

[Bar05] Barzilay R., McKeown K. Sentence Fusion for Multi Document News Summarization. Computational Linguistics, Vol. 31, Issue 3, pp. 297-328, ISSN: 0891-2017, 2005.

[Bol01] Bolshakov I., Gelbukh. A. A Very Large Database of Collocations and Semantic Links. Proc. NLDB'2000: 5th International Conference on Applications of Natural Language to Information Systems, France, In: Mokrane Bouzeghoub et al. (Eds.) Natural Language Processing and Information Systems. Lecture Notes in Computer Science, ISSN 0302-9743, Springer-Verlag, Vol. 1959, pp. 103-114, 2001.

[Bol04a] Bolshakov I., Gelbukh A. Computational Linguistics: Models, Resources, Applications. IPN/UNAM/FCE, ISBN 970-36-0147-2, 2004.

[Bol04b] Bolshakov I. Getting One's First Million... Collocations. Lecture Notes in Computer Science, Springer-Verlag, ISSN 0302-9743, Vol. 2945, pp. 226-239, 2004.

[Bol05] Bolshakov I., Galicia-Haro S., Gelbukh A. Detection and Correction of Malapropisms in Spanish by means of Internet Search. 8th International Conference Text, Speech and Dialogue (TSD-2005), Czech Rep. Lecture Notes in Artificial Intelligence (indexed in SCIE), ISSN 0302-9743, ISBN 3-540-28789-2, Springer-Verlag, Vol. 3658, pp. 115-122, 2005.

[Bol08] Bolshakov I. Various Criteria of Collocation Cohesion in Internet: Comparison of Resolving Power. Computational Linguistics and Intelligent Text Processing (CICLing-2008, Israel). Lecture Notes in Computer Science, Springer-Verlag, Vol. 4919, pp. 64-72, 2008.

[Bri98] Brin S., Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, Vol. 30, pp. 1-7, 1998.

[Bru01] Brunn M., Chali Y., Pinchak C. Text Summarization Using Lexical Chains. Proc. of Document Understanding Conference 2001, <http://duc.nist.gov/pubs.html#2001>.

[Car98] Carbonell J., Goldstein J. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. SIGIR'98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, pp. 335-336, 1998.

[Car03] Carlson L., Marcu D., Okurowski M. E. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In: Jan van Kuppevelt and Ronnie Smith (Eds.), Current Directions in Discourse and Dialogue. Kluwer Academic Publishers, 2003.

[Car04] Carberry S., Elzer S., Green N., McCoy K., Chester D. Extending Document Summarization to Information Graphics. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 3-9.

[Car06] Carberry S., Elzer S., Demir S. Information Graphics: An Untapped Recourse for Digital Libraries. SIGIR'06, ACM 1-59593-369-7/06/0008, 2006.

[Chu04] Chuang T. W., Yang J. Text Summarization by Sentence Segment Extraction Using Machine Learning Algorithms. Proc. of the ACL-04 Workshop, Spain, 2004.

[Cha99] Chambers Lance D. Practical Handbook of Genetic Algorithm Complex Coding System, CRC Press, 1999.

[Cha04] Chali Y., Kolla M. Summarization techniques at DUC 2004. Proceedings of the 2004 Document Understanding Workshop Boston, USA, May 6-7, 2004.

[CICLing] CICLing. Conference on Intelligent Text Processing and Computational Linguistics (2000-2009), <www.CICLing.org>.

[Cri05] Cristea D., et al. Summarization through Discourse Structure. CICLing 2005, LNCS, Vol. 3406, Springer-Verlag, pp. 621-632, 2005.

[Cor04] Corston-Oliver S., Ringger E., Gamon M., Campbell R. Task-Focused Summarization of Emails. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 43-50.

[Dan03] Daniel N., Radev D., and Allison T. Sub-Event Based Multi-Document Summarization. Proc. of the HLT-NAACL Workshop on Text Summarization, pp. 9-16, 2003.

[Dau04] Daume H., Marcu D. Generic Sentence Fusion and Ill-defined Summarization Task. Proc. of ACL Workshop on Summarization, pp.96-103, 2004.

[Dav07] D'Avanzo E., Elia A., Kuflik T., Vietri S. LAKE System at DUC-2007. Proc. of Document Understanding Conference 2007/, <http://duc.nist.gov/pubs.html#2007>.

[Den06] Denicia-Carral C., Montes-y-Gómez M., Villaseñor-Pineda L., García-Hernández. R. Text Mining Approach for Definition Question Answering, 5th International Conference on Natural Language Processing (FinTal), LNCS, Springer-Verlag 2006.

[Dia06] Dia Q., Shan J. A New Web Page Summarization Method. SIGIR'06, ACM 1-59593-369-7/06/0008, 2006.

[Dor04] Doran W. et al. Assessing the Impact of Lexical Chain Scoring Methods and Sentence Extraction Schemes on Summarization. CICLing, LNCS, Vol. 2945, Springer-Verlag, pp. 622-630, 2004.

[Duc] DUC. Document Understanding Conference, <www-nlpir.nist.gov/projects/duc>.

[Edm69] Edmundson H. P. New Methods in Automatic Extraction. In Journal of the ACM 16(2), pp. 264-285, 1969.

[Esh91] Eshelman Larry J. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, In: G. Rawlins (Ed.), Foundations of Genetic Algorithms, pp. 265-283, Morgan Kaufmann, 1991.

[Eva05] Evans D., McKeown K. Identifying Similarities and Differences Across Arabic and English News. Proc. of the International Conference on Intelligence Analysis. McLean, VA, 2005.

[Far04] Farzindar A., Lapalme G. Legal Text Summarization by Exploration of the Thematic Structure and Argumentative Roles. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 27-33.

[Fil04] Filatova E., Hatzivassiloglou V. Event-Based Extractive Summarization. Proc. of ACL Workshop on Summarization, pp.104-111, 2004.

[Fil07] Filippova K., Mieskes M., Nastase V., et al. Cascaded Filtering for Topic-Driven Multi-Document Summarization. Proc. of Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Fer07] Ferrández S., Ferrández A. The Negative Effect of Machine Translation on Cross-Lingual Question Answering. Lecture Notes in Computer Science, Springer-Verlag, Vol. 4394, pp. 494-505, 2007.

[Fut04] Futrelle R. Handling Figures in Document Summarization. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 61-65.

[Gal07] Galicia Haro S. N., Gelbukh, A. IPN, ISBN 970-36-0265-7, 2007.

[Gar04] García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text, 9th Iberoamerican Congress on Pattern Recognition (CIARP), LNCS, Vol. 3287, pp. 478-486, Springer-Verlag, 2004.

[Gar06] García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection, CICLing, LNCS, Vol. 2945, pp. 514-523, Springer-Verlag, 2006.

[Gel03a] Gelbukh A., Sidorov G., Sang Yong Han. Evolutionary Approach to Natural Language Word Sense Disambiguation through Global Coherence Optimization. WSEAS Transactions on Communications, ISSN 1109-2742, Issue 1, Vol. 2, pp. 11-19, 2003.

[Gel03b] Gelbukh A., Bolshakov I. Internet, a True Friend of Translator. International Journal of Translation, ISSN 0970-9819, Vol. 15, No. 2, pp. 31-50, 2003.

[Gel03c] Gelbukh A., Sidorov G., Automatic Selection of Defining Vocabulary in an Explanatory Dictionary. Proc. CICLing-2002, 3rd International Conference on Intelligent Text Processing and Computational Linguistics, Mexico. Lecture Notes in Computer Science, ISBN 3-540-43219-1, Vol. 2276, Springer-Verlag, pp. 300-303, 2002.

[Gel04a] Gelbukh A., Sidorov G., Sang Yong Han, Hernandez-Rubio E. Automatic Enrichment of Very Large Dictionary of Word Combinations on the Basis of Dependency Formalism. In: Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, Humberto Sossa (Eds.), MICAI-2004, Mexican International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, Springer-Verlag, ISSN 0302-9743, Vol. 2972, pp. 430-437, 2004.

[Gel04b] Gelbukh A., Bolshakov I. On Correction of Semantic Errors in Natural Language Texts with a Dictionary of Literal Paronyms. In: Jesus Favela, Ernestina Menasalvas, Edgar Chávez (Eds.), Advances in Web Intelligence (AWIC-2004, 2nd International Atlantic Web Intelligence Conference, Mexico). Lecture Notes in Artificial Intelligence, Springer-Verlag, ISSN 0302-9743, Vol. 3034, pp. 105-114, 2004.

[Gel06] Gelbukh A., Sidorov, G. Procesamiento automático del español con enfoque en recursos léxicos grandes. IPN, ISBN 970-36-0264-9, 2006.

[Gel08] CV and web-page of Prof. Ph.D. Alexander Gelbukh, <www. Gelbukh. com>

[Gol89] Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[Hac05] HaCohen-Kerner Yaakov, et al. Automatic Extraction and Learning of Keyphrases from Scientific Articles. CICLing, LNCS, Vol. 3406, Springer-Verlag, pp. 645-657, 2005.

[Hac04] Hachey B., Grover C. A Rhetorical Status Classifier for Legal Text Summarization. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 35-42.

[Har79] Hartigan J. A., Wong M. A. A *k*-Means Clustering Algorithm. Applied Statistics, Vol. 28(1), pp. 100-108, 1979.

[Hau04] Haupt Randy L., Haupt Sue Ellen. Practical Genetic Algorithms 2nd Edition (ebook), Wiley, 2004.

[Has07] Hassan, S., Mihalcea, R., Banea, C. Random-Walk Term Weighting for Improved Text Classification. Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007), Irvine, CA, 2007.

[Her06a] Hernández-Reyes E., García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. Document Clustering Based on Maximal Frequent Sequences, 5th International Conference on Natural Language Processing (FinTal), LNCS, Springer-Verlag, 2006.

[Her06b] Hernández-Reyes E., García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. Document Representation Based on Maximal Frequent Sequence Sets. 11th Iberoamerican Congress on Pattern Recognition (CIARP), LNCS, Springer-Verlag, 2006.

[Hil02] Hardy Hilda: Cross-Document Summarization by Concept Classification. Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, pp. 121-128, 2002.

[Hov03] Hovy E. The Oxford Handbook of Computational Linguistics, Chapter about Text Summarization, In Mitkov R. (Ed.), NY, 2003.

[Ina08] El laboratorio de Tecnologías del Lenguaje de la Coordinación de Ciencias Computacionales del Instituto Nacional de Astrofísica, Óptica y Electrónica, headed by Ph.D. Manuel Montes y Gómez and Ph.D. Luis Villaseñor Pineda, <ccc.inaoep.mx/labtl>.

[Jai99] Jain A. K., Murty M. N., Flynn P. J. Data Clustering: A Review. ACM Computing Surveys, Vol. 31(3), pp. 264-323, 1999.

[Kle99] Kleinberg J. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, Vol. 46, Num. 5, pp. 604-632, 1999.

[Kol05] Kolla M., Chali Y. Experiments in DUC 2005. Proceedings of the 2005 Document Understanding Workshop, Vancouver, Canada, October 9-10, 2005.

[Kos04] Koster, C. H. A. Transducing Text to Multiword Units, Workshop on Multiword Units MEMURA at the 4th International Conference on Language Resources and Evaluation (LREC-2004), Lisbon, Portugal, 2004.

[Kup95] Kupiec J., Pedersen J.O., and Chen F. A Trainable Document Summarizer. Proc. of the 18th ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 68-73, Seattle, 1995.

[Ledo03] Ledo Mezquita Y., Sidorov G., Gelbukh A. Tool for Computer-Aided Spanish Word Sense Disambiguation. Computational Linguistics and Intelligent Text Processing (CICLing-2003, Mexico). Lecture Notes in Computer Science, ISSN 0302-9743, ISBN 3-540-00532-3, Springer-Verlag, Vol. 2588, pp. 277-280, 2003.

[Lin97] Lin C. Y., Hovy E. Identify Topics by Position. Proc. of the 5th Conference on Applied NLP, 1997.

[Lin03a] Lin C. Y. ROUGE: A Package for Automatic Evaluation of Summaries. Proc. of the Human Technology Conference (HLT-NAACL), Canada, 2003.

[Lin03b] Lin C. Y., Hovy E. Automatic Evaluation of Summaries Using N-Gram Co-Occurrence Statistics. Proc. of the Human Technology Conference, Canada, 2003.

[Lin04a] Lin C. Y. Looking for a Few Good Metrics: Automatic Summarization Evaluation - How Many Samples Are Enough? Proc. of NTCIR Workshop, Japan, 2004.

[Lin04b] Lin C. Y., et al. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In: Proc. of the 20th International Conference on Computational Linguistics (COLING), Switzerland.

[Lin04c] Lin C. Y., et al. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. Proc. of the 42nd Annual Meeting of the ACL, Spain, 2004.

[Liu06a] Liu D., He Yanxiang et al. Multi-Document Summarization Based on BE-Vector Clustering. CICLing 2006, LNCS, Vol. 3878, Springer-Verlag, pp. 470-479, 2006.

[Liu06b] Liu D., Wang Y. et al. Multiple Documents Summarization Based on Genetic Algorithm. FSKD, LNAI 4223, pp. 355-364, Springer-Verlag, 2006.

[Liu06c] Liu D. et al. Genetic Algorithm Based Multi-Document Summarization. PRICAI, LNAI 4099, Springer-Verlag, pp. 1140-1144, 2006.

[Li07] Li J., Sun L. A Query-Focused Multi-Document Summarizer Based on Lexical Chains. Proc. of Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Luh57] Luhn H. P. A statistical Approach to Mechanical Encoding and Searching of Literary Information. IBM Journal of Research and Development, pp. 309-317, 1957.

[Luh59] Luhn H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, pp. 159-165, 1959.

[Luh75] Luhn H. P. A Statical Approach to Mechanical Encoding and Searching of Literary Information. IBM Journal of Research and Development, pp. 309-317, 1975.

[Mad07] Madnani N., Zajic D., Dorr B., et al. Multiple Alternative Sentence Compressions for Automatic Text Summarization. Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Man99] Manning C. Foundations of Statistical Natural Language Processing, MIT Press, London, 1999.

[Man07] Manning C. An Introduction to Information Retrieval. Cambridge University Press, 2007.

[Mar00a] Marcu D. The Theory and Practice of Discourse and Summarization. The MIT Press, chapter 9, 2000.

[Mar00b] Marcu D. The Rhetorical Parsing of Unrestricted Texts: A Surface-Based Approach. Computational Linguistic, Vol. 26 (3), 2000.

[Mar01] Marcu D. Discourse-Based Summarization in DUC-2001. Document Understanding Conference 2001, <http://duc.nist.gov/pubs.html#2001>.

[Mck03] McKeown K., Barzilay R., Chen J., Elson D., Klavans J., Nenkova A., Schiffman B., Sigelman S. Columbia's Newsblaster: New Features and Future Directions. Proc. of the Human Language Technology Conference, Vol. II, 2003.

[Mel99] Melanie Mitchell. An Introduction to Genetic Algorithms (ebook), MIT Press, 1999.

[Men05] Méndez Cruz C., Medina Urrea A. Extractive Summarization Based on Word Information and Sentence Position. CICLing, LNCS, Vol. 3406, Springer-Verlag, pp. 641-644, 2005.

[Mih04] Mihalcea, R., Tarau, P. TextRank: Bringing Order into Texts. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Spain, 2004.

[Mih06] Mihalcea R. Ramdom Walks on Text Structures. CICLing 2006, LNCS, Vol. 3878, pp. 249-262, Springer-Verlag, 2006.

[Mit97] Mitra M., Singhal Amit, Buckley C. Automatic Text Summarization by Paragraph Extraction. ACL/EACL Workshop on Intelligent Scalable Text Summarization, Spain, pp. 31-36, 1997.

[Mon01] Montes y Gómez M., Gelbukh A., López López A. Mining the News: Trends, Associations, and Deviations. Computación y Sistemas, Revista Iberoamericana de Computación, ISSN 1405-5546, Vol. 5 (1), pp. 14-24, 2001.

[Mon02] Montes y Gómez M., Gelbukh A., López López A. Text Mining at Detail Level Using Conceptual Graphs. In: Uta Priss, Dan Corbett, Galia Angelova (Eds.), Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures (ICCS), Bulgaria. LNCS, Springer-Verlag, ISSN 0302-9743, Vol. 2393, pp. 122-136, 2002.

[Mor91] Morris J., Hirst G. Lexical Cohesion Computed by Thesaurus Relations as an Indicator of the Structure of Text. Computational Linguistics, Vol. 18, pp. 21-45, 1991.

[Neg02] Negnevitsky M. Artificial Intelligence: A Guide to Intelligent Systems, Addison Wesley, Harlow, England, 2002.

[Nen04] Nenkova A., Passonneau R. Evaluating Content Selection in Summarization: The Pyramid Method. In: Proc. of NLT/NAACL-2004, 2004.

[Nen05a] Nenkova A., Siddharthan A., McKeown K. Automatically Learning Cognitive Status for Multi-Document Summarization of Newswire. Proc. of HLT/EMNLP-05, 2005.

[Nen05b] Nenkova A., Vanderwende L. The Impact of Frequency on Summarization (MSR-TR-2005-101). Redmond, Washington: Microsoft Research, 2005.

[Nen06] Nenkova A. Understanding the Process of Multi-Document Summarization: Content Selection, Rewriting and Evaluation. Ph.D. Thesis, Columbia University, 2006.

[Net04] Neto L., Freitas A., et al. Automatic Text Summarization using a Machine Learning Approach. Proc. of the ACL Workshop, España, 2004.

[Nad06] Nadjet Bouayad-Agha et al. A Sentence Compression Module for Machine-Assisted Subtitling. CICLing, LNCS, Vol. 3878, Springer-Verlag 2006.

[Ora05] Orasan C. Automatic Annotation of Corpora for Text Summarization: A Comparative Study. CICLing, LNCS, Vol. 3406, Springer-Verlag, pp. 658-669, 2005.

[Ott06] Otterbacher J., Radev D., Kareem O. News to Go: Hierarchical Text Summarization for Mobile Devices. SIGIR'06, ACM 1-59593-369-7/06/0008.

[Pas05] Passonneau R., Nenkova A., McKeown K., Sigleman S. Pyramid Evaluation at DUC-05. Proc. of Document Understanding Conference, <http://duc.nist.gov/pubs.html#2007>.

[Por80] Porter M.F. An Algorithm for Suffix Stripping. Program, Vol. 14, pp. 130-137, 1980.

[Rad03] Radev D., Tam D. Single-Document and Multi-Document Summary Evaluation via Relative Utility. Proc. of the ACM International Conference on Information and Knowledge Management (CIKM-03), 2003.

[Rad04] Radev D., Jing Hongyan, Budzikowska Malgorzata: Centroid-Based Summarization of Multiple Documents: Sentence Extraction, Utility-Based Evaluation and User Studies. Information Processing and Management, Vol. 40, pp. 919-938, 2004.

[Ree07] Reeve L. H., Han H. A Term Frequency Distribution Approach for the DUC-2007 Update Task. Proc. of Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Sal88] Salton G., Buckley C. Term-Weighting Approaches in Automatic Text Retrieval, Information processing and Management, Vol. 24, pp. 513-523, 1988.

[Sal89] Salton G. Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer. Addison Wesley Publishing Company, 1989.

[Sar05] Sarkar K. et al. Generating Headline Summary from a Document Set. CICLing, LNCS, Vol. 3406, Springer-Verlag, pp. 637-640, 2005.

[Sek02] Seki Y. Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles. Proc. of the Third NTCIR Workshop, 2002.

[Shr04] Shrestha L., McKeown K. Detection of Question-Answer Pairs in Email Conversation. Proc. of Coling 2004, Switzerland, 2004.

[Sid05] Sidorov G. Proyecto Conacyt: Diccionarios dinámicos orientados al dominio como una herramienta para medición y representación de tendencias de desarrollo de disciplinas científicas y tecnologías e interacciones entre ellas, 2002-2005.

[Sil02] Silber H., McCoy K. Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization. Computational Linguistics, 28 (4), pp. 487-496, 2002.

[Son04] Song Y., et al. A Term Weighting Method Based on Lexical Chain for Automatic Summarization. CICLing 2004, LNCS, Vol. 32945, Springer-Verlag, pp. 631-634, 2004.

[Sor03] Soricut R., Marcu D. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In: HLT-NAACL, 2003.

[Teu04a] Teusel S., Halteren H. van. Agreement in Human Factoid Annotation for Summarization Evaluation. Proc. of the 4th Conference on Language Resources and Evaluation (LREC), 2004.

[Teu04b] Teusel S., Halteren H. van. Evaluating Information Content by Factoid Analysis: Human Annotation and Stability. EMNLP, pp. 419-426, 2004.

[Una08] Group of NLP of Universidad Nacional Autónoma de México headed by Galicia Haro Sofia N.

[Una05] Una-May O' Reilly et al. (Eds.) Genetic Programming Theory and Practice II, Springer, 2005.

[Van04] Vandeghinste V., Pan Y. Sentence Compression for Automated Subtitling: A Hybrid Approach. Proc. of ACL Workshop on Summarization, pp. 89-95, 2004.

[Ver04] Verbeek Jakob. Mixture Models for Clustering Dimension Reduction. Thesis, ISBN 90-776-125-4, 2004.

[Ver05] Vergara-Villegas O., García-Hernández R. A., Carrasco-Ochoa J. A., Pinto R., Martínez-Trinidad J. F. Data Preprocessing by Sequential Pattern Mining for LZW, 6th Mexican International Conference on Computer Science (ENC), IEEE Computer Society Press, 2005.

[Ver07] Verma R., Chen P. A Semantic Free-Text Summarization System Using Ontology Knowledge. Proc. of Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Vil03] Villaseñor-Pineda L., Montes-y-Gómez M., Vaufreydaz D., Serignat J-F. Elaboración de un corpus balanceado para el cálculo de modelos acústicos usando la web. Proc. of the International Conference on Computing, 2003.

[Vil04] Villaseñor-Pineda L., Montes-y-Gómez M., Vaufreydaz D., Serignat J-F. Experiments on the Construction of a Phonetically Balanced Corpus from the Web. Lecture Notes in Computer Science, ISSN 0302-9743, Springer-Verlag, Vol. 2945, pp. 410-413, 2004.

[Vil06] Villatoro-Tello E., Villaseñor- Pineda L., and Montes-y-Gómez M. Using Word Sequences for Text Summarization. 9th International Conference on Text, Speech and Dialogue (TSD). Lecture Notes in Artificial Intelligence, Springer-Verlag, 2006.

[Voo04] Voorhees Ellen M. Overview of the TREC-2003 Question Answering Task. In: Proc. of the 12th Text Retrieval Conference (TREC-2003), pp. 54-68, 2004.

[Wan04] Wan S., McKeown K. Generating "State-of Affairs" Summaries of Ongoing Email Thread Discussions. Proc. of Coling 2004, Switzerland, 2004.

[Wan05] Wang R., et al. LexTrim: A Lexical Cohesion Based Approach to Parse-and-Trim Style Headline Generation. CICLing 2005, LNCS, Vol. 3406, Springer-Verlag, pp. 633-636, 2005.

[Wan06] Wan X., Yang J. et al. Incorporating Cross-Document Relationships Between Sentences for Single Document Summarizations. ECDL, LNCS, Vol. 4172, Springer-Verlag, pp. 403-414, 2006.

[Wei06] Xu Wei, Li Wenjie et al. Deriving Event Relevance from the Ontology Constructed with Formal Concept Analysis. CICLing 2006, LNCS, Vol. 3878, Springer-Verlag, pp. 480-489, 2006.

[Yin07] Ying J.-C., Yen S.-J., Lee Y.-S. Language Model Passage Retrieval for Question-Oriented Multi Document Summarization. Proc. of Document Understanding Conference 2007, <http://duc.nist.gov/pubs.html#2007>.

[Zho05] Zhou Q., Sun L. IS_SUM: A Multi-Document Summarizer based on Document Graphics and Lexical Chains. Proc. of Document Understanding Conference 2005, <http://duc.nist.gov/pubs.html#2005>.

Appendices

Appendix A. List of stop-words

a	doesn’t	is	other	there
about	done	isn’t	our	theirs
after	due	it	out	them
again	during	its	over	then
all	each	it’s	overall	there’s
almost	either	itself	per	these
also	enough	just	perhaps	they
although	especially	kg	possible	this
always	etc.	km	previously	those
am	even	largely	quite	through
among	ever	like	rather	thus
an	first	made	really	to
and	followed	mainly	regarding	under
another	following	make	resulted	until
any	for	may	resulting	up
approximately	found	max	same	upon
are	from	me	seem	use
as	further	might	seen	used
at	give	more	several	using
be	given	most	she	various
because	giving	mostly	should	very
been	had	must	show	was
before	hardly	my	showed	we
being	has	myself	shown	were
between	have	nearly	shows	what
both	having	neither	significant	when
but	here	no	significantly	whereas
by	he	nor	since	which
can	he’s	not	so	who
can’t	her	now	some	while
could	his	obtain	somehow	with
couldn’t	how	obtained	such	within
did	however	of	suggest	without
didn’t	if	often	than	would
do	I’m	on	that	you
don’t	in	only	the	
does	into	or	their	

Appendix B. Examples of results

Detailed (complete) results for the best three results from Table IV.2 (see experiment 1):

First result: *M*, 1, *best*
1 ROUGE-1 Average_R: 0.44128 (95%-conf.int. 0.43352 - 0.44889)
1 ROUGE-1 Average_P: 0.45609 (95%-conf.int. 0.44790 - 0.46415)
1 ROUGE-1 Average_F: 0.44840 (95%-conf.int. 0.44047 - 0.45615)

1 ROUGE-2 Average_R: 0.18676 (95%-conf.int. 0.17845 - 0.19498)
1 ROUGE-2 Average_P: 0.19341 (95%-conf.int. 0.18455 - 0.20230)
1 ROUGE-2 Average_F: 0.18994 (95%-conf.int. 0.18135 - 0.19849)

1 ROUGE-SU4 Average_R: 0.20883 (95%-conf.int. 0.20138 - 0.21582)
1 ROUGE-SU4 Average_P: 0.21618 (95%-conf.int. 0.20873 - 0.22331)
1 ROUGE-SU4 Average_F: 0.21235 (95%-conf.int. 0.20483 - 0.21947)

Second Result: *W*, *f*, *best*
1 ROUGE-1 Average_R: 0.44609 (95%-conf.int. 0.43850 - 0.45372)
1 ROUGE-1 Average_P: 0.45953 (95%-conf.int. 0.45160 - 0.46749)
1 ROUGE-1 Average_F: 0.45259 (95%-conf.int. 0.44479 - 0.46048)

1 ROUGE-2 Average_R: 0.19451 (95%-conf.int. 0.18664 - 0.20256)
1 ROUGE-2 Average_P: 0.20048 (95%-conf.int. 0.19229 - 0.20892)
1 ROUGE-2 Average_F: 0.19740 (95%-conf.int. 0.18936 - 0.20566)

1 ROUGE-SU4 Average_R: 0.21420 (95%-conf.int. 0.20755 - 0.22133)
1 ROUGE-SU4 Average_P: 0.22085 (95%-conf.int. 0.21387 - 0.22813)
1 ROUGE-SU4 Average_F: 0.21742 (95%-conf.int. 0.21061 - 0.22462)

Third result: *W*, *f*, *1best+first*
1 ROUGE-1 Average_R: 0.46576 (95%-conf.int. 0.45877 - 0.47292)
1 ROUGE-1 Average_P: 0.48278 (95%-conf.int. 0.47547 - 0.49004)
1 ROUGE-1 Average_F: 0.47399 (95%-conf.int. 0.46693 - 0.48132)

```
1 ROUGE-2 Average_R: 0.21690 (95%-conf.int. 0.20915 - 0.22497)
1 ROUGE-2 Average_P: 0.22495 (95%-conf.int. 0.21659 - 0.23345)
1 ROUGE-2 Average_F: 0.22080 (95%-conf.int. 0.21278 - 0.22909)

1 ROUGE-SU4 Average_R: 0.23330 (95%-conf.int. 0.22668 - 0.24045)
1 ROUGE-SU4 Average_P: 0.24207 (95%-conf.int. 0.23508 - 0.24941)
1 ROUGE-SU4 Average_F: 0.23754 (95%-conf.int. 0.23075 - 0.24472)
```

Appendix C. Examples of Maximal Frequent Sequences

Flights were cancelled
Sunday night
Prensa Latina
Civil Defense
Hurricane Gilbert
The Dominican Republic
The south coast
The national weather service said
The Cayman Islands
Cancun and Cozumel
In Mexico City
Quintana Roo state
Over the water
Roamed the streets of Cancun
The Yucatan peninsula
Tropical storm
Low pressure
Caused coastal flooding
San Francisco area
Have to pay
Have earthquake insurance
Insurance companies
Long term
A special session
To deal with
Department of transportation
Gasoline tax increase
Might collapse in an earthquake
The White House
The California earthquake
Bush and his aides
The insurance industry
Exposure to catastrophes
On an inflation adjusted basis
Structural damage
State farm
Personal property

Earthquake insurance
Plenty of experience
Year after year
For the purpose of
Whenever I needed him
The Royal Marines School of Music
Military installations
Private security
Irish Republican Army
Opening day record for
Restaurant in Moscow
The Soviet Union
The Big Mac
Moscow McDonalds
Above and beyond the usual guest
Vice president of the United States
Wal Mart discount city
Most important retailer of his generation
British Prime Minister John Major
Vote for Major
Major was elected
Was elected to Parliament
Leader of the Conservative Party
Associated Press

Appendix D. Examples of generated summaries

Original text A

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. “There is no need for alarm”, Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday. Cabral said residents of the province of Barahona should closely follow Gilbert’s movement. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo. The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a “broad area of cloudiness and heavy weather” rotating around the center of the storm. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday. Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico’s south coast. There were no reports of casualties. San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night. On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast. Residents returned home, happy to find little damage from 80 mph winds and sheets of rain. Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane. The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

Model summary 1

Tropical Storm Gilbert in the eastern Caribbean strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15mph with a broad area of cloudiness and heavy weather with sustained winds of 75 mph gusting to 92 mph. The Dominican Republic’s Civil Defense alerted that country’s heavily populated south coast and the National Weather Service in San Juan, Puerto Rico, issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Model summary 2

Hurricane Gilbert is moving toward the Dominican Republic, where the residents of the south coast, especially the Barahona Province, have been alerted to prepare for heavy rains, and high winds and seas. Tropical Storm Gilbert formed in the eastern Caribbean and became a hurricane on Saturday night. By 2 a.m. Sunday it was about 200 miles southeast of Santo Domingo and moving westward at 15 mph with winds of 75 mph. Flooding is expected in Puerto Rico and the Virgin Islands. The second hurricane of the season, Florence, is now over the southern United States and downgraded to a tropical storm.

System summary (obtained automatically using the proposed method from experiment 1)

The national weather service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm. Hurricane Gilbert swept toward the Dominican Republic Sunday, and the civil defence alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday. Cabral said residents of the province of Barahona should closely follow Gilbert's movement.

Original text B

The Irish Republican Army claimed responsibility for a huge explosion Friday that reduced a three-story military barracks on the southeast coast of England to rubble, killing 10 people and injuring 22, eight seriously. It would be one of the outlawed IRA's deadliest attacks on the main British island. Nine marine musicians and one civilian died in the blast, which also damaged dozens of nearby homes and could be heard two miles away. The musicians were between the ages of 16 and 20 as are most of the recruits in the school. A police spokesman said forensic experts are still trying to determine with certainty that the explosion was the result of a bomb. But he said the characteristics of the blast and a statement claiming responsibility appeared to confirm that it was the work of the IRA. Security sources said they believe that at least two IRA "active service" units, each composed of four or five members, are operating in Britain and continental Europe. One member, known as the "Jackal" after the assassin in the Frederick Forsyth novel "The Day of the Jackal," has been eluding the authorities for two years. He has been identified as Patrick Sheehy and has been linked to the IRA's last successful mainland bombing attack–on an army barracks at Mill Hill in August, 1988. One soldier was killed in that incident. Sheehy and another wanted Irishman, John Conaghty, were linked to an IRA bomb factory in North London that the police stumbled upon last December while in pursuit of a car thief. A search turned up automatic and semiautomatic weapons, ammunition, 150 pounds of Semtex high explosive and a "hit list" of 100 British political figures and other officials headed by Prime Minister Margaret Thatcher. Friday's explosion occurred about 8:30 a.m. in a lounge at the Royal Marines School of Music near Deal, on the English Channel in the county of Kent. At the school are about 250 recruits who receive military and musical training before joining Royal Marines bands. The roof of the three-story barracks collapsed, trapping victims beneath the rubble. Firefighters used thermal cameras and dogs to search the debris for victims and survivors. Heavy lifting gear was brought to the scene from a nearby site where a tunnel is being built beneath the English Channel. Rescuers shouted for quiet as they used high-technology listening equipment in an effort to trace the sound of faint heartbeats. "I looked up from the sink and I just saw the whole building explode," Heather Hackett, a 26-year-old Deal housewife, told the British Press Assn. She said she told her children to run for cover, but as they did, her kitchen window shattered. "The whole window was blown across the kitchen," Hackett recalled. Her 2-year-old son, Joshua, was hit by a shard that embedded itself in his back but caused no serious injury. "I just screamed and ran out of the room," she said. "The bang was so loud I thought the whole house was coming in." 'Appalling Outrage' Defence Secretary Tom King visited the scene and called the bombing "an appalling outrage committed against unarmed bandsmen"–people who worked for charity, who have given great enjoyment to millions right across the country, right across the world. "The real evil of these murders is that the people who commit them, the 'godfathers' who send them to commit them, know that they will actually achieve nothing. Terrorism is not going to win. We shall find the people responsible for this outrage sooner or later, as we have already found some of those responsible for the earlier outrages, and they will be brought to justice." The authorities have been on high alert, expecting IRA attacks in connection with last month's 20th anniversary of the introduction of British troops into Northern Ireland. The republican underground organization opposes British rule in the predominantly Protestant province and is fighting to join the mainly

Roman Catholic south in a united, independent Ireland. Visit to Ulster But in a statement telephoned to a Dublin news agency, Ireland International, Friday's attack was linked to Thatcher's visit last week to units of the controversial Ulster Defence Regiment in Northern Ireland. The locally recruited, overwhelmingly Protestant Ulster Defense Regiment has come under fire in connection with an investigation into the leak of secret government lists of suspected IRA members to Protestant assassination squads. It is widely hated by the Catholic minority in the province, and the Irish government in Dublin has urged Britain to disband the force. "Mrs. Thatcher visited Ireland with a message of war at a time when we want peace," the statement claiming responsibility for the Deal attack said. "Now in turn we have visited the Royal Marines in Kent. But we still want peace, and we want the British government to leave our country." The statement was signed "P. O'Neill, Irish Republican Publicity Bureau," a signature that has appeared on earlier IRA bombing claims. Friday's attack was the worst on the mainland since the virtually simultaneous bombings of July, 1982, directed at ceremonial military units in London's Hyde Park and Regent's Park. Eleven bandsmen and mounted guards were killed in those incidents. Eight persons were killed by IRA car bombs outside Harrods department store here in December, 1983, and 21 were killed and 162 injured in two Birmingham public house bombings in the fall of 1974. An attempted barracks bombing was averted last February when a sentry came upon two intruders who had managed to get inside a military camp in Shropshire. There has been a series of bomb and automatic rifle attacks this year on British soldiers and their families stationed in West Germany. Earlier this month an IRA gunman shot to death an army wife, Heidi Hazell, 25, in her car near her home at Dortmund.

Model summary 1

A huge explosion yesterday in the lounge of the Royal Marines School of Music killed 10 and injured 22, eight seriously. The School is located in Deal on the English Channel. Eyewitness accounts of neighbors attest to the strength of the blast. Investigators said that it was probably a bomb blast, and the IRA has claimed responsibility. The British think that at least two IRA "active service" units, each with four or five members, operate in Britain and continental Europe. Increased IRA activity had been anticipated because last month marked the 20th anniversary of British troops entering Northern Ireland.

Model summary 2

In what they said was a response to Prime Minister Thatcher's "declaration of war" in a speech to the Ulster Defense Force, the Irish Republican Army claimed responsibility for an explosion which leveled a three-story barracks in Deal, killing 10 and injuring 22. The barracks, which belonged to the Royal Marines Music School, was the latest in a series of IRA bombings of military facilities. Security forces believe at least two IRA "active service units" are operating in Britain and Europe. Two members of these groups, Patrick Sheehy, known as the Jackal, and John Conaghty are being sought in connection with earlier attacks.

System summary (obtained automatically using the proposed method from experiment 1)

The Irish Republican Army claimed responsibility for a huge explosion Friday that reduced a three-story military barracks on the southeast coast of England to rubble, killing 10 people and injuring 22, eight seriously. It would be one of the outlawed IRA's deadliest attacks on the main British island. Nine marine musicians and one civilian died in the blast, which also damaged dozens of nearby homes and could be heard two miles away. The musicians were between the ages of 16 and 20 as are most of the recruits in the school. A police spokesman said forensic experts are still trying to determine with certainty that the explosion was the result of a bomb.

Automatic Text Summarization with Maximal Frequent Sequences se terminó de imprimir el 15 de diciembre de 2013, en los talleres de Ediciones Verbolibre, S.A. de C.V., 1o. de mayo núm. 161-A, Col. Santa Anita, Deleg. Iztacalco, México, D.F., C.P. 08300. Tel.: 3182-0035. <edicionesverbolibre@gmail.com>. La edición consta de 500 ejemplares.