Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Loïs RIGOUSTE

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Evolution of a Text Summarization System in an Automatic Evaluation Framework

N. Japkowicz

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

S. Spakowicz

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

J.-P. Corriveau

H. Victor

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

EVOLUTION OF A TEXT SUMMARIZATION SYSTEM IN AN AUTOMATIC EVALUATION FRAMEWORK

Loïs Rigouste
Supervisors: Nathalie Japkowicz and Stan Szpakowicz

September 19, 2003

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-612-90347-8
Our file Notre référence
ISBN: 0-612-90347-8

Abstract

CALLISTO is a text summarizer that searches through a space of possible configurations for the best one. This is different from other systems since it allows CALLISTO 1) to choose adequate components based on results obtained on the training data (and thus, to choose a configuration better adapted to the problem) and 2) to allow different texts to be summarized in different ways. The purpose of this thesis is to find out how the initial space CALLISTO explores can be modified to improve the overall quality of the summaries produced.

The thesis reviews and evaluates the first arbitrary design choices made in the system, through a fully automated framework based on a content measure proposed by Lin and Hovy. We tried different modifications to CALLISTO such as replacing the internal evaluation measure, testing other discretization processes, changing the learning algorithm or adding new features to characterize the input text. We found that Naive Bayes outperformed the current learner C5.0, by identifying one configuration working satisfactorily for all texts.

Contents

List of Figures

List of Tables

Acknowledgements

- Many thanks to my supervisors for their support and helpful advice. Many ideas in this thesis are theirs.
  - Thanks to Nathalie Japkowicz for her enthusiastic comments and constructive criticism. Nathalie was supportive even in times when her responsibilities to her baby required most of her attention.
  - Thanks to Stan Szpakowicz for his quick answers and rigorous methodology. Stan showed me that you need to work until your arguments are perfect and that there is no such thing as a perfect argumentation!
- Grateful thanks to Terry Copeck, a great researcher and programmer, who was always quick and reliable when I needed a favour. Without his help, this thesis would not have been possible.
- Thanks to Stan Matwin for his judicious suggestions.
- Thanks to Chin-Yew Lin, Eduard Hovy, Chris Paice and Steffi Brueninghaus for answering my e-mails quickly and accurately.
- Thanks to all my lab-mates and colleagues who provided a wonderful working environment and who were always available for help and discussion. I would especially like to thank Fernanda, Vivi, Mario, Nicolas, Marina, Mohak, Jelber and Magda.
- Finally, thanks to all the welcoming people I met in Canada, who made my stay so enjoyable and unforgettable, in spite of the winter that lasted way too long.

Resources Used

This thesis was written entirely in LaTeX. The scripts used in the various experiments and in the automatic evaluation framework were written in Perl and are all available on request from the author (rigouste@aist.enst.fr).
The CALLISTO text summarization program has been written in Perl by Terry Copeck, from the University of Ottawa. I also used Porter's stemmer implementation by Steffi Bruninghaus from the University of Pittsburgh.
The two figures in chapter 3 were done with Unix XFig. The two figures in chapter 4 were done with Matlab 6.0. The other graphs in the thesis were drawn with Gnuplot 3.7.3 and its latex output terminal.
Rulequest C5.0 1.16 is used inside CALLISTO and was run for the Adaboost experiments, among others. In chapter 6, we used the following implementations of learning algorithms:

- Stuttgart Neural Networks Simulator 4.2, available from http://www-ra.informatik.uni-tuebingen.de/SNNS/
- Ripper 2.5 C-code written by W. Cohen
- SVM^{light} 5.0, available from http://svmlight.joachims.org/

Lastly, we used the Perl package Math::MatrixReal 1.9 for the Regression computations (http://www.leto.net/code/Math-MatrixReal/).

Chapter 1

Introduction

1.1 Automatic Text Summarization

There exist several definitions of Automatic Summarization. The one given by Mani in [Mani, 2001a], on page 1, identifies the task through its goal: "to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs". This presentation is very broad and can cover other similar tasks like multimedia summarization. Also, it emphasizes the need to take into account the user's or application's needs, following [Sparck Jones, 1998], who stated the importance of context factors in the whole summarization task. Sparck Jones, on page 1 in [Sparck Jones, ibid.], focuses on the methods: a summary is "a reductive transformation of source text to summary text through content reduction by selection and/or generalization on what is important in the source." The definition of Kupiec, Pedersen and Chen [Kupiec *et al.*, 1995], page 1, is more user-oriented: "The goal is to generate a concise document description that is more revealing than a title but short enough to be absorbed in a single glance." Boguraev and Kennedy [Boguraev and Kennedy, 1997] (page 2) criticize the underlying assumptions of the above definitions: "To a large extent, the shared (and for the most part unspoken) intuition is that summaries are to documents very much what abstracts are to full length articles. This reflects another pervasive assumption: namely that there is a canonical, definitive (or at least optimal) summary for any document." Then they list a few counter-examples where efficient summaries are not only texts, such as library indexes or tables of content.
One obvious application of Automatic Summarization is the World Wide

Web, the users looking for an efficient way to browse a large amount of information. The combination of a search engine with a good text summarizer would be a satisfactory solution and would help identify very quickly if a document is relevant or not. For this application, the summary needs not to cover the most important aspects of the source but rather give a good overview of what it is about[1]. Note that, besides, the summaries generated can take the query of the user into account to guarantee that the piece of text shown will actually help him to determine whether or not the document is useful for him. More generally, query-oriented Automatic Summarization can help in all cases when one needs to go through a large collection of texts for a particular information not necessarily covered by the general-purpose human-written abstracts, for instance proceedings of conferences or medical literature on a certain topic, as described in [Mani, ibid.]. Mani also evokes even higher-level applications, involving tasks more difficult than single text summarization, such as "intelligence gathering" (retrieving and summarizing information about a given topic) or "search engine hits" (summarizing the list of documents output by a search engine to produce an overview entirely readable by the user in a limited amount of time).
We will focus on Text Summarization, where the input is a series of paragraphs, a book or a news article, whatever the length is, and the output is an organized text dealing with the same topic and significantly shorter. The length of the summary is generally expressed as a fraction of the length of the original text, which is called the *compression rate*. Typical compression rates are between 15% and 25% although, as pointed out by Mani, these figures, valid when working on short (a few thousands words) news articles, may not be suitable when working on much longer documents.
The process of summarization is decomposed into three basic stages by Sparck Jones [Sparck Jones, 1998]:

- Interpretation: Understanding, more or less deeply, what the source is about and how it is articulated.
- Transformation: Deciding which ideas to eliminate and which ones to emphasize.
- Generation: Building sentences for the final summary, based on the structure produced at the previous stage.

This framework is very general and we will see that Automatic Summarizers are rarely divided into three such independent modules. The minimal

[1] In this case, as we will clarify later, we talk about *indicative* summaries.

elementary sub-tasks are in general much more low-level than those and, according to the methodology chosen, one of the stages may be more important than the others whereas another could be skipped altogether[2]. We will now present various definitions related to Automatic Summarization, thereby discussing the diversity of possible methods and output forms.

1.2 Definitions

1.2.1 Abstract vs. Extract

The distinction between *abstracts* and *extracts* depends on how different from the source the vocabulary used is. More specifically, Mani [Mani, ibid.] proposes the following definition on page 6: "An abstract is a summary at least some of whose material is not present in the input."
This definition is accurate but might be a bit too restrictive. An extract would indeed be only a sequence of sentences from the original text. It is possible to choose larger units such as paragraphs and produce a summary exclusively containing material from the source all the same, but not smaller units. If one wants to select only pieces of sentences or phrases, she will have to apply some pasting strategy on this material and this kind of re-writing can be viewed as producing "material not present in the input."
There are various degrees of re-writing. Thus, while it is undeniable that operations like generalization[3] are beyond the possibilities of extracts, it seems like simpler modifications such as truncations of parts of sentences, like subordinate phrases or adjectives, are less demanding and produce outputs close to extracts. That is why Mani [Mani, ibid.] identifies intermediate processing between extracting and abstracting, called *text compaction*, which covers that case. This form is often used in automatic summarization, dividing the task into two main steps: identifying first the most relevant sentences in the text and then conducting a work of revision on the resulting extract.
[Oakes and Paice, 1999] sums up the link between those definitions and the different methods they entail in automatic summarization on page 1: "The two traditional approaches to automatic abstracting are:

1. Extraction, where specific sentences are selected from the source text if they contain significant words or possess other "importance cues"

[2]It is the case in *Sentence Extraction* for instance, where the generation stage is absent or, at least, can be seen as merged with the transformation stage.

[3]For example, referring to a series of similar terms like "tigers, lions and panthers" by a generic term such as "felines".

such as position within the text, e.g. Pollock and Zamora's ADAM system [Pollock and Zamora, 1975]. The problems with this approach are that importance cues are often not reliable, and that the extracted sentences do not always constitute a coherent text.

2. Summarization, where a representation of the original document such as a conceptual dependency graph or a semantic net is produced after a detailed semantic analysis of the text, such as Rau's SCISOR system [Rau, 1987]. This approach requires a very large knowledge base and tends to be domain specific."

1.2.2 Indicative, Informative and Critical Summaries

The classical distinction between *indicative* and *informative* abstracts [Mani, ibid.] is linked to what the summary is intended to do. An indicative abstract is only meant to give an idea of the main topic of the document or, more exactly, the main contributions of the text to the topic. It is not meant to replace the source but to help the reader decide if it is worth investigating in more detail. In that respect, indicative abstracts are sufficient for combinations with information retrieval search engines, an application we were discussing above.
On the contrary, informative abstracts are intended to replace the source and cover all important aspects of it. However, this definition is somewhat unrealistic and usual texts are not redundant to the point that you can condense all the ideas in 25% of the original length. Therefore, an informative abstract is usually seen as one grasping most of the salient information from the text. This is highly subjective and there is a continuum between an abstract totally indicative and an abstract wholly informative, most of the summaries being both. The example Mani [Mani, ibid.] gives to help understand the difference between the two, comes from the American National Standards Institute (ANSI) standard. "In the case of reports of scientific investigations, indicative abstracts should contain information about an article's purpose, scope, and approach, but not results, conclusions, and recommendations; informative abstracts, on the other hand cover all these different aspects."
A third possible category for abstracts, but rather complementary to the last two, is *critical evaluative* [Mani, ibid.]. This kind of summary not only presents the content of the source but also gives comments about it. This definition covers most of the critical review such as journalists' book or movie reviews for instance. We mention it for the sake of exhaustivity but it

almost does not exist in Automatic Summarization and will not be referred to in the remainder of the thesis.

1.2.3 Salience and Coherence

We have an intuitive notion of what a good abstract is when we read one. However, Automatic Summarization needs to formalize a bit more the idea of quality of an abstract. In an attempt to do so, Mani [Mani, ibid.] suggests two definitions, one regarding informativeness and the other cohesion.
The *salience* of a sentence (or any other unit) in a document is "the weight" attached to this sentence and measures how much it represents of the whole text or its relevance with respect to a given query. More informally, Marcu simply explains in [Marcu, 1999] on page 3 that "the salient units are the most important units in the corresponding text span." [Boguraev and Kennedy, 1997] gives a more technical definition as the "relationship between the distributional prominence of linguistic expressions, computed as a function of their occurrence in the text, and the topical prominence of the objects and the events they refer to." Now, how this weight is computed practically is one of the main problem in Automatic Summarization.
Besides this idea of relevance, a good abstract is also *coherent*, if it reads well, does not present redundant information or dangling references. The last problem is most typical of automatic extracts since it often results from the extraction of two sentences which are not consecutive in the initial text. They may contains identical anaphora (such as *it*) referring to different entities, with no way of disambiguating without the intermediate sentences from the source.
This distinction between informativeness and readability will be explained in further details in Chapter 2, when we will discuss Evaluation.

1.2.4 User-, topic- or query-focused vs. generic summaries

This distinction relies on the assumption prior to summarization and the aims of the abstract. If we know what the summary will be used for, we can favour some ideas over others. For example, if we were to summarize a national weather forecast and if we knew the region the reader lives in, a *user-focused* summary could describe more of the local weather and give only a brief overview for the rest of the country.
Similarly, a topic-focused summary presents in more detail the passage of the source related to a particular theme. The idea of query-focused summaries

is again related to Information Retrieval: if we want to propose to the user a summary with every document retrieved, we can use his query to make the suggested abstract or extract more relevant to his needs. For instance, if a user types the query "Skating, Ontario" and a tourist brochure about Ottawa is retrieved, our reader will likely be more interested by the Canal in Winter than Parliament Hill on Canada Day[4].
Finally, if we have no clue about what the user wants or if our summary is intended to be indicative of the whole document and not focusing on some parts of it, we say that the output has to be *generic.*

1.2.5 Shallow vs. Deep approaches

Still following Mani [Mani, ibid.], but now focusing more on Automatic Summarization in its own rights rather than abstracting in general, we can make a difference between systems according to the level of semantic knowledge or deep natural language processing techniques they use. Shallow approaches do not go linguistically beyond the syntactic level and most often rely on statistical and/or Machine Learning techniques. On the contrary, deeper approaches involve some semantic understanding of the source, for example building trees representing the rhetorical structure [Marcu, 1999]. [Aone *et al.*, 1997] gives the following analysis, on page 1: "Summarization Research and System development can be broadly characterized as frequency-based, knowledge-based or discourse-based. These categories correspond to a continuum of increasing understanding of a text and increasing complexity in text processing."
This distinction is particularly important in Automatic Summarization because it is an area of Natural Language Processing in which both techniques (shallow and deep) have proven to bring interesting results, the former in general outputting extracts whereas the latter are more likely to produce abstracts. This also will be studied in more detail in the next chapter.

1.3 Outline

In this thesis, we will consider an automatic summarizer designed at the University of Ottawa [Copeck *et al.*, 2002]: CALLISTO. There are too many parameters in the system to find the most adequate set quickly and manually. Therefore, the goal of the thesis is to find a way to study and reliably improve the performance of CALLISTO.

[4]Both are great but our user asked for skating!

This first chapter has introduced vocabulary and ideas related to automatic summarization or even to summarization in general. The second chapter attempts to establish a more exhaustive literature review on the topic and also tackles the problem of evaluation in Natural Language Processing and in Automatic Summarization. This is because we will need a way to make sure that we have actually improved the system. We want to improve the performance of CALLISTO on a corpus of texts in a statistically significant way. Therefore, checking a few documents is not satisfactory.
Chapter 3 describes CALLISTO in more detail, trying to make the distinction between the Natural Language and Machine Learning parts of the system and how they fruitfully interact with one another. Chapter 4 is about our evaluation framework based on a measure of similarity between an abstract and a reference summary proposed by Lin and Hovy [Lin and Hovy, 2002]. We also talk about the limitations of this method and how we integrate them into our methodology.
Chapter 5, 6 and 7 deal with modifications done to the system regarding the different degrees of freedom. Chapter 5 is about the measures, discretization process and selection of configurations. We show how we assumed a perfect learning algorithm to conduct some experiments related to the Machine Learning part of the system but not directly to the learning algorithm. Chapter 6 explains how we tested several learning algorithms other than C5.0 [Quinlan, 1997], the one currently used in the system. We show that Naive Bayes outperforms C5.0 for our application. Chapter 7 is about miscellaneous smaller experiments regarding feature selection and the way the different tools inside the system have an influence on its global performance. Chapter 8 concludes on the findings of this thesis and proposes some directions for future work.

Chapter 2

Literature review

Automatic Summarization and Summary Evaluation

2.1 Introduction

The first research attempts in Automatic Text Summarization go back to the late fifties and Luhn's article [Luhn, 1958]. In Chapter 1, we have focused on the diversity of outputs, methods and applications in the area. However, as noted in [Sparck Jones, 1998], most of the practical achievements were obtained in *text extraction* and in *fact extraction*. The former is about extracting the most relevant units, often sentences, from the text whereas the latter consists in filling in a pre-defined template extracting knowledge from the source and outputting this information either as such or processed by a text generation phase. Neither of the methods seems to be that much better than the other and Sparck Jones [Sparck Jones, 1998] summarizes the pros and cons of both methods very simply (page 3): "it is necessary to get more power in automatic summarising than text extraction delivers, and more flexibility than fact extraction offers."
In this study, we will limit our attention to Sentence Extraction Systems since CALLISTO extracts sentences. Indeed, abstract creation is a different task altogether and for the rest of our work, only the different techniques

used in extraction, and eventually applicable to our system, will be of interest.
A large part of this thesis regards evaluation of the quality of the summaries produced. Therefore, the last sections of this literature review will be dedicated to Evaluation, first in Natural Language Processing in general and then in Automatic Summarization.

2.2 Sentence Extraction

2.2.1 Advantages and Drawbacks of Sentence Extraction

One obvious advantage of Sentence Extraction in Text Summarization is that it enables us to use the author's syntax and leaves to the program only smaller tasks of formulation to handle, if any. Thus, the problem is turned into finding the most relevant parts of a text and the output is only the $C\%$ most interesting sentences where $C\%$ is the desired compression rate.
Why work on sentences and not on other units? As pointed out in [Mani, 2001a], sentences are the smallest coherent units. That is to say that working with smaller units (phrases or words) would require the effort of building sentences, which we want to avoid: we would need a large amount of syntactic and semantic knowledge in order to make a coherent text from a few keywords or keyphrases. On the other hand, larger units such as paragraphs or sections are not always linguistically meaningful[1] and, above all, offer less flexibility in terms of extracting the right number of units to fit the compression rate.
The method of sentence extraction however has intrinsic limits. The most important may be that it does not offer any guarantees regarding readability or global coherence of the extract. One recurrent problem indeed is the one of dangling references. Sentences are not independent from one another[2]. The final extract may contain pronouns without their referent, because one sentence has been extracted without the previous one, or some phrases such as "however" or "therefore" may be out of context and thus irrelevant or maybe even misleading. Hence the idea of extracting larger basic units. But even though it would reduce the number of such dangling references, some

[1] On page 46, Mani points out that paragraphs are not "a traditional linguistic unit" but often "reflect publishing and formatting conventions".

[2] This dependence may moreover be used to build Automatic Summarization Systems. For instance, Marcu [Marcu, 1999] identifies semantic relationships between clauses such as causality, elaboration or concession. This kind of approach relies heavily on Discourse Theory.

might still remain in the final extract. Another solution is to use various *fixing* strategies to identify anaphora and dangling phrases (see for example [Mani *et al.*, 1999] and the section "Revision" in this Chapter).
Other criticism can be addressed to the choice of sentence extraction for summarization: it is likely not to be applicable for book summarization for instance or multiple document summarization, in other words, every group of texts for which it is not proven that a few sentences can offer a global understanding of what is said. It seems that summarizing at very high compression rate is a different task implying advanced semantic knowledge or understanding [Mani, 2001a]. We are less concerned about this kind of issue though, since we are primarily working on single news articles for which we know that several acceptable summaries can be obtained by sentence extraction[3].
Many sentence extraction systems make use of machine learning techniques and therefore need to be trained on a corpus. The expectations of this training phase are that some patterns will emerge from the data and will thus help to predict the right parameters or rules to apply on an unseen text. Obviously, this operation takes advantage of regularities found in the training data, which may not all be valid for any new document. This kind of over-fitting[4] is likely to be impossible to avoid since every genre has its own characteristics and, therefore, various rules for summarization[5]. This justifies the concept of training the system on a corpus, presenting specific learnable features: it seems impossible to construct sentence extractors applicable to any kind of document[6].
Now, we will have a closer look at two articles which pioneered Automatic Summarization:

- [Edmundson, 1969] presented basic principles to guide sentence selec-

[3]And even, quite often, extraction of the first sentences of the text leads to a good summary, as we will see in the section "Baseline" of chapter 4.

[4]The meaning of over-fitting here is close to the usual Machine Learning sense. When a learner focuses too much on too few data and identifies properties only valid for this training test, sometimes even for only one example, it does not perform well on the test set.

[5]For example, as said before, in a news article, the first lines may be the best summary whereas, according to Mani [Mani, 2001a], in a scientific article, better chances are to find relevant information in the conclusion.

[6]Above all, because, as said before, sentence extraction itself is not applicable to every kind of documents. More generally, as highlighted by Sparck Jones [Sparck Jones, 1998], no automatic summarizer may be applicable to all kind of document. Hence we have to determine specific context factors for every system (see the beginning of Chapter 3 for a study on CALLISTO context factors).

tion. Most of these techniques are still in wide use in today's sentence extraction systems, although they have been refined.

- [Kupiec *et al.*, 1995] successfully used advanced Machine Learning techniques (more specifically Bayesian learning) for text summarization purposes. Again, the ideas developed in this paper have been widely re-used since then.

2.2.2 New Methods in Automatic Abstracting (H.P. Edmundson, 1969)

Edmundson's study is, first of all, interesting because it is the first one to identify several features characterizing "extract-worthy" sentences. Before him, Luhn [Luhn, 1958] had created an automatic system performing sentence extraction but it was only based on the frequencies of words in the document. His assumption was that the sentences to extract were the ones containing the most relevant words, relevant being here defined by a range of frequencies[7].
Based on two corpora (one of general scientific articles and the other more specifically dedicated to chemistry), Edmundson keeps this technique called *key-words* in his paper but adds three more:

- The *title* feature consists of giving more weights to the sentences containing words used in the title or headers of the article. The assumption here is that the author of the paper has chosen titles and sub-titles representative of the major topics he deals with and that, therefore, explanations or developments on these terms have to be looked for inside the document.

- The *cue-word* feature establishes two lists of fixed expressions[8]: the bonus words and the stigma words. On the one hand, the former (e.g. "Important", "Significant" or "Finally") are a hint that the sentence could be relevant to extract and must therefore be given a higher weight. The latter, on the other hand, (e.g. "Insignificant", "Detail" or "Parenthetical") rather indicate that the sentence should *not* be selected and therefore that its weight should be decreased.

[7]The most frequent words were excluded as being stop-words, which is mostly done nowadays by a pre-processing filter, and the most rare were discarded as not representative of the article.

[8]Actually four lists, because there were two more. One, called the null list, contained neutral cue-words. This list seems totally useless today compared to an adequate stop-word list. The other, called the residue list, was for rare words.

- Finally, the *location* feature rewarded sentences found at specific positions in the text where there usually are extract-worthy sentences (e.g. after the headings "Introduction" or "Conclusion", at the beginning or the end of first and last paragraphs).

It is interesting to note, as emphasized by Mani [Mani, 2001a] that those features are used, at least partially, by professional human abstractors, to skim a text for significant and condensed information.
The evaluation method used by Edmundson is also worth mentioning. He divided his corpora in two sets, one was used to set the values of the different parameters (in particular the lists of words) and the other to evaluate the performance of different linear combinations of the features. In that part, all the sentences had been tagged and Edmundson could therefore compare the predictions of the system with this manually created set of extract-worthy sentences. In addition, eventually, the extracts produced were read and commented by human judges to point out their strengths and weaknesses. The result that the best method is a combination of title/cue-word/location without key-words is not very important[9]. The methodology however is remarkable and can be viewed, especially with the use of training and test data, as a first attempt to apply Machine Learning techniques to Automatic Text Summarization. In practice though, Edmundson did not have at hand all the Machine Learning tools available today. As a matter of fact, the paper [Kupiec *et al.*, 1995] by Kupiec, Pedersen and Chen, presented in the next section, shows how such techniques can be successfully combined with the features Edmundson used.

2.2.3 A Trainable Document Summarizer (J. Kupiec, J. Pedersen and F. Chen, 1995)

The methodology presented in this article is somewhat similar to the one adopted by Edmundson. They identified the following list of features:

- *Paragraph*: Where is the sentence inside the paragraph, beginning, middle or end?
- *Fixed-phrase*: A list of cue-phrases has been established.
- *Length Cut-off*: Very short sentences are discarded.

[9] It may be indeed highly domain-dependent.

- *Thematic Word*: Does the sentence contain some of the most frequent *content words*[10]?
- *Capitalized Word*: Does the sentence contain any proper name?

Now, to discriminate between these features, they use a Bayes Classifier:

$$P(s \in S|F_1, F_2, \ldots, F_k) = \frac{\Pi_{j=1}^{k} P(F_j|s \in S)P(s \in S)}{\Pi_{j=1}^{k} P(F_j)}$$

$P(s \in S)$ is the probability for the sentence s to be in the summary S and it is the inverse of the total number of sentences, hence a constant independent from s. $F_1, F_2, \ldots, F_k$ are the different features, $P(F_j|s \in S)$ and $P(F_j)$ can be evaluated on a corpus. Kupiec, Pedersen and Chen had a corpus of scientific articles with human-written abstracts.
To evaluate the different versions of their system (exactly the same way as Edmundson did), they aligned the human-written abstracts with extracts, identifying, first with a program and then manually, if a sentence from the abstract was similar to a sentence from the text, or resulting from joining two sentences, or being part of one sentence and so on.
Thus the classifier is trained and tested on the corpus by cross-validation and this Machine Learning methodology was subsequently used in many other systems. As we will show in detail in Chapter 3, CALLISTO has indeed a Machine Learning component and follows the same global sentence extraction methodology, giving relevance weights to sentences. However, it attempts to extend the choices of key-words selection using more advanced techniques and deals with location issues thanks to segmentation algorithms. The basic ideas underlying the system are therefore not fundamentally different from the ones used by Edmundson, Kupiec, Pedersen and Chen but CALLISTO takes advantage of up-to-date tools to perform the basic steps.

2.2.4 Revision

For the sake of completeness, we now dedicate a brief section to *revision* operations. They are important in Sentence Extraction because they can bring a solution, at least partial, to the problem of dangling references evoked earlier. The principle is to apply deeper syntactic or semantic knowledge than the one generally involved in extraction to identify redundancies or incoherence in the extract. An obvious and easy operation is to suppress

[10]We call *content words* the meaningful words from the text. Practically, a list of stop words is used to discard less meaningful terms, such as articles or prepositions.

some short connecting phrases like "for example", "hence" or "therefore" which guarantee discourse cohesion in the full text but are likely to be totally useless, or even misleading, when the sentence is taken out of its context. In [Mani *et al.*, 1999], Revision is presented as a complementary module to apply to the output of an Automatic Summarization System, typically a sentence extractor. This paper proposes several heuristics to improve the quality of the final summary such as:

- Elimination: Parenthetical material is removed, as well as the ambiguous connectives discussed above.

- Aggregation: This kind of operations applies to two parsed sentences. Coreferential noun phrases are identified and operations are carried out on the parse trees to combine the information extracted from both sentences in one. An example of such a revision would be changing "CALLISTO is a sentence extractor. It has a Machine Learning component." to "CALLISTO is a sentence extractor which has a Machine Learning component."

- Smoothing: The sentences created by aggregation are often improvable and smoothing consists of applying simple rules such as deletion of "which is" or replacement of "which have" by "with". In the previous example, the sentence would become "CALLISTO is a sentence extractor with a Machine Learning component.". Some dangling references are also solved at this point, replacing definite pronouns with the corresponding coreferential proper name or definite noun phrases with indefinite ones when appropriate. For readability, the converse operation can be conducted as well, that is introducing a pronoun if the same reference is mentioned in two consecutive sentences.

Another strategy is to avoid selecting sentences with anaphora altogether [Paice, 1990]. In any event, revision involves deep semantic operations[11]. These are often costly but solve some of the issues brought by the sentence extraction methodology. The question is to know whether or not raw extracts are satisfactory for the application and if not, investigating revision algorithms may be helpful. CALLISTO, however, does not have a Revision module and the aim of the thesis is to improve it as a sentence extractor. We will consequently not focus more on such post-extraction operations.

[11] For instance, [Mani *et al.*, 1999] had a parser, a name tagger and a coreference algorithm.

2.3 Evaluation

2.3.1 Evaluation in Natural Language Processing

Evaluation is considered a difficult task in many areas of Natural Language Processing. Whether the focus is on summarization, translation or generation, measuring how well a given system performs is highly subjective and therefore challenging in its own right. Researchers in automatic summarization have used several quite different techniques with various methodologies, costs and degrees of reliability.
In a paper proposing a theoretical model for evaluation, Popescu-Belis [Popescu-Belis, 1999] draws a parallel between evaluation in NLP and Software Engineering Principles. On page 2, he recalls that "the programming cycle is traditionally divided in three stages: specification, realization and verification & validation", the last stage being the closest to what we informally called evaluation. Strictly speaking, verification consists of checking that some piece of software performs accordingly to its specifications. However, in many areas of NLP, including Automatic Summarization, the specifications are rather loosely defined and Popescu-Belis explains it on page 3 saying that "there are no formal methods to define correctness". That is why he concludes that "traditional" evaluation in NLP is rather "performance checking".
Also borrowed from Software Engineering vocabulary, evaluation can be conducted in several different modes:

- The *black-box* mode means checking only the inputs and outputs of a system to evaluate if the requirements are met. This mode has been widely preferred so far in NLP evaluation, in particular for competition-like events.

- The *glass-box* mode is more analytical and consists of examining the structure of the program itself.

- Popescu-Belis adds another mode, close to the black-box mode but not exactly identical and particularly relevant to NLP: *user satisfaction.*

In their very detailed study of evaluation in NLP, Galliers and Sparck Jones (1995) identify a major criterion to distinguish between evaluation methods.

- *Intrinsic* evaluation methods focus on the system's main objective when defined.

- *Extrinsic* evaluation methods suggest that the system be judged on other tasks related to its function and for which it is easier to set up an evaluation framework.

We will discuss this criterion in detail in the next section, but, first, let us give a brief summary of the overview by Galliers and Sparck Jones of evaluation methods adopted so far in NLP.

- Machine Translation: linguistic models are applied, working in black-box mode on prototypes.
  Error analysis can be determined in terms of the number of words to modify to get a proper translation or, even more subjectively, a human evaluation of "fidelity" with respect to information content.
  Some examples of Machine Translation Systems for which interesting evaluations were conducted are ALPAC (1966), TAUM-AVIATION (1980), SYSTRAN (1991) (see [Falkedal, 1994] for more detail about those three). JEIDA [Nagao, 1989] and JTEC [JTEC, 1989] are two state-of-the-art reports in Machine Translation with a few paragraphs about evaluation. More recently, BLEU [Papieni *et al.*, 2001] is an automatic evaluation method that we will study in Chapter 4.

- Message Understanding: the Message Understanding Conferences (MUC) [Grishman and Sundheim, 1996] were among the most advanced and useful projects in NLP Evaluation. Black-box testing was conducted over a corpus of short articles, for which the participants were asked to fill a template answering the most basic questions regarding the information delivered (such as who, when, where, what?). The results were calculated in terms of precision and recall.

- Database query: LUNAR [Woods, 1973] and TQA [Damerau, 1979] are examples of database query systems which were evaluated by submitting to them several questions from their supposed final users, falling in the range of what they are supposed to answer. Then NLS [Jarke *et al.*, 1985] was evaluated with the objective to see if it was worth replacing the language SQL by this system. The results computed were the number of answers relevant to the queries and the time needed by the system to answer. Finally, ATIS [Boisen and Bates, 1992] is an annotated database which provides a good framework for Black-Box testing in Database query. It is still in wide use in more recent research as [Mooney, 1997] shows. Furthermore, this paper, which presents CHILL, a system for Natural Language Understanding, uses another benchmark, specific to inductive logic programming.

It consists of geography questions presented both in Prolog and Natural Language forms.

- Text Summarization: in 1979 and 1980, the FRUMP [DeJong, 1982] system was connected to the United Press International News stories and asked to output summaries of every incoming news item. The results were then classified as correct, partially correct or wrong by human judges.
  More recently, TIPSTER [TIPSTER, 1998] and DUC [DUC, 2001 and 2002] ranked a set of automatic summarization systems based, again, on human judges' evaluation.

2.3.2 Evaluation in Automatic Summarization

Evaluation in Automatic Summarization presents the same characteristics as Evaluation in NLP in general and has been a very active research theme in the last ten years.
Faced with the difficulty of evaluating the quality of a summary in the abstract, one may look at the alternative approach: determine the suitability of a summary for a particular task. [Galliers and Sparck Jones, 1995] distinguishes between *extrinsic* and *intrinsic* methods. The former test a summarizer in relation to some task, whereas the latter are intended to give an absolute measure of the quality of a system's production.

2.3.3 Extrinsic Methods

Extrinsic methods include *relevance assessment* and *reading comprehension.* In both cases human judges work on the output of the system. This view is close to asking feedback from the end users, except that, in order to get a more accurate and reliable measurement, those final users are sometimes professional summarizers or journalists.
They may be asked to determine if a document summary correctly identifies its relevance to a search on a given topic. For example, in [TIPSTER, 1998], there were 5 categories and, given a document (which could be either the source, a human-produced abstract or a machine-produced summary), judges were asked to determine which of them the document belonged to. As pointed out by Mani [Mani, 2001b], the goal of the study is to determine whether the summaries are informative enough to save time in this categorization activity. The paper shows the usefulness of summarization for this task since the judges categorized the documents much faster with the summaries than with the whole texts and, in general, as accurately.

In [Morris *et al.*, 1992], which draws an interesting parallel between Automatic Summarization and Information Theory, the judges were asked to answer questions about a text's topic using only its condensed version. One can argue that this kind of evaluation is always biased toward a single task. As a result, evaluation might sometimes reward poor, or at least poorly readable, summaries that happen to contain answers to specific questions. However, the fact that the resources to apply these methods are numerous and easily available, like the Test Of English as a Foreign Language (TOEFL) or the Graduate Record Examinations (GRE), makes them very attractive altogether.

2.3.4 Intrinsic Methods

Mani [Mani, 2001b] emphasizes the difference between evaluating *quality*, how well a summary is written, and *informativeness*, how much knowledge it contains.

Quality

Readability is a highly subjective criterion whose measurement is difficult to automate. Mani gives the example of automatic style checkers that rely on very simple statistical considerations such as average lengths of words and sentences. A more reliable analysis should undoubtedly involve human judges rating how well a summary reads.
This point is less important in Sentence Extraction. Our policy of building a summary by concatenating sentences extracted from the full text allows us to be a little less concerned about this issue; we take advantage of the original author's writing skills. This does not mean that we avoid problems of dangling references and irrelevant logical links, but these can be addressed later as described earlier in the Revision section.

Informativeness

Some evaluation methods compare the information contained in the abstract with that in the full document. This is feasible when one has a way of measuring informativeness, but will, in most cases, involve human judges at some point. It is a separate and non-trivial task to discern the most important sentences in the original text (that is, to recognize varying relevance) which may well be as difficult as the task of summarizing itself[12]. For instance, in

[12]We can even say that it is almost summarizing since it is the main focus of Sentence Extraction.

[Minel *et al.*, 1997], subjects have their attention drawn to the number of rhetorical links in the initial text covered by the summary.
But again, this kind of method is very expensive and cannot be conducted as much as needed. On the contrary, it is generally easier to find datasets with human reference summaries. Although human judges have compared generated summaries against an ideal abstract with success [DUC, 2001 and 2002], there are also a wide range of automatic methods [Donaway *et al.*, 2000] that compute quality measures at the sentence or word level.

- A good measure of the performance of a sentence extraction system is *recall* over the best sentences: the number of sentences from the reference extract that occur in the summary, divided by the total number of sentences in the reference extract[13]. Note that the reference summary may rather be an abstract than an extract. It is still possible to use this measure but a pre-processing phase to find the extracts which match best the ideal abstracts is needed. See the paper already studied [Kupiec *et al.*, 1995] for an idea of such a process and [Banko *et al.*, 1999] for a more detailed example.

- The alternative is a content-based measure which tries to compute how close the summary is to the model. None of the several existing methods is an entirely satisfactory way of comparing two summaries of a given text. Improving on their performance would require readability evaluation and deep semantic techniques. Mani [Mani, 2001a] suggests that such a method is adequate for sentence extracting techniques applied to news reports when the reference abstracts include much text taken unchanged from the source[14]. A good suggested extract will have a lot in common with the model abstract as long as it contains the most relevant sentences. For example, synonymy needs not be the problem that it would be if we were working with more completely rewritten reference abstracts. In the final analysis, almost as with sentence recall, a summary will score well if and only if the best sentences are extracted from the source. Note, however, that an important issue with using such a measure is to prove its reliability,

[13] A more complex variant would give weights to sentences in the document, and use these weights to compute the score of a summary.

[14] In news report summarization, satisfactory abstracts can often be obtained by selecting the few most relevant sentences, or pieces of sentences, and pasting them together to form a coherent summary (and it is the case of the abstracts used in DUC [DUC, 2001 and 2002]). The reference built is very close to an extract but is strictly speaking an abstract, according to the definition we gave in chapter 1.

that is: how can we make sure that the scores we get are indeed representative of the *quality* of the summary? One solution, as we will see in Chapter 4 is to establish that it is correlated to human judgment.

To conclude, Evaluation in Automatic Summarization is a very promising area. In particular, some wide-scale projects [Baldwin *et al.*, 2000] are conducted and could, in the next few years, bring interesting frameworks, profitable to all researchers working in the field. In the meantime, we will show in Chapter 4 how some of the measures mentioned combined with resources available from DUC [DUC, 2001 and 2002] enabled us to create a framework to evaluate and eventually improve CALLISTO.

Chapter 3

CALLISTO

An Automatic Text Summarization System

3.1 Introduction

The CALLISTO Summarization System was designed in 1999 at the University of Ottawa by Terry Copeck, Nathalie Japkowicz and Stan Szpakowicz [Copeck *et al.*, 2002]. We present their work in this chapter and describe the system as it was in 2001, as a starting point of this thesis. The basic idea of the system is that the very difficult task of text summarization can be achieved through simpler Natural Language Processing techniques, such as Key Phrase extraction and Segmentation. Those can be performed by freely available tools, found on the Internet or in the research community in NLP. The aim of CALLISTO is therefore to find the best way to combine them in order to summarize texts automatically. To achieve this goal, CALLISTO has a Machine Learning component, currently C5.0 [Quinlan, 1997], which enables us to identify the best configuration of the system for each text.
To begin with, let us discuss the context factors defined by Sparck Jones in [Sparck Jones, 1998] and propose a broad definition of CALLISTO by instantiating these factors.

- Input form: So far CALLISTO, like most Automatic Summarization

Systems has been evaluated only on *short news stories in English.* A priori, it could be employed for any other kind of documents but there are a few obvious limits. The methodology could be exported to other languages but the system as such cannot since the segmenters and key phrase extractors are English-specific. Another problem is inherited from text extraction in general : summarizing documents of more than a few dozens of pages to an abstract of reasonable size involves summarizing at very low compression rates[1]. As stated by Mani [Mani, 2001a], this requires reformulation and deep semantic understanding of the topic, which is not what CALLISTO can offer. However, as long as this length constraint is satisfied, CALLISTO might be efficient whatever genre we consider, although, again, this has not been verified practically.

- Input subject type: The subject type is *ordinary*, no specific background knowledge is assumed, neither from the readers of the source articles, nor for the potential readers of the summaries.

- Input unit: CALLISTO is primarily intended to summarize *single* documents. The methodology can be applied to clusters of texts but it is not clear how sentence extraction is suitable for this task.

- Situation: The situation in which the summaries produced by CALLISTO could be used is not known a priori so it is described by Sparck Jones as "*floating*".

- Audience: This is related to the subject type above, the audience is *untargeted.*

- Use: Again, as CALLISTO is a research project, specific applications are not defined yet. However, if we had to formulate a broad guess, we could say that the main use of CALLISTO would be to give an idea of the source to a potential reader without having to go over the whole article. Ideally all the relevant information is in the summary and we do not need the source anymore for most applications. In practice, this goal is probably impossible to achieve and getting *an output semi-indicative and semi-informative* for every input would already be an achievement.

[1]For a small book of about a hundred pages for example, a 25% summary would be 25 pages which is likely to be too long for most applications. A more acceptable size of 1 or 2 pages represents a 1 or 2 % compression rate which may be impossible to achieve with sentence extraction.

- Output material: CALLISTO is not selecting any particular information from the source but attempts to output a *covering* summary.
- Output format: Being a sentence extraction system, CALLISTO delivers *running texts.*
- Output style: As said before, we aim for *informative* summaries. Right now though, many of the extracts produced by the system are not, the majority being *indicative* only.

3.2 Segmentation and key phrase extraction

A segmenter is a tool which takes a text as input and breaks it into a semantically coherent group of sentences. There are several ways to do it, and some are studied in the next section, throughout the presentations of the tools we are using. The basic idea is to identify shifts in (lexically expressed) topics. A simple way to segment, relying entirely on the author's organization is to consider the text paragraph by paragraph, assuming that every paragraph deals with a different topic. However, this is generally not true and, even if it were, one may sometimes want to group two adjacent paragraphs together arguing that, even though they are not tackling exactly the same subjects, they are semantically closer to one another than they are to the rest of the text. Therefore, more intelligent methods than paragraph segmentation are generally needed.
A key phrase extractor analyzes a text to retrieve the most relevant words to characterize it. Ideally, the output of such a tool is equivalent to the list of key-words authors of scientific articles sometimes provide with their papers. The simplest way to do it, following Luhn [Luhn, 1958], is to select the most frequent words in the text, once stop-words have been removed. The assumption is that when a term is important, the author emphasizes it and explains it at length throughout the paper. But there are many methods to refine that and we will review some of them.
But, first, let us present the algorithm behind the summary generation component of CALLISTO.

1. We apply a segmenter to the input text. The source is then divided into segments.
2. Now, from a key phrase extractor, we get a set of most relevant phrases.
3. The segments can thus be ordered by decreasing order of key phrase density.

4. For each segment, all the sentences are in turn ordered by key phrase density.

5. From the first segment, we collect all sentences containing more than *hitcount* key phrases. When there are no more of those in the first segment, we repeat the process for the second one and so on, until we have *sentcount* sentences.

Parameters of the system:

- *hitcount*: minimal number of key phrases a sentence must contain to be in the summary.
- *sentcount*: number of sentences to extract.

Figure 3.1 summarizes this process.

The main assumption under this algorithm is that the density of key phrases is a measure of the "informativeness" of the segments. The more key phrases a segment contains, the more representative it is of the text. As they are supposed to form semantically coherent units, they can be viewed as a sequence of document fragments on certain topics dealt with in the text. Getting a summary from this list of segments ordered from the most important to the least important (by decreasing key phrase density) just consists in choosing which kind of units to extract. It would make sense to extract whole segments but, as emphasized in chapter 2, this method does not allow us to achieve the desired compression rate appropriately. Hence the idea is to extract sentences from the most informative segments. How many will we extract from each segment? There is no straight-forward answer. The algorithm implemented in CALLISTO uses key phrase density again. It assumes that the sentences from a relevant segment are worth extracting as long as they contain more than *hitcount* key phrases, *hitcount* being a parameter of the system.
As we said before, we use several segmenters and key phrase extractors. Before going on to explain how we choose between those, let us present them briefly. We will also discuss in detail the different options in the system.

3.3 The modules and parameters of the system

3.3.1 The segmenters

The two main techniques used in Segmentation are *lexical cohesion methods* (identifying segment boundaries as shifts in the vocabulary used) and *multi-*

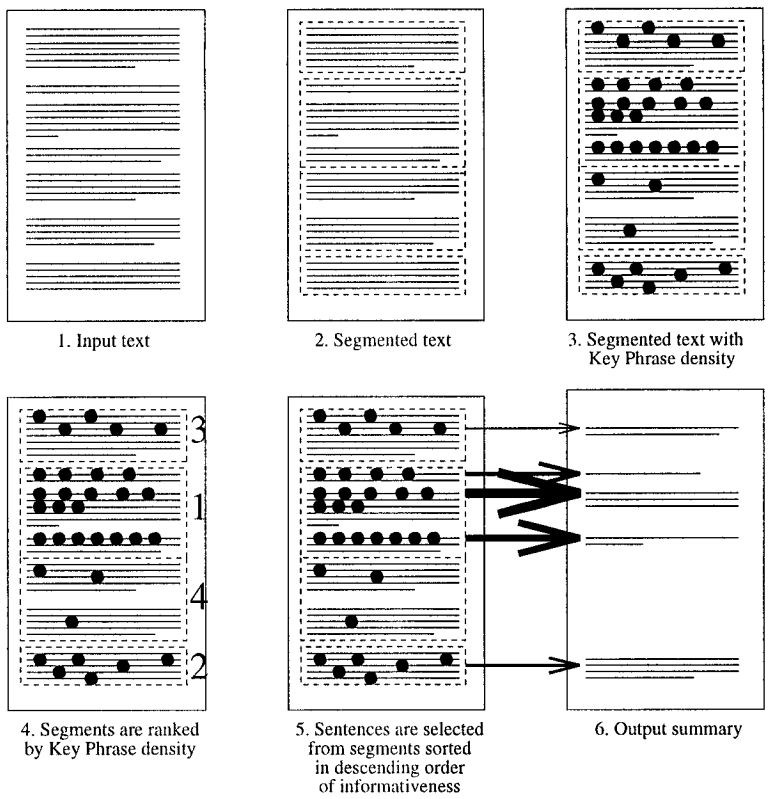

Figure 3.1: The algorithm for sentence extraction by segmentation and key phrase extraction.

source methods (adding other techniques, such as cue phrase detection or machine learning) [Choi, 2000]. The three tools used in CALLISTO all exploit lexical cohesion. An alternative to these tools, implemented in the system, is to skip the segmentation step altogether.

- TextTiling: The algorithm is described in [Hearst, 1994]. The principle is to compare adjacent blocks of texts (not necessarily paragraphs or sentences but sequences of a certain number of words) for lexical similarities. Hearst assumes that the more terms are repeated from one block to another, the less likely is a semantic shift at that point. This "shift probability" is computed for every word in the text, evaluating the similarity between the n words before and the n words after, n being a parameter to be determined by experiments (Hearst reports that a value of 120 worked well for her corpus). Thus she gets a curve

representing the evolution of block similarity throughout the text. The next step is to map the actual paragraph breaks to the local extrema of the curve. Adding a few ad-hoc heuristics to correct some other problems such as short paragraphs, the segmentation of the text is determined by that combination.

- Segmenter: The leading idea of Segmenter is the same as in TextTiling, i.e. the shifts are identified according to lexical similarity between consecutive paragraphs. But the method, described in [Kan *et al.*, 1998], is slightly different. The types of words considered are restricted to proper and common noun phrases, personal and possessive pronouns. Similar phrases constitute a chain if they are sufficiently close and if they are either identical or have the same head noun[2]. For every chain, *front*, *rear* and *during* paragraphs are identified and assigned scores accordingly[3]. Those scores are summed over all terms considered and the semantic boundaries are associated with the paragraph breaks having the highest scores. The evaluation of the method presented by Kan et al. shows better results than TextTiling but, since it was conducted over very few texts (20), the relevance of this experiment can be questioned.

- C99: C99 [Choi, 2000] uses the cosine similarity measure between sentences (after filtering stop words and stemming) to see how close any two sentences of the text are to one another. These results are then entered into a matrix and converted into local ranking (arguing that for short texts, absolute values of that measure are not meaningful), which improves "contrast" and points out which pairs of sentences are closest. Eventually clustering is achieved by successive splitting operations, the text being considered as one segment for initialization, in order to maximize the sum of rank values, that is the sum of similarities within each clusters. The evaluation method consisting of concatenating various sequences of paragraphs from different news stories seems reasonable. On the experiments conducted over these concatenations of texts, C99 consistently outperforms TextTiling and Segmenter.

- None: An additional option in CALLISTO is not to segment the text.

[2]The case where the head nouns are identical but the adjectives are different, e.g. "white wine" and "red wine" is dealt with separately.

[3]For noun phrases for instance, *front* and *rear* paragraphs are given a high score, indicating that the beginning or end of the chain is likely to indicate a topic shift.

It comes down to selecting the sentences containing the highest number of key phrases, wherever they are in the text, and is equivalent to earlier approaches in text summarization. From an experimental point of view, this option will also enable us to check whether the initial assumption that the text should be split by topic before extracting sentences is valid.

3.3.2 The key phrase extractors

- Extractor: The paper [Turney, 2000] is a detailed account of several experiments conducted by Turney to apply successfully Machine Learning techniques to Key Phrase Extraction. With a lot of interesting results reported and a suggested algorithm, it is an excellent state-of-the-art report for this task. The first part of the article studies how C4.5 with various options can be used for Key Phrase Extraction. The second part presents the algorithm leading to the program Extractor used in CALLISTO. We will not present in detail all the steps here but the basic ideas are to stem the words on a fixed length, extract the most frequent content words and then the most frequent content phrases[4]. The content phrases are used to expand the list of words, which serves as the basis for ranking the output key phrases after several post-processing steps (adding suffixes, capitalization, and so on). This algorithm has several ad-hoc heuristics using parameters to be determined by Machine Learning. A genetic algorithm does this and the final results are that Extractor performs better than the algorithms using C4.5 with optimal parameters.

- Kea: Based on Turney's article, [Frank *et al.*, 1999] apply Naïve Bayes[5] to keyphrase extraction using only two parameters: the frequency score output by the TF.IDF technique[6] and the position inside the document (in percentage of the length). The class is positive if the phrase considered is a key phrase, negative otherwise (binary learning problem). After training the classifier on about 50 documents, the authors discovered that its performance was equivalent to that of Extractor.

[4]Content phrases are defined as the groups of consecutive content words, without stop words or punctuation between them. See Section 3.4.2 for further details.

[5]The principle of Bayes classifiers has been briefly presented in Chapter 2 for the study of [Kupiec *et al.*, 1995]. See Chapter 6 for more details.

[6]The idea of TF.IDF is to divide the term frequency by the number of documents in which the term appears to reward terms making a document unique in a corpus.

- NP Seeker: [Barker and Cornacchia, 2000] presents a system focusing on noun phrases for key phrase extraction. No Machine Learning is involved. Noun phrases are first identified in the document with the help of dictionaries and the ones with the most frequent base nouns are selected as candidate key phrases. A few rules regarding the number of words in the phrase, or the cases of whole inclusion of one candidate into another one, are then applied to the list to get the final output of the system. The experiments presented in the paper compare NP Seeker and Extractor and find that the latter tends to extract more relevant individual key phrases but that the list of the former were more coherent on the whole. But measures of agreement such as kappa suggest that the different judges do not agree very well with one another and it is therefore difficult to draw definite conclusions about the results.

3.3.3 Matching

As we will have to compare key phrases to words in the text (to count the number of occurrences), we have to decide if we look for an exact match or a match between the stems of the phrases. Stemming makes sense for some cases, for instance so that the singular and plural of the same noun are not seen as two different words, but it is not computationally negligible and may not make that much of a difference in the end.
Therefore both decisions can be seen as relevant and this is one more parameter for the system:

- Exact matching: Two words are matched if and only if all their letters are identical.

- Stem matching: Kea proposes a module which operate stemming thanks to the iterated Lovins stemmer (described for example in [Turney, 2000]). As explained in [Copeck *et al.*, 2002], it was repackaged to be used as a standalone program.

Note however that in CALLISTO as it was submitted to DUC 2002 [DUC, 2001 and 2002], stemming was not used under the assumption that the computation time was not worth the improvement in quality[7].

[7]During the preparation for DUC 2001, it was noted that, whether stemming or not, very similar sets of matches were obtained. Since stemming is moderately computationally intensive, and here was motivated solely in the expectation that it would increase the set of matches, failure to achieve the increase is sufficient grounds to eliminate using

3.3.4 Number of sentences

One of the parameters *sentcount* is the number of sentences to extract. So far, the different values of this parameter for the training data for DUC were 3/4/5/8/12. When the input text is shorter than *sentcount*, the whole text is output as a summary.

3.3.5 Number of key phrases

Every key phrase extractor used has as input parameter the number of key phrases to be extracted from the text so this becomes a parameter *keycount* for CALLISTO. For the training data for DUC, *keycount* was set to 3/4/5/8/12. The key phrase extractor considered may output fewer key phrases if *keycount* is greater than the maximum number of key phrases that the system finds for the given text.

3.3.6 Minimum number of hits in a segment

We have already described *hitcount* in the previous section. It specifies the minimum number of key phrases in a sentence to be extracted. If no more sentences contain more than *hitcount* key phrases in the given segment, another segment is considered. In DUC 2002, a constant *hitcount* set to 2 seems to bring good results. However, it would be interesting to test the values 1 and 3 as well[8]. Note that there is a relationship between *hitcount* and *keycount*. If very few key phrases are extracted then *hitcount* has to be adjusted to a low value. On the contrary, if the value of *keycount* is high, it makes more sense to have *hitcount* similarly high.

3.4 Key Phrases in Abstract and Machine Learning

3.4.1 Decision tree induction

While the steps described in the previous section always produce a summary no matter which segmenter and key phrase extractor are chosen, we do not know whether the summary is appropriate. We are in a situation where the dependencies between the input parameters and the final quality obtained

the procedure. The evaluation framework we present in this thesis could confirm this hypothesis.

[8] A value greater than 3 would allow very few sentences to be extracted.

cannot be determined by a human because there are too many summaries involved. So it seems reasonable to apply a Machine Learning algorithm, to find the best combination of segmenter and key phrase extractor for every text. We have chosen C5.0 [Quinlan, 1997] because it is fast and has proven to work well in a large number of applications. However, we have no reason a priori to prefer it to other algorithms so we will also consider other learners and see what the results are in Chapter 6.
Now, to have C5.0 learn something, we need to give it labeled training examples as in any supervised learning framework. These training examples will consist of a *text vector* that represents the input document, the configuration tested[9] and the result obtained in term of the quality of the summaries. Thus we need a method to evaluate the quality of the summaries produced. It is time-consuming but feasible to apply each summarizer configuration to all documents in the training set. It is, however, not realistic to ask a judge to rate all the resulting summaries[10]. The application of supervised learning to text summarization is paradoxical—never has the need for quick and reliable evaluation methods been so pressing, yet the only reasonably trustworthy methods require human judgment at some point. We currently address this issue by having the CALLISTO system rate each summary it produces by computing a score based on Key Phrases in Abstract (KPiAs). We will first enumerate the features in the text vector and then explain the evaluation method in detail.

3.4.2 The text vector

For every document, we calculate a set of features which represents the texts. This text vector will be described in longer terms in Chapter 7, which deals with Feature Selection but the attributes used for DUC 2002 are briefly described in table 3.1, reproduced from [Copeck *et al.*, 2002].
A few ideas exposed in this table need to be clarified:

- The character, word, sentence and paragraph counts are representative of the length of the text and of its organization. Thus they could be important attributes to characterize the document.

- The connectives are some prepositions which indicate a rhetorical shift as described in [Marcu, 1999]. They are clues regarding the logical

[9]The six parameters described in the previous section also constitute a vector of attributes that we will now refer to as the *system vector*.

[10]Our training data, obtained courtesy of the Document Understanding Conference, contains more than 1100 texts; the total number of summaries generated exceeds 300,000.

evolution of the argumentation. "Also", "but", "wherever" are a few examples of elements on this list. The intuition behind this count is that the number of such clues could be related to the complexity of the text and therefore to how difficult it is to summarize.

- The number of proper nouns and acronyms are the number of *entities* (person, place, organization) referred to in the text. It is likely that the number of entities matters regarding the way to summarize. For example, if we have very few entities in the text, they may all be present in the summary and the summarization effort could bear on the actions they perform. On the contrary, if there are many entities, we will have to get rid of some of them to reduce the document.

- For the number of content phrases, we count the number of phrases once the stop words have been discarded. For example, the sentence "CALLISTO is a Text Summarization System with a learning component" contains the stop words "is", "a", "with" and "a". Therefore, there are three content phrases: "CALLISTO", "Text Summarization System" and "learning component". The assumption here is that the number of content phrases is close to the number of concepts or ideas referred to in the text. Although two different content phrases could refer to the same person, place, action..., we suppose that the more content phrases a text contains, the more complex it is and therefore the more ideas have to be conveyed in the summary.

- For the last 12 attributes (that is x-instance content phrases/content bigrams or bigrams), we make a distinction according to the number of instances, that is, the number of occurrences of the same phrase in the text. Thus, if we consider content phrases, a phrase which occur only once in the text will be counted in cphr, twice cphr2 and so on. Therefore the following relation holds:

$$contcnt = cphr + cphr2 + cphr3 + cphr4$$

We count the number of instances with the idea that a text containing few content phrases repeated many times is very different from a text containing a lot of unique content phrases, even though their number of content phrases is close or identical. Based on the same idea as for the proper noun and acronym count, two such documents may need to be summarized differently.

- Finally, we consider bigrams, that is to say phrases of two words. Content bigrams are the ones containing only content words. In our previous examples, there are four content bigrams: "Text Summarization", "Summarization System" and "learning component". We count content bigrams because they offer a more fine-grained description than content phrases. Following the same example, "Text Summarization" and "Summarization System" are two different concepts whereas they are covered by a unique content phrase "Text Summarization System". Thus the number of content bigrams could be a useful feature to characterize the document.

- For the non-content bigrams, or, simply, bigrams, stop words are not excluded and, therefore, we may get more irrelevant counts but we hope to get an insight about the style of the author counting the number of x-instance bigrams. Similarly to the x-instance content phrases, a document where the same bigrams are repeated several times may not be summarized the same way as one where there are many unique bigrams.

These properties have been chosen following the intuitive ideas described above and because they are easy and fast to compute. We assume that, along with the system vector, they will provide the learning program C5.0 with enough knowledge about the texts to find patterns in the training dataset. In chapter 7, we will investigate other possible attributes for the text vector and see if we can apply feature selection algorithms to select only the counts most relevant for our task.

3.4.3 The Key Phrase in Abstract Evaluation Method

The Key Phrase in Abstract (KPiA) method requires a model abstract for each text in the training data. We remove stop words from the model and count occurrences in the summary of the sequence of content words (called key phrases, even though they may not be real phrases) between them. The formula below favours the first occurrence of a KPiA:

$$Score = \sum 1.0 * KPiA_{unique} + \sum 0.5 * KPiA_{duplicates}$$

We then compute the *nearly best* score from the text: we extract the n sentences with the highest density of KPiAs, if n is the desired length of the summary. Placed in document order, those sentences are taken as a model extract. We rate this model using the formula above. A flaw of this

Features	Number of	Features	Number of
chars	characters	cphr4	four-or-more instance content phrases
words	words	cbig	one-instance content bigrams
sents	sentences	cbig2	two-instance content bigrams
paras	paragraphs	cbig3	three-instance content bigrams
conncnt	connectives	cbig4	four-or-more instance content bigrams
pncnt	proper nouns, acronyms	abig	one-instance bigrams
contcnt	content phrases	abig2	two-instance bigrams
cphr	one-instance content phrases	abig3	three-instance bigrams
cphr2	two-instance content phrases	abig4	four-or-more instance bigrams
cphr3	three-instance content phrases		

Table 3.1: The different counts used for the Text Vector

procedure is that the reference may not achieve the highest score possible. Due to the bonus term in the formula, the sentences containing the largest number of KPiAs may not be the most informative[11]. This is why the model is labeled "nearly best". Extracted summaries are rated by dividing their score by that of the model. Because the model is only "nearly best", the quotient can be greater than one. When this happens we take the rating to be one. The full algorithm is presented in appendix A.
Also in appendix A, we study the complexity of this algorithm. We get an approximate complexity of $O(N * n)$ where N is the length of the original text and n the desired length of the resulting summary. Computing the score for the nearly best summary is the bottleneck in the process. The reason

[11]Consider an extreme example, where you would have the same KPiA ten times in three different sentences, and no other sentences contain more than ten KPiAs. This three-sentence extract would get a KPiA score of 15.5. Now, assume that there are two more sentences with only eight KPiAs but all different. This two-sentence extract would score 16 with one less sentence.

why we conducted this complexity study is because the KPiAs evaluation method is another choice we want to challenge. This evaluation method has the obvious advantages of being automatic, easily computed and reasonably fast. However, it is not entirely flawless as we will see in chapter 4 and we will discuss an alternative evaluation method.
This measure gives a continuous number between 0 and 1, while C5.0 classifies cases in terms of a discrete set of values, enumerated or computed. The continuous input must therefore be discretized before training can proceed. We can choose among several discretization methods, depending on the character of our data. We chose to employ the K-means procedure [Mac Queen, 1967]. On preliminary experiments, 5 classes seemed to give good results so the 5-means algorithm has been used for DUC 2001 and 2002. This discretization process will be studied in Chapter 5, to check if K-means was the best choice and what the optimal number of classes is, if any.
Figure 3.2 gives an overview of the CALLISTO system, with its learning component.

3.5 Does CALLISTO Produce Good Summaries?

Although C5.0 helps determine which segmenter and key phrase extractor are best suited to a given text, the way in which these tools are combined and data handled relies on our intuition. Our initial choices have been confirmed by the results achieved in the 2001 Document Understanding Conference since CALLISTO did reasonably well and was ranked among the best systems. In DUC 2002, CALLISTO did a bit worse, but still around the baseline. We now need a systematic way of evaluating the quality of the summaries produced. It would also be helpful to see how well CALLISTO performs when parameter values change. The arbitrary design choices we would like to question are:

- Evaluation measure (Chapter 5): Could we use other measures than KPiAs?
- Discretization process (Chapter 5): Are there other ways to discretize than K-means? Is there an optimal number of classes?
- Learning algorithm (Chapter 6): Are other algorithms than C5.0 usable[12]? If so, do they give good results?

[12]There is a large number of examples in our dataset and the number of attributes is not negligible either so we can only consider the fastest learners.

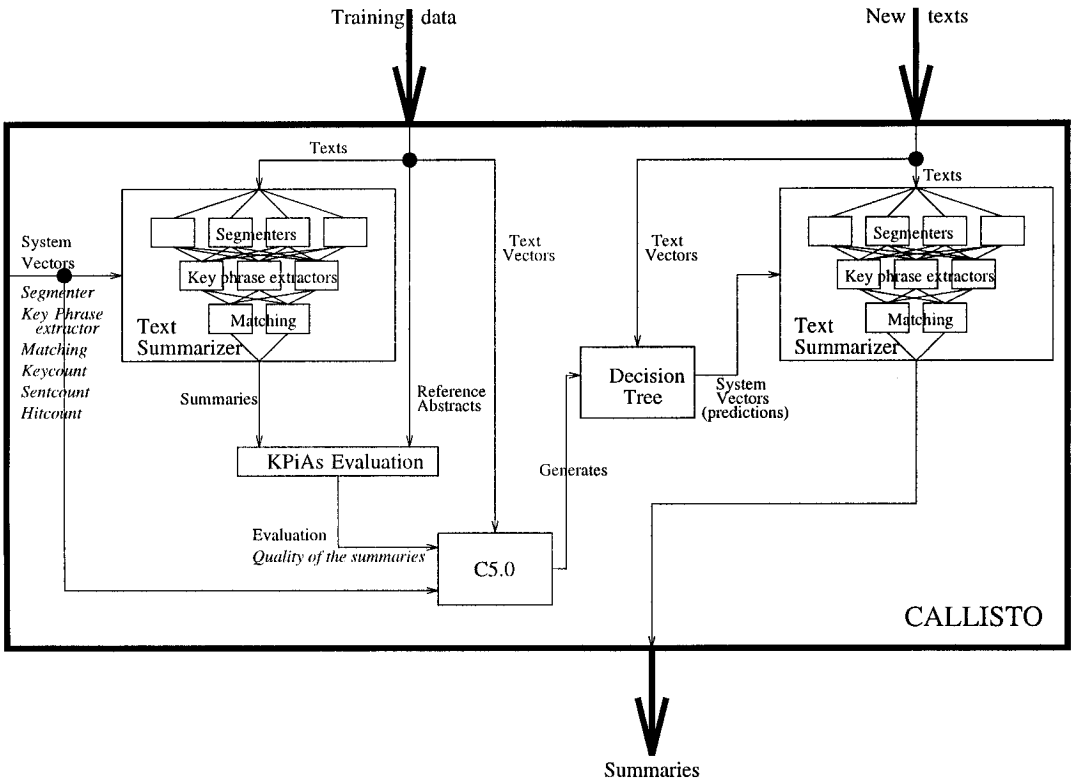

Figure 3.2: Overview of the CALLISTO System.

- Feature selection (Chapter 7): Which are the best attributes to characterize the text?
- Parameters of the system (chapter 7): What is the influence of the different segmenters and key phrase extractors used? Does the performance decrease when we try not to use some of them? Does the quality of the summaries improve if we replace the desired length of a summary by the compression rate?

As already noted, CALLISTO produces such volume of intermediate data that we can only succeed by building a fully automated evaluation method. We cannot afford human evaluation if we want to perform many experiments and evaluate hundreds of thousands of summaries. A fundamental characteristic of our algorithm should be that we can have confidence in its results as somehow representative of the quality of the summaries. It is clear that intuitions are unlikely to be sufficient. We will explain in the next chapter

what evidence we are seeking and the reason why we did not use the KPiA algorithm to measure the quality of the summaries produced. Eventually, we will present the automatic evaluation system we chose.

Chapter 4

Evaluating CALLISTO's performance

Our evaluation framework is based on a measure proposed by Lin and Hovy [Lin and Hovy, 2002]. This is a weighted average of similarity with a human written abstract, which consists of evaluating the overlap in groups of words (unigrams and bigrams) between both texts. We first present their method and explain that it is fully automatic, not relying on costly manual evaluation, but, at the same time, correlates well with human judges. As it was, to the best of our knowledge, the only measure respecting those criteria and as we had no resources to prove the correlation ourselves, we had no choice but selecting that one. Then we discuss its strength and limitations in more detail and describe how we built our framework based on this measure.

4.1 Lin and Hovy's Measure

This method is based on one that was originally developed by IBM [Papieni *et al.*, 2001] to evaluate document translation automatically. Let us first study this original paper.

4.1.1 BLEU: a Method for Automatic Evaluation of Machine Translation (Papieni et al., 2001)

One of the main problems in Evaluation for Automatic Translation is that several quite different translations can be regarded as equally good or acceptable[1]. Thus the problem is to cope with those numerous models and the principle "the closer a machine translation is to a professional human translation, the better it is".
The idea behind the method is to use *n-gram* precision. That is to say, given a set of reference translations and a candidate sentence, a measure of the quality is the number of common groups of consecutive n words (n-grams) between the sentence and one of the reference translations, out of the total number of n-grams in the sentence. But there are two main issues with such an evaluation.

- Translations much longer than the reference translation have to be penalized, even though they share many common words with the golden standards. Hence the authors propose a *modified unigram precision*, consisting of counting as wrong every instance of a word beyond the maximum number of instances of that word in a single reference sentence.

- In Information Retrieval, the Recall measure has been introduced to take into account the fact that the system must not only be accurate (precision) in its predictions but also has to retrieve as many documents as possible from the database[2]. Similarly, some sentences could be too short and yet get a high modified unigram precision if the words used are in the reference translation set. To address this issue, Papieni, Rouckos, Ward and Zhu assign a brevity penalty if the candidate translation has fewer words than its *best match* equivalent reference translation[3].

The method is then extended similarly to handle n-grams with $n > 1$ (in their experiments, they also considered bigrams, trigrams and quadrigrams).

[1] It is actually a problem which exists as well in Automatic Summarization as we will discuss it later. However, we will not address it directly in this work since in the dataset we have, there is only one reference summary per text.

[2] For example, a system which would retrieve only one document, which is relevant, out of a hundred in a database could be deemed as poor for many applications, even if it is true that it does not retrieve any irrelevant document.

[3] That is, for every candidate sentence, the one from the reference corpus with the highest number of matches is retrieved. Put together, these sentences constitute the *best match* equivalent reference translation.

The authors note that the scores decrease exponentially when n increases (since the probability to get common phrases decreases when the length of the phrases increases). However, longer n-grams are important too, since they seem to be better indicators of readability than unigrams. That is why the authors finally combine them in a uniform geometric average.
In the remainder of the article, Papieni *et al.* compare their methods to human evaluation. It turns out that it correlates very well, giving the same ranking of a set of translations produced by two humans and three translation systems. All the differences found between systems are statistically significant. In conclusion, even though BLEU obviously misses some linguistic details and subtleties, it seems to be an efficient and affordable way to experimentally evaluate the difference in performances between two systems. The authors emphasize that this method may be particularly useful in Machine Translation to evaluate the effects of daily changes on Automatic Systems, efficiently supplementing expensive human evaluations. This is exactly what we are seeking for CALLISTO! Now, how can we adapt this method to Automatic Summarization and be sure it correlates equally well with human judgments? That is what Lin and Hovy have been looking at in [Lin and Hovy, 2002].

4.1.2 Manual and Automatic Evaluation of Summaries (Lin and Hovy, 2002)

This paper uses the data from DUC 2001 [DUC, 2001 and 2002] to conduct several experiments with manual and automatic evaluation methods. The authors have access to all the summaries produced by the various systems involved and to their respective evaluations by human assessors through the Summary Evaluation Environment tool, implemented by Lin [Lin, 2001]. SEE enables judges to compare a proposed summary to a reference summary in terms of overlap between *units*. In DUC 2001, these units were sentences. This point of view is equivalent to deciding if each sentence is *totally*, *mostly*, *somewhat*, *hardly* or *not at all* covered in the summary to evaluate. Using those degrees of coverage, Lin and Hovy compute a weighted Recall measure:[4]

$$\frac{\text{Number of Model Units marked}}{\text{Total number of Model Units in the model summary}}$$

[4]Note that, in the article, there was also a coefficient C in the formula depending on the completeness of coverage: 1 for all, 3/4 for most, 1/2 for some, 1/4 for hardly any, and 0 for none. However, as there were already enough differences to compare with one measure, Lin and Hovy decided, for the sake of simplicity, to set C to 1 in all their experiments.

The results confirm a former study [Rath *et al.*, 1961] about the instability of human evaluation. It turns out that the summaries evaluated have received at least two different coverage scores for the same units in 8 to 20% of the cases. Furthermore, the ranking of the different systems changes depending on the formula chosen to combine the different human judgments[5]. But, as Lin and Hovy notice, the rerankings are not producing very different global results and, on the whole, clusters of systems and humans appear and are ranked the same way whichever measure is chosen. For instance, the worse of the two human summarizers is always much better than the best automatic summarizer.

In this context, we cannot expect an Automatic Evaluation Method to correlate perfectly with a human judge, since two human judges poorly agree with one another. However, we want this method to identify properly the clusters established by all judges and by the original ranking. Based on [Papieni *et al.*, 2001], Lin and Hovy have tried to adapt this automatic measure to summarization. They keep the idea of comparing a text with one or several model summaries by matching N-grams one to four tokens in length. But as they had only one model unit per summary, they did not compute the intricate modified precision with brevity penalty but used simple recall[6] instead:

$$\frac{\text{Number of matched n-grams between Model Unit and Summary}}{\text{Total number of n-grams in Model Unit}}$$

Lin and Hovy considered several N-gram matching methods: with or without stemming, with several weights for various N-grams (chiefly unigrams and bigrams). Based on the scores obtained they proposed a ranking for the systems that participated in DUC 2001, and compared this ranking with the official one established by human judges. Based on the Spearman rank-order correlation coefficient (ρ)[7], they found that *all* the methods considered correlate well with human evaluation, at least as well as two people agree. This finding is significant. It suggests that mechanical assessment cannot achieve perfect agreement with human judgment because human evaluation is itself inherently confounded. There does not exist one, and one only,

[5]The configurations tried are Maximum, Minimum, Average, Majority and Original human judgments.

[6]One problem is that the length of the summary is not considered in the evaluation. We will return to this in more detail.

[7]The Spearman rank-order correlation coefficient is used to measure how close the rankings produced by two ordinal variables are. Computing it includes computing the sum of the squared differences between rankings and, in the end, the coefficient is between 0 and 1, 0 meaning no correlation and 1 meaning complete correlation.

accurate human rating. Nonetheless, the mechanical assessment that Lin and Hovy propose is as acceptable as a human assessment.

4.1.3 Description of the method

Our framework implements the method in [Lin and Hovy, 2002] which they found to agree most closely with the official DUC ranking. Weights of 1/3 are applied to unigrams and 2/3 to bigrams output by the Porter stemmer after stop words have been removed from the text. This produces reference lists of stemmed unigrams (length N) and of stemmed bigrams (length $N-1$) from the model summary. The summary under evaluation is then stemmed similarly and the number of its stemmed unigrams and stemmed bigrams appearing in the reference lists counted. This number is the overlap. The recall measure is computed as:

$$Recall = \frac{1}{3}\frac{Overlap_{Unigrams}}{N} + \frac{2}{3}\frac{Overlap_{Bigrams}}{N-1}$$

The algorithm we wrote to compute the Recall measure can be found in appendix B, along with the corresponding complexity study.
The complexity of this algorithm is in $O(n^2)$ if n is the length of the summary to evaluate. It is therefore about the same as our KPiA method when the text is not very much longer (compression rate above 10%) than the summary, which is the case for news articles. In practice, however, KPiA calculation is faster because of the absence of stemming and because the bottleneck[8] in its procedure occurs only once for each text, no matter how many extracts are evaluated. The maximal complexity of Lin and Hovy's method is encountered in every run, that is for every summary generated for each text[9]. Therefore, here, contrary to the KPiA method, the worst case is the average case as well.

4.1.4 Lin and Hovy's method and CALLISTO

Among the various methods in Automatic Evaluation, very few are suitable for our needs. Let us come back to the classification presented in chapter 2.

[8]Recall that the normalization step is by far the most time-consuming of the KPiA algorithm.

[9]This number, which is the number of possible system vectors, was 300 in most of our experiments.

Extrinsic methods

Extrinsic methods do not suit our application.

- First, this kind of evaluation is always biased toward a single task. As a result, even if it works, evaluation might sometimes reward poor, or at least poorly readable, summaries that happen to contain answers to specific questions.
- A more serious failing is that in our case the only data available—approximately 1100 newspaper articles with abstracts manually prepared by DUC assessors—is not suited to relevance assessment or reading comprehension because we only have model abstracts and texts. If we plan on building a reading comprehension framework for example, questions or themes would need to be developed and answered manually for every text. Indeed, automatic fact extraction methods are not simple and reliable enough to make sure that we would be only evaluating CALLISTO[10].
- The decisive factor is that, if conducted manually, extrinsic methods are highly time-consuming and we have many thousands of summaries to evaluate. We must therefore look elsewhere for more promising approaches.

Intrinsic methods

As already stated in Chapter 2, CALLISTO is a system extracting full sentences. That is, we are less interested, at least for now, in methods evaluating readability and quality. We will therefore focus on informativeness.
As highlighted in [Jing *et al.*, 1998], most of the methods judging informativeness evaluate the suggested summary against an "ideal" summary. This is what we want to do since the DUC 2002 dataset comes with a reference summary for every text. We employ a sentence-extracting summarizer so one option would be *recall* on the most relevant sentences. But the model summaries are abstracts. Although abstracts do not always differ much from extracts, their vocabulary may not be quite identical. Extracts are literal copies of fragments of the source document, while abstracts may have wording changed for clarity. A hard pre-processing phase would therefore be needed to equate sentences prior to computing recall. As a result, our only

[10]The results we would obtain might as well depend on the performance of the fact extracter on the reference abstract and on the summaries produced by CALLISTO. In any event, they are difficult to trust without careful human double checking.

practical choice is to use a content-based measure. Furthermore, as pointed out in chapter 2, this kind of measure is particularly suited when working on DUC news reports because the reference abstracts are mostly cut and paste material from the source. As they rarely use synonyms or generalizing terms, the content phrases between the models and the summaries to test are likely to be the same.
Now, not any content-based measure will do: we need more insight into its reliability. How do we know that the scores obtained are indeed representative of the quality of the summaries? So far, in summary evaluation, no automatic measure has received a wide agreement in the research community. Therefore, a method to show that a measure is reliable is to establish it correlates well with human judgment. That is why we will not use the KPiA evaluation method used for training CALLISTO in its current incarnation: we do not know how it correlates with human judgment and the histogram study in section 4.2.4 rather provides evidence to the contrary. Now, in [Lin and Hovy, 2002], Lin and Hovy found that many of their measures were in good agreement with human evaluation, agreeing at least as well as two human judges with one another. The one which agrees most is the one we have presented in the previous section and that is the one we are going to use in our framework. But first, let us discuss in detail its main pros and cons and present more arguments distinguishing the KPiA method from the one we chose.

4.2 Strengths and Limitations of the method

The strengths of the method have already been discussed: it is easy to compute without human intervention (though the stemming step is a bit time-consuming) and correlates well with human judgment. Let us now look at other criteria defined in [Popescu-Belis, 1999] and related to evaluation measures.

4.2.1 Coherence

[Popescu-Belis, 1999] also cites coherence as a criterion a measure should respect. There are two conditions related to coherence. The first one is the *Upper Limit condition*: a score of 1 is obtained by perfect[11] answers and

[11]The word "perfect" is the one used in the article. However, we have already emphasized there does not really exist such thing as a perfect summary. Therefore, we will rather talk about model or reference summaries.

only by them. Our measure does not strictly speaking verify this equivalence and it is part of its drawbacks. Let us discuss why.

- If a summary scores 1 then it is a reference summary.
  If a summary scores 1, it contains the same content words as the model in the same order. There can be minor variations on stemming and stop words but nothing really important for informativeness. Therefore, the suggested summary *contains* the model. However, as we are using recall, there could, as well, be irrelevant sentences before or after, which are not taken into account. This is a first flaw of the method and we address it at the end of this section by introducing *F-Score*.

- If we have a reference summary then it scores 1.
  Prior to considering this rule, we need to answer another question: what is a reference summary? One observation that is widely accepted in Automatic Evaluation now, and that is stated in [Jing *et al.*, 1998] and [Lin and Hovy, 2002], is that there is not one, and one only, acceptable model for a given text. Even if we ignore the problems of synonymy and lengths, there can be several salient aspects in a text and no objective way to determine whether one is more important than the others. This is one intrinsic difficulty of text summarization and we have not found a way to overcome it in the current version of the framework. One solution is to follow [Papieni *et al.*, 2001] and use several reference abstracts. But it requires human resources that we do not have, on the one hand, to write these different models, and, on the other hand, to repeat an evaluation like [Lin and Hovy, 2002] to make sure that this new method still agrees with human judgment.

The *Lower Limit condition* is subject to the same flaws but it seems like a less serious issue than in the case of the Upper Limit condition.

- All the responses receiving a 0 should be the worst possible ones.
  A score of 0 means that the suggested abstract shares absolutely no content word with the model summary. It can be assumed that it is indeed a very bad summary even though the reservations expressed about various different acceptable summaries are still applicable here. That is, a summary receiving a 0 could happen to have some value if it contained only synonyms of the content words in the reference abstract. But, as said before, this case is particularly unlikely to happen in sentence extraction.

- All the poor responses should receive a low score.
  From the correlation in [Lin and Hovy, 2002], this can also be assumed

to be true. A system that performs poorly is ranked low by the method, as much as by human judges.

The last criterion for coherence in [Popescu-Belis, 1999] is *regularity* or, maybe, the term *monotonicity* would be more appropriate for this property. That is: if a summary is better than another then it should receive a better score. The argument is the same as the one just stated: as the correlation with human judgment is assumed, we can talk about *coarse* monotonicity. Absolute monotonicity seems impossible to achieve since human judges do not agree well enough with one another. However, a measure that identifies the same clusters in the data, as the human judges do, can be viewed as acceptable in terms of monotonicity.

4.2.2 Readability

In addition to the problems of coherence just discussed, it is important to note that Lin and Hovy's method, no matter how well it agrees with human evaluation, is far from ideal since it cannot consider issues such as readability, consistency or internal coherence of the suggested summary. Those are typical recurrent problems in summarization by sentence extraction and are not addressed here. However, given state-of-the-art text summarization, it seems reasonable to dismiss them for the time being: let us find a reliable way to extract the right sentences from a text and then we will be able to consider readability evaluation. Of course, there exist some methods to "repair" a text built from extracts and make it more readable as discussed in section 2.2.4 which deals with revision.

4.2.3 F-Score

A tested summary should agree with the model not only in content but also in length, an aspect which is measured by *Precision*. Combining both informativeness and brevity is possible through F-Score, a measure often used in Information Retrieval:

$$\text{F-Score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Popescu-Belis in [Popescu-Belis, 1999] explains that the F-Score formula uses the harmonic average[12] rather than, for instance, the arithmetic average because we want F-Score to be low, whenever one out of Recall and Precision is really low[13]. The summaries have to be both informative and concise, not only one of those.
In [Papieni *et al.*, 2001], using F-Score as such was not possible because of multiple references. This is a problem that we do not have. Therefore there is no reason not to consider F-Score, along with Recall. The only point is that we need to prove the correlation with human judgment. As discussed in [Lin and Hovy, 2002], the important point is to preserve ranking. And we have a dataset of 334210 summaries on which it is very easy to see if the measures correlate. So we computed the correlation of Recall and F-Score with the Spearman Rank-order correlation[14] as Lin and Hovy do in their article:

$$\boxed{\rho = 86.94\%}$$

So, now, as Recall correlates with human judgment at more than 98% [Lin and Hovy, 2002] and the correlation between Recall and F-Score is satisfactory, we can assume that F-Score correlates well with human judgment. The transitivity rule we use here is only acceptable because the correlation scores are very high. It is strictly speaking only true when the Spearman rank-order correlation is 100%[15]. However, as both scores are high, we can assume that the correlation between F-Score and human judgment is good. Note that this practical correlation between F-Score and human judgment was also intuitively predictable. Let us consider the unigram case, O being the overlap in stemmed content unigrams between the summary to test, of

[12] If we have two numbers x and y different from 0, we can compute a weighted average $\frac{1}{\alpha\frac{1}{x}+(1-\alpha)\frac{1}{y}}$ with $\alpha \in [0,1]$. If $\alpha = \frac{1}{2}$, we get the harmonic average:

$$\frac{1}{\frac{1}{2}(\frac{1}{x}+\frac{1}{y})} = \frac{2*x*y}{x+y}$$

If x or y are 0, the harmonic average is also 0.

[13] Typically, if one is 0 and the other 1, we prefer the result to be 0 rather than .5.

[14] If we want to compare two rankings of n objects, and we denote a_i, b_i the rankings of the same object i, the Spearman Rank-order correlation is:

$$\rho = 1 - 6\frac{\sum_{i=1}^{n}(a_i - b_i)^2}{n(n^2-1)}$$

It is equal to 1 (or 100%) for a perfect correlation and to -1 for a "perfect" disagreement.

[15] In that case, if two measures have exactly the same ranking than a third one, the correlation between the two is 100% too.

length S, and the model summary, of length M:
$Precision_{Unigrams} = \frac{O}{S}$ and $Recall_{Unigrams} = \frac{O}{M}$
Therefore:

$$Precision = \frac{M}{S} * Recall$$

Therefore, Precision is a multiple of Recall divided by the length of the suggested summary (for a given text, M is a constant). Finally, combining Recall which correlates well with human judgment and the principle "the shorter, the better" (which also agrees with the human definition of a good summary) should intuitively give a measure that correlates well too. And if Precision and Recall correlates well with human judgment, it should also be true for their harmonic average: F-Score.
Let us emphasize that Recall, Precision and F-Score as we use them are similar to the Information Retrieval context. Indeed, if we consider a classical information retrieval experiment, the results are reported in terms of:

$$Precision = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

and:

$$Recall = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents in the database}}$$

Therefore, a similar relation holds between both:

$$Precision = \frac{\text{Total number of documents in the database}}{\text{Total number of documents retrieved}} * Recall$$

In the same way M is constant for a given text, the total number of documents in the database is not different from one experiment to another if the database is the same[16]. This discussion leads to one conclusion : in the Information Retrieval results as in our framework, there are two degrees of freedom and we need two measures to take them into account. If we do not want to favour one over the other (in our case, informativeness over brevity), we can use an average between those measures, as F-Score is, to report the results. Therefore, both Recall and F-Score will be reported in the rest of the thesis, Recall being the initial measure of informativeness used by Lin and Hovy and F-Score being an indicator of the quality of the compromise between information content and brevity.
Before moving on to the evaluation framework, let us emphasize that given the flaws discussed in this section, it should be clear that neither F-Score nor

[16]Which is generally the case: it does not make much sense to compare Recall and Precision scores obtained on different datasets.

Recall are reliable as an absolute way of judging a system and it would not make sense to base an event such as DUC, for instance, on these measures only. However, it seems satisfactory enough to give hints about the output of our system and when we need frequent and reasonably quick evaluations.

4.2.4 Recall and F-Score vs. KPiA

We now study practical differences between the measures on our dataset. First, we look at statistical properties with three histograms and then, we consider a specific example to illustrate what the measures do and what kind of distinctions there exist between them.

Histograms

We have generated summaries with CALLISTO's text summarization component[17]. Altogether, there are 334210 summaries for 1121 texts. The three measures were calculated for each of them and the histograms presented in Figure 4.1 are the distributions over the dataset.
The first comment is that they do not look alike at all. In particular, the KPiA histogram shows a very high peek for 1 (more than 15000 summaries). This means that there are a large number of summaries which are as good as or better than the one identified during the normalization process. On the contrary, Recall and F-Score have a more regular shape[18] and, above all, no accumulating in the area of *very best* summaries. This shows that KPiA evaluation may not correlate very well with Recall and F-Score, a disagreement which is slightly confirmed by the Spearman rank-order correlation:

- Recall and KPiA: 73.20%
- F-Score and KPiA: 63.80%

More generally, this is a very undesirable property for our task because we are especially interested in discriminating among the very best summaries. We want the learning algorithm to identify the combination of text vectors and system vectors leading to the best summaries. Now, it is unlikely that the quality of all these summaries ranked 1 is indeed equivalent and therefore the KPiA measure does not provide great help to discriminate between good and very good summaries.
These histograms also enable us to deliver a few comments about the discretization process. As we said in chapter 3, it is currently done through the

[17] See the next section for the parameters used.
[18] The F-Score curve is almost a gaussian.

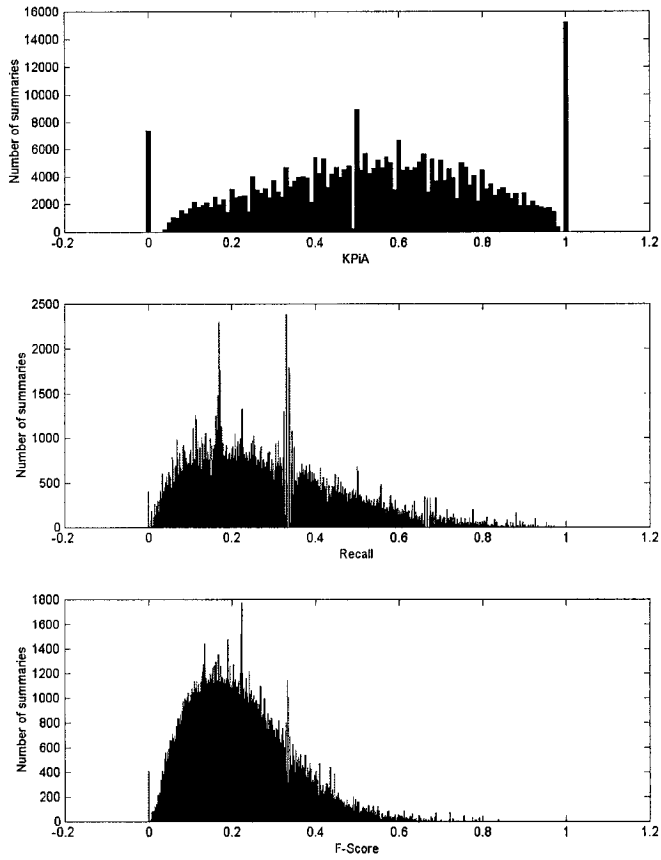

Figure 4.1: Histograms.

K-means algorithm [Mac Queen, 1967]. The use of this algorithm is justified when the underlying distribution in the data is approximately K gaussians. Such an assumption could work for the KPiA distribution. This measure has a tendency to create clusters because the scores before normalization are smaller than in the other two evaluation procedures and, consequently, the number of different possible scores after normalization is smaller too[19].

[19]For a given abstract, there are fewer Key Phrases than content unigrams and bigrams. The example presented in the next subsection will help to understand why.

For Recall and F-Score, however, this discretization method does not seem more appropriate than other much simpler ones that we will talk about in the next Chapter.

Comparison on an Example

Now, to understand better the differences between the three evaluation measures, we will study an example. That is a text from DUC 2002 dataset which has been carefully chosen as one on which the measures do not agree very much[20].
Uninteresting results occur for short texts (less than 12 sentences) because the best configurations are necessarily those for which the whole text is extracted. Besides, on this text, the abstract agrees well with the text and is mostly but not exclusively an extract as one can see in tables 4.1 and 4.2.
This example can be seen as typical of the documents we are dealing with since the first paragraph is already a good summary. And the human-written abstract begins with sentences extracted from the beginning of the text, only slightly combined and re-formulated. The last sentence is different since it covers the middle of the text (two paragraphs) in a few words. The end of the text is not taken into account. However, it does not bring much new information and can be seen as an extension of the beginning. Therefore, from a human point of view, the reference abstract is indeed good.
In tables 4.3 and 4.4, we show what the abstract looks like at intermediate steps needed by the evaluation methods before getting the measure. For the former, it is a representation of the reference summary after filtering the stop words and stemming the others, which will be used to compute Lin and Hovy's Recall and F-Score scores. Proper nouns are generally not part of the stop word list so they are kept in the list and are given more importance than they had at the beginning. In our example, the proper nouns (Yasser Arafat, PLO or Khalil Wazir for instance) represent almost 25% of the content stemmed list whereas they represented only 10% in the original reference abstract. In this case, this is good since they are the focus of the article. In some other cases, it could be less appropriate. Besides, the filtering is not perfect and keeping the stem "unit" for "United States" does definitely not make much sense[21]. The other method outputs a list of

[20]However, we must mention that, in general, the order of the measures is: F-Score smallest, Recall and KPiAs largest. Only on very specific cases, another order can be observed.

[21]This is because "states" is on the stop word list and "united" is not.

Yasser Arafat on Tuesday accused the United States of threatening to kill PLO officials if Palestinian guerrillas attack American targets. The United States denied the accusation. The State Department said in Washington that it had received reports the PLO might target Americans because of alleged U.S. involvement in the assassination of Khalil Wazir, the PLO's second in command.

Wazir was slain April 16 during a raid on his house near Tunis, Tunisia. Israeli officials who spoke on condition they not be identified said an Israeli squad carried out the assassination. There have been accusations by the PLO that the United States knew about and approved plans for slaying Wazir.

Arafat, the Palestine Liberation Organization leader, claimed the threat to kill PLO officials was made in a U.S. government document the PLO obtained from an Arab government. He refused to identify the government.

In Washington, Assistant Secretary of State Richard Murphy denied Arafat's accusation that the United States threatened PLO officials. State Department spokesman Charles Redman said the United States has been in touch with a number of Middle Eastern countries about possible PLO attacks against American citizens and facilities. He added that Arafat's interpretation of those contacts was "entirely without foundation." Arafat spoke at a news conference in his heavily guarded villa in Baghdad, where extra security guards have been deployed. He said security also was being augmented at PLO offices around the Arab world following the alleged threat.

He produced a photocopy of the alleged document. It appeared to be part of a longer document with the word "CONFIDENTIAL" stamped at the bottom. The document, which was typewritten in English, referred to Wazir by his code name, Abu Jihad. It read: "You may be aware of charges in several Middle Eastern and particularly Palestinian circles that the U.S. knew of and approved Abu Jihad's assassination."

"On April 18th (a) State Department spokesman said that the United States 'condemns this act of political assassination,' 'had no knowledge of' and 'was not involved in any way in this assassination." "It has come to our attention that the PLO leader Yasser Arafat may have personally approved a series of terrorist attacks against American citizens and facilities abroad, possibly in retaliation for last month's assassination of Abu Jihad." "Any possible targeting of American personnel and facilities in retaliation for Abu Jihad's assassination would be totally reprehensible and unjustified. We would hold the PLO responsible for any such attacks." Arafat said the document reveals the U.S. administration is planning, in full cooperation with the Israelis, to conduct a crusade of terrorist attacks and then to blame the PLO for them. "These attacks will then be used to justify the assassination of PLO leaders." He strongly denied that the PLO planned any such attacks.

Table 4.1: The original article (Associated Press, ap880510-0178)

The United States has denied an accusation by Yasser Arafat that the United States is threatening to kill PLO officials if Palestinian guerrillas attack American targets. The State Department said it has received reports that the PLO might target Americans because of alleged US involvement in the assassination of Khalil Wazir, the PLO's second in command, in a raid on his house near Tunis on 16 April. Israeli officials said that an Israeli squad carried out the assassination. Apparently, Arafat's claims stem from his interpretation of US correspondence with Middle East countries about possible PLO attacks against American citizens and facilities.

Table 4.2: The human-written abstract

unit den accus yasser arafat unit threaten kill plo offici palestinian guerrilla attack american target depart receiv report plo target american alleg involv assassin khalil wazir plo command raid house tuni april israe offici israe squad carr assassin apparent arafat claim stem interpret correspond middle east countri plo attack american citizen facil

Table 4.3: Model summary after filtering and stemming

Key Phrases[22]. A priori, it gives acceptable results on that text. One of its common drawbacks, which is to extract long and incoherent phrases, is not too serious here and the phrases "Palestinian guerrillas attack American targets" or "alleged US involvement" are in fact relevant. Some of the last ones may not be ("house" or "april") but the whole list can be deemed somehow representative of the article.

Now, we present three summaries output by CALLISTO with their respective scores with each of the methods.

- With c-k-e-3-8-2[23], in table 4.5, a score of 1 is achieved with KPiA, meaning that the summary does as well as the normalization process, selecting the sentences by KPiA density. The evaluation method shows some limitations here, not only with the fact that the text is not co-

[22]The numbers near the phrases are the numbers of times the given phrase is found in a reference extract, computed from the text as explained in chapter 3, and used for normalization.

[23]C99, Kea, Exact matching, Keycount=3, Sentcount=8, Hitcount=2.

Palestinian guerrillas attack American targets 1		
alleged US involvement	Israeli squad carried 1	Middle East countries
kill PLO officials 2	US correspondence	American citizens 2
received reports 1	target Americans 1	interpretation 1
assassination 8	Yasser Arafat 2	United States 6
Khalil Wazir 1	PLO attacks 1	threatening 1
accusation 2	facilities 3	command 1
denied 3	house 1	Tunis 1
April 2	raid 1	PLO 10

Table 4.4: The Key Phrases in Abstracts

herent[24] (the last quote is wrongly attributed to Yasser Arafat), but also by tolerating low information sentences such as "Arafat spoke at a news conference in his heavily guarded villa in Baghdad, where extra security guards have been deployed", which does not even contain any KPiA. One of the reasons why the summary is ranked high despite this sentence may be that there are many other and different KPiAs in the other sentences (which means that achieving a score above 1 is possible on this text). Apart from the end, this summary can be deemed as acceptable but is heavily penalized by Recall and F-Score because it does not include the sentence mentioning the assassination of Khalil Wazir and because it contains a lot of redundant information (implying a loss in Precision and, therefore, in F-Score).

- With c-n-e-8-3-2[25], in table 4.6, we can examine one of the very rare cases when F-Score is larger than Recall and KPiA. As expected, this is an excellent summary which illustrates our intuition of this measure being a good compromise between informativeness and brevity. As a matter of fact, it is exactly the leading text, except that the information about date and place of the assassination has been discarded, exactly as in the human-written abstract. The fact that the Recall and KPiA scores are lower can be explained by this missing information plus the end of the human-written abstract (regarding the origins of the report), which is not tackled here.

[24]This problem can occur as well with Lin and Hovy's Recall and F-Score, it is just slightly less likely because, in the filtered and stemmed bigram list, the order of the text is kept (therefore, bigram counts somehow reward a summary which preserves ordering) whereas KPiAs do not take it into account.

[25]C99, NP Seeker, Exact matching, Keycount=8, Sentcount=3, Hitcount=2.

Yasser Arafat on Tuesday accused the United States of threatening to kill PLO officials if Palestinian guerrillas attack American targets. Arafat, the Palestine Liberation Organization leader, claimed the threat to kill PLO officials was made in a U.S. government document the PLO obtained from an Arab government. In Washington, Assistant Secretary of State Richard Murphy denied Arafat's accusation that the United States threatened PLO officials. State Department spokesman Charles Redman said the United States has been in touch with a number of Middle Eastern countries about possible PLO attacks against American citizens and facilities. He added that Arafat's interpretation of those contacts was "entirely without foundation." Arafat spoke at a news conference in his heavily guarded villa in Baghdad, where extra security guards have been deployed. "It has come to our attention that the PLO leader Yasser Arafat may have personally approved a series of terrorist attacks against American citizens and facilities abroad, possibly in retaliation for last month's assassination of Abu Jihad. We would hold the PLO responsible for any such attacks."

Table 4.5: Configuration c-k-e-3-8-2 (R=0.39 F-S=0.27 KPiA=1)

Yasser Arafat on Tuesday accused the United States of threatening to kill PLO officials if Palestinian guerrillas attack American targets. The State Department said in Washington that it had received reports the PLO might target Americans because of alleged U.S. involvement in the assassination of Khalil Wazir, the PLO's second in command. Israeli officials who spoke on condition they not be identified said an Israeli squad carried out the assassination.

Table 4.6: Configuration c-n-e-8-3-2 (R=0.54 F-S=0.63 KPiA=0.38)

Yasser Arafat on Tuesday accused the United States of threatening to kill PLO officials if Palestinian guerrillas attack American targets. The State Department said in Washington that it had received reports the PLO might target Americans because of alleged U.S. involvement in the assassination of Khalil Wazir, the PLO's second in command. Israeli officials who spoke on condition they not be identified said an Israeli squad carried out the assassination. "It has come to our attention that the PLO leader Yasser Arafat may have personally approved a series of terrorist attacks against American citizens and facilities abroad, possibly in retaliation for last month's assassination of Abu Jihad." Arafat said the document reveals the U.S. administration is planning, in full cooperation with the Israelis, to conduct a crusade of terrorist attacks and then to blame the PLO for them. "These attacks will then be used to justify the assassination of PLO leaders."

Table 4.7: Configuration s-n-e-8-5-2 (R=0.64 F-S=0.51 KPiA=0.37)

- With s-n-e-8-5-2[26], in table 4.7, KPiA is still quite low but Recall is larger than F-Score. The extract begins the same way as the previous one but adds a few sentences from the end of the text. But the latter introduces problems of coherence, such as a quote the origin of which is unknown and a dangling reference "the document". The last two sentences cover acceptably the last one of the human-written abstract, hence the score is good with Recall. The Precision remains good so F-Score is still large too. Key Phrase in Abstract, however, still outputs a bad score, probably due to the fact that the KPiAs involved are the same repeated several times and there are some minor variations to which KPiA is sensitive ("Arafat" instead of "Yasser Arafat" for example).

To sum up, we believe that this analysis, along with the study of histograms in the previous subsection, has:

- highlighted the difficulty of ranking an extract in general, even for humans.
- subsequently illustrated the challenge of finding a measure reliably accounting for the quality of a summary. A priori, each of the three methods considered have advantages and drawbacks.

[26]Segmenter, NP Seeker, Exact matching, Keycount=8, Sentcount=5, Hitcount=2.

- shown the limitations of KPiA and the need to test the other two methods for evaluation inside CALLISTO. Until now, indeed, we have treated CALLISTO as a black box for which we simply evaluate the output. If, however, we look at the internal mechanisms, we see that the C5.0 machine learner is being trained to improve KPiA ratings rather than Recall and F-Score values. An obvious improvement is to train C5.0 with Recall or F-Score data obtained from Lin and Hovy's method. If C5.0 can be applied to this data successfully we would expect to get better results ultimately when the function used for evaluation is the same as the one used in training. Changing the evaluation measure will be one of the first experiments presented in the next chapter.

4.3 The Evaluation Framework

4.3.1 Methodology

This study has been motivated by the need to evaluate the results of CALLISTO and to assess the impact of its modifications on the quality of the summaries produced by the system. Design choices at several levels were initially made without knowing that they were indeed apt. The evaluation framework described here should enable us either to validate these choices, or to identify better alternatives.
The dataset used consists of 1121 texts from the training and test sets, each one coming with a reference abstract. We applied the text summarization component of CALLISTO to each of these texts with the following set of parameters:

- Segmenter: TextTiling, Segmenter, C99 or None.
- Key Phrase Extractor: Extractor, Kea or NP Seeker.
- Matching: Exact[27].
- Sentcount: 3, 4, 5, 8 or 12.
- Keycount: 3, 4, 5, 8 or 12.
- Hitcount: 2.

[27] As we said previously, the alternative value Stemmed has not been evaluated yet.

We trained and evaluated the system's learning component in ten-fold cross validation[28]. The evaluation was conducted using Lin and Hovy's method[29]. Because this measure has been shown to correlate highly with the ratings of human judges, we can assume that the results we obtain are indicative of the actual quality of the summaries.

4.3.2 Statistics on the summarization data

Statistics on all summaries generated on the DUC dataset may be of interest. The average rating achieved using Lin and Hovy's method is:

- Recall: 0.287, with a standard deviation of 0.173.
- F-Score: 0.219, with a standard deviation of 0.118.

These figures give an idea of the scores which would be achieved without the learning step, if the configuration of CALLISTO were chosen at random for every new text. The standard deviations are quite large. This can be explained by the fact that very different news stories are dealt with, particularly in terms of length, and the segmentation/key phrase extraction method applied by CALLISTO works more or less successfully depending on the text.
Given that the system computes many alternative summaries for each text, it is possible to compute an average maximal and minimal possible score over this data set—these would be the worst and the best summary for every text. We found that, on average, Recall falls in the range $[0.063, 0.543]$ and F-Score in the range $[0.070, 0.378]$. Results can be normalized in terms of these ranges. Indeed, it does not make much sense to report the absolute values when we know that there are values we will not be able to reach. Therefore in the rest of the thesis the results will be reported in terms of percentage and the top lines in figure 4.2 will correspond to 100% in the

[28] In a k-fold cross validation approach, "each example from the data is used exactly once in a test set, and $k-1$ times in a training set" [Mitchell, 1997] (page 150). In our case, the data is divided into 10 equal parts, and learning repeated 10 times: 9 parts are used for training, 1 for evaluation. In effect, the results of learning – averaged – are more reliable.

[29] For each run, it is only necessary to evaluate one summary per text, the one that CALLISTO predicts will be the best. Actually, we *have* run the evaluation separately on all possible summaries generated by the system. The process was very time-consuming but presented two advantages: cutting the evaluation time in all subsequent experiments, and allowing us to compute statistics, such as the potential maximum score of the system on this data.

results of the next Chapters[30].
For instance, as we will see in the next Chapter, using KPiA evaluation and a 5-means discretization, the current version of CALLISTO gives a score of 0.324 with a standard deviation of 0.176 for Recall and a score of 0.224 with a standard deviation of 0.115 for F-Score; this represents 59.7% of the best possible score for Recall and 59.3% for F-Score. Similarly, the average score of all summaries of the 300 documents in the dataset is 52.8% of the best possible score for Recall and 57.9% for F-Score. Therefore the current version of CALLISTO is slightly above a random selection of CALLISTO's configurations. This means that we take advantage of the learning step but not as much as we would want: it is not close to the maximum so there is a lot of room for improvement in the remaining $100 - 59.3 = 41.7\%$.

4.3.3 Baseline

[Brandow *et al.*, 1995] has established that a very efficient baseline for automatic summarization of news reports is just extracting the first few sentences of the text (approximately the first one or two paragraphs). This can be explained by the journalistic style: when one reads a news report, one expects to find the answer to the W-Questions (who, when, where, what, possibly why) in the first lines. This can be deemed an acceptable summary in many cases. Therefore, it is important to include such challenging baseline in every evaluation study. Note that this baseline is highly genre-specific. If we were to work on another kind of texts, it would be quite probably much less interesting.
We have extracted the first sentences of each text in the corpus and run Lin and Hovy's evaluation method on them. The length of the summary is one of the parameter and can take the following values in our dataset: 3, 4, 5, 8 and 12. We have computed the baseline for every one of those. Standard deviations are computed in each case. The global average and standard deviation are calculated on the whole set of summaries produced and are presented in the last line of table 4.8.

The global averages which constitute the scores to beat are 0.291 that is 53.6% of the best possible score for Recall and 0.246 that is 65.1% for F-Score [31]. As expected and as we will see in the next chapters, this baseline

[30] Note that the figures on the x axis have no meaning there but will have some in later experiments.

[31] Note that, if we consider only 8-sentence extracts, both scores are even slightly better. But it would be equivalent to knowing in advance the optimal length of summaries, which

	Average Recall	Average F-Score
3-sentence extracts	0.201 +/- 0.120	0.229 +/- 0.128
4-sentence extracts	0.234 +/- 0.138	0.242 +/- 0.128
5-sentence extracts	0.266 +/- 0.145	0.251 +/- 0.128
8-sentence extracts	0.338 +/- 0.165	0.257 +/- 0.118
12-sentence extracts	0.414 +/- 0.180	0.253 +/- 0.109
Average	0.291 +/- 0.169	0.246 +/- 0.123

Table 4.8: Baseline scores

is extremely challenging, in particular for F-Score[32].

4.3.4 Statistical Significance

To establish the statistical significance thresholds of our results, we investigated methods used in Information Retrieval [Hull, 1993], since retrieving the most relevant sentences from a document can be seen as equivalent, at least from a statistical point of view, to retrieving the most relevant documents from a dataset.
Two-way ANOVA appears to apply well to our case since it enables us to get significance results when more than two methods are to be compared[33]. Following the notation used in [Hull, 1993], we denote y_{ij} the score of method (or configuration) j for query (or text) i, where $i = 1...n$ (n is equal to the number of texts, already known: 1121) and $j = 1...m$ (m will be the total number of configurations, not known yet). Let $\bar{y}_j$ represent the average score for method j and $\bar{y}$ equal the overall average.
We first have to perform an F-test (not to be confused with the F-Score measure) where F = MSR/MSE, where MSR is the mean-squared difference between retrieval methods and MSE is the mean-squared error. It has the

is probably dataset-dependent, and considering this baseline is therefore not really fair for CALLISTO.

[32]The fact that the baseline is more challenging for F-Score than for Recall can be explained as follows: the leading text does not necessarily contain all the relevant information (some conclusions may be stated somewhere else) but it contains very few sentences that could not belong to a summary. As the journalist has to expose the main elements, the leading text is very precise in term of informativeness.

[33]We have many configurations of the system to evaluate so pair-tests, i.e. tests which evaluate statistical significance for only two methods at the same time, would hardly be doable.

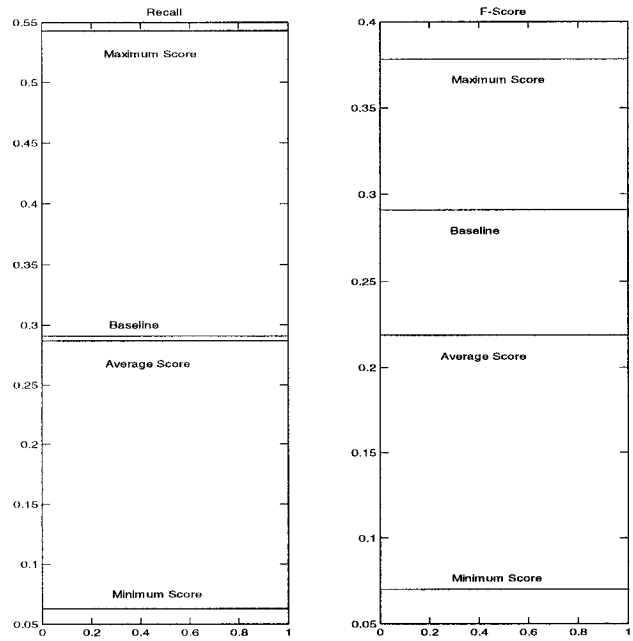

Figure 4.2: Statistics from the data.

form:

$$F = \frac{MSR}{MSE} = \frac{\frac{n\sum_j(\bar{y}_j - \bar{y})^2}{m-1}}{\frac{\sum_{i,j}(y_{ij} - \bar{y}_i - \bar{y}_j + \bar{y})^2}{(n-1)(m-1)}}$$

Then if F is much larger than 1, two methods can be seen as significantly different if the difference in their scores is above:

$$\frac{q^{\alpha}_{m,\nu} s}{\sqrt{n}}$$

where $q^{\alpha}_{m,\nu}$ is the studentized range statistic for $\nu = (n-1)(m-1)$ at significance level α and $s = \sqrt{MSE}$. We will specify the value of the statistical significance when reporting the results in the next chapter.

4.3.5 Order of the experiments

Now that we have a framework for evaluating the modifications we are trying on the system, we have another problem to solve: that of finding a global maximum when only local maxima are available.
Indeed, we can see the various configurations of the system as degrees of freedom and testing which is the best solution for a given problem, with all the other parameters remaining the same, is equivalent to finding one local maximum. Now, we would like to be as close as possible to the very best configuration of CALLISTO among all possibilities, that is the best solution regarding all parameters.
We have found no way of solving this problem absolutely and given the time needed for each experiment and the large number of possible experiments, it is likely that there does not exist any. Therefore, the order in which we solve the various problems is necessarily arbitrary. One reasonable assumption is that the different parameters are independent and therefore, the order of the experiments does not matter: once we have found the optimal solution for one particular degree of freedom, we keep this best solution for all subsequent experiments. Experiments show that this point of view is sometimes justified (for example, we will see that the influence of boosting is the same whether we use KPiA or Recall for training) but most of the time exaggerated (the same choices of measure do not give equivalent results depending which learning algorithm is used). Therefore, in what follows, we will try to justify the order of the experiments and explain intuitively why we think the best configuration we have is close to the local maximum, even if we do not have the time to conduct all possible experiments.
In addition, even if we have some arguments to believe in F-Score as a good measure to represent the quality of a summary, we have already mentioned that Lin and Hovy's study [Lin and Hovy, 2002] is limited to one set of news articles and one set of systems. Therefore, as already stated at the end of section 4.2.3, we believe that further evidence would eventually be needed to confirm the results obtained thanks to the framework. Some experiments involving human judges and the versions of CALLISTO we found interesting are part of future work to check if the results of automatic evaluation indeed correlate with the opinion of human judges.

4.4 Conclusion

In this chapter, we have presented an automatic evaluation framework for text summarization. Let us emphasize that this framework is neither system

specific nor data specific and the same methodology is applicable given:

- An automatic text summarization system, with a learning component.
- A dataset consisting of sufficiently many texts with model abstracts or extracts, only one for the time being but, following BLEU, it is reasonable to think about extending the method to handle several references replacing F-Score by the modified unigram precisions with weighted brevity penalty, as explained in [Papieni *et al.*, 2001].

The code to compute the similarity scores of Recall, Precision and F-Score between a model abstract and a given summary is in Perl and available on request, as the code to compute the statistical significance of the results. We believe that this framework could be useful to others in the text summarization research community because:

- It consists in a fully automatic way to assess effects of modifications to a text summarizer.
- It can, therefore, be applied to large corpora such as the one we have from DUC 2002, over 1000 texts, thus offering highly significant results. In the automatic summarization literature, corpora used for evaluation are usually much smaller, since human judgment is always involved at some point.
- The baseline on F-Score is very demanding and Galliers and Sparck Jones [Galliers and Sparck Jones, 1995] insist on the importance of having an efficient baseline in an evaluation framework.
- From the Information Retrieval literature, we provide a way of evaluating the statistical significance of the results.

We will now show in the remaining chapters how this framework is useful for CALLISTO.

Chapter 5

Internal Evaluation Mechanisms

Selecting Adequate Measures, Discretization and Selection Processes

5.1 Introduction

Now that we have a framework to evaluate modifications of the system, we can try settings different from the ones currently used in CALLISTO and observe the effects on the quality of the summaries. Before talking in the next chapter about the important choice of an adequate learning algorithm, there are a few issues associated with the Machine Learning component of the system into which we would like to have some insight. Those questions are listed below and will be explained in detail in section 5.2.

- What is an adequate evaluation measure of the quality of the summaries to train CALLISTO?
- Which discretization algorithm is appropriate for this measure and how many classes should we choose?

- Which strategy should we adopt to select a configuration among those yielding the best summaries?

We have thought of two different ways to test these changes.

- Skip the actual learning step altogether. In this experiment, we assemble the training file as if we wanted to feed it to C5.0: text vectors, system vectors and scores obtained (either with KPiA or with Recall or F-Score) discretized into several classes. But, instead of applying a machine learning algorithm and working on its predictions, we directly apply the selection process on the examples among the best class in this input file. This is equivalent to testing the system with a perfect learning algorithm: What would happen if the learner could predict the right classes all the time? Our hope there is to be able to study the effects of the choices of the measure, discretization process and selection strategy independently from the learner. This is discussed in section 5.3.

- Keep the current learning algorithm in the system: C5.0 [Quinlan, 1997] and make the changes presented in section 5.2. This way, we get results obtained with a real version of CALLISTO and we will be able to check if the observations made with a perfect learning algorithm are also valid for the real system. The results are given in section 5.4.

5.2 Internal Evaluation Mechanisms

5.2.1 The evaluation measure

In the previous chapter, we have already discussed, at length, the issues related to the use of the Key Phrase in Abstract evaluation method inside the system. We want to try to train the system, instead, with Recall and F-Score and see if we get any improvement. As we have seen in the previous chapter, Recall strictly measures informativeness and F-Score represents a trade-off between information content and brevity (i.e. the length of the summary is taken into consideration as well). Now, since we are evaluating the overall results of our system with the two measures just mentioned, we should get better results if we drop KPiA from the training process, assuming that those functions are learnable at all. The solution, however, is not that obvious because it is possible, for instance and as we will see later on, that the dependencies between the attributes and the F-Score measure

are too difficult to learn and that, paradoxically, we get better results using KPiA.

5.2.2 The discretization algorithm

As we are using classification and not regression algorithms[1], we need to discretize the class values. Recall that what we have in the training data are combinations of text vectors and system vectors associated with a given quality, the number of KPiAs, a score between 0 and 1[2]. We want to convert this number into a class value in the classical Machine Learning meaning. Now, the question is: how do we transform this continuous value into a discrete set of classes? This well-known problem is called *discretization* and it actually covers two sub-problems: which algorithm do we use and how many classes do we want?
Surprisingly, and despite substantial work[3] on feature discretization [Dougherty *et al.*, 1995], there are not many widely accepted algorithms for class discretization. Indeed, most attribute discretization algorithms (such as 1R [Holte, 1993] or entropy-based methods [Fayyad and Irani, 1993]) make heavy use of the information about the class to discretize. [Dougherty *et al.*, 1995] talk about *supervised* methods. What we have to solve is a different problem altogether since we do not have classes yet, we are limited to what this article identifies as *unsupervised* methods. Following Dougherty *et al.*, [Torgo and Gama, 1997] describe three methods commonly used for class discretization.

- The *equal width* method consists in discretizing the training data so that every class covers an equal range of values. It seems to be the most straight-forward way to discretize. One needs to go over the data once to determine the minimum m and maximum M and, then, the

[1] When the system was designed, using C5.0 for the learning component of CALLISTO seemed like a good idea *a priori* since it has been shown to be one of the best algorithms on many Machine Learning datasets. That is where the initial choice of classification over regression comes from. However, this is something we would like to test as well and we will discuss the use of regression algorithms in the next chapter.

[2] The problem is not different if we replace KPiA by Recall or F-Score as suggested in the previous subsection. Those are also measures between 0 and 1.

[3] Machine Learning is a very active research area and Machine Learning methods work best when all attributes and classes are discrete. Therefore, to take advantage of the most recent achievements in the field for problems involving continuous features, in particular in Decision Tree and Naive Bayes classification, which will be explained in more detail in the next chapter, it is usual to look for efficient discretization methods.

thresholds, if $k > 1$ classes are used are determined by the formula:

$$Threshold_i = m + \frac{i(M-m)}{k} \text{ with } i = 1 \ldots k-1$$

- The *equal frequency* method considers the numbers of examples in each class and makes sure they are equal. The assumption here is that the learning process will be better if the classes are balanced, whatever the values of the class are. If the distribution of examples is not uniform (and ours is certainly not), the thresholds will be completely different than those determined with the equal width method. The algorithm is not more time-consuming. We need to go over the data once too, and, if the total number of examples is N and the desired number of classes is k, define a threshold as the medium value between the examples number $i * int\left(\frac{N}{k}\right)$ and $i * int\left(\frac{N}{k}\right) + 1$ for $i = 1 \ldots k-1$. As the equal width method, the cost of equal frequency calculation is linear in the number of training examples.

- The k-means procedure [Mac Queen, 1967] relies on the prior assumption that the data is distributed into k gaussians. The idea is to compute iteratively the thresholds, each class being defined by the mean of the examples in it, so that every example has a class value closer to the mean of its class than any other. The thresholds are typically initialized with the equal width method and then the distance of each example to the means of the different classes is computed and its label is changed if it turns out that the example is closer to another class than the one it currently belongs to. If the class of at least one example has changed, all the means have to be re-computed and therefore another iteration is needed. Determining the average complexity of the k-means algorithm is tricky as several analyses show [Kanungo *et al.*, 2000]. It is in $O(i*k*N)$, if k is the number of classes, N the number of examples and i the number of iterations. Optimal implementations reduce the complexity to $O(k * N)$. It is still much longer than the two previous methods anyway. So we have to see if the improvement in the quality of the summaries is worth spending more time on the discretization step.

As for the number of classes, there is no ideal number, *a priori*, and preliminary experiments with the original version of the system seemed to show that using up to 20 classes was still relevant and that the error rate was

not dropping significantly despite the increase in the number of classes[4]. Therefore, we will keep trying all numbers of classes between 2 and 20.

5.2.3 The selection strategy inside the best class

Assume now that we have already discretized the data. Hence we have a "best" class, representing CALLISTO's configurations for which the evaluation measure is close to 1[5]. Then, to find the best configuration for a given new text, we compute the text vector and estimate, thanks to the trained learner, to which classes the associations of this text vector with every possible system vector belong. We discard the ones not belonging to the best class[6]. It would be nice to have only one configuration labeled as the best for each text and we would just apply it to get the resulting summary. It is, however, seldom the case. In general, we get a *set* of *best* configurations, each belonging to the best class and with no way, as far as the learning algorithm is concerned[7], to determine which one to choose. In the original version of the system, we simply picked the first one we found. We call this random strategy *first found.*

There are two other classifier-independent methods we could have thought about and experimented with. The first one consists in limiting the choice to the configurations producing the shortest summaries. After all, one of our aims is brevity and therefore, if the predictions are all equally good, we better keep the shortest. Inside the set of such configurations, we pick at random. This strategy is called *shortest.*

The last method is more complex and more time-consuming as well. The idea follows: Instead of using this set of possible configurations as an opaque bag from which we pick only one configuration, let us examine it and try to find the configuration most representative of this set. To do so, we compute

[4] As the number of classes increase, the learners tend to build models which overfit the data (since fewer training examples are available per class). Therefore, the generalization power and overall performance of the learner is worse on unseen examples.

[5] How close depends on the number of classes. The more classes there are, the smaller the number of examples belonging to the best one and, therefore, the better the quality of those is. When there are fewer classes, the best one contains more examples with scores further from 1.

[6] Or second best if there are no configurations for the best class, or third best if the first two classes are empty and so on.

[7] This is not always true. As we will see with Naive Bayes in the next Chapter, some learning algorithms also output some confidence in their predictions and this is one more way to discriminate between configurations.

a very simple distance over the system vectors[8] and select the one which is *closest* to all the others. It means that for each vector, we calculate the sum of the distances to all the others and keep the one for which this sum is the smallest. The intuition here is that if a specific value for a parameter is often predicted as producing good summaries for the given text, it may be a good idea to have this value in the configuration vector we select. A similar possibility to exploit this idea would be to choose, for each parameter, the value which is the most often predicted as giving good results. However, the problem with this strategy is that we are not sure that the configuration vector assembled from this majority vote predicts a good configuration (there could be unforeseeable dependencies between parameters). Therefore, we prefer to pick a vector which is in the set of very good summaries and *close* to all the others. As stated earlier, this strategy, called *similar*, is much more expensive than the others since it is quadratic in the number of best configurations. *Shortest* is linear and *first found* is constant.
Finally, other choices are possible depending on the learning algorithm used. C5.0, for example, produces a confidence score with each of its predictions. We can use this number to select the configuration predicted best by C5.0 with the highest confidence. This strategy called *confidence* is learner-dependent and cannot be tested when we skip the learning step. We will try it with C5.0 in section 5.4.3.

5.3 A perfect learning algorithm

5.3.1 Experiments

The number of prior choices we just discussed can be evaluated without considering the learning component of CALLISTO. If we want to test the *real* version of the system in cross validation, we will base the choices of the best configurations on C5.0 predictions for each text. To short-circuit the whole process, we can consider the input data directly (instead of the predictions), choose a measure, proceed to discretization and then choose a selection process and see where these basic steps lead to in our framework. There is therefore no learning whatsoever and this experiment is useful to check independently the effects of the different measures, discretization and selection processes. It is actually equivalent to having a perfect learner

[8]This distance is the sum of absolute differences, also known as L_1-distance. For each parameter, we count 1 if the values for the two vectors considered are different and 0 if they are the same. We then simply add up these numbers for the six parameters.

instead of C5.0, that is "What score would we achieve if the learning component of the system were able to predict the right class for every example?" This is a way of simplifying the problem and examining the issues we just presented apart from the learning algorithm.
As we will discuss it in the next section, we have to be careful about interpreting these results. In particular, if we trust the results without thinking about what they mean, we should conclude that the best measure will be Recall when we evaluate with Recall and F-Score when we evaluate with F-Score. The restriction we discussed when dealing with a real learning algorithm does not hold here: as we base our selection on the true scores, we will necessarily obtain those same large scores after the selection process. The interesting question here is only "what is the influence of the selection process?" However, these results will not imply anything regarding a version of CALLISTO with a real learning algorithm. It is possible that Recall or F-Score, or both of them, are too difficult to learn and that, for a given learning algorithm, we obtain better final results while keeping KPiA, for instance. Similarly, the choice of the discretization method is not independent of the learning algorithm so the following results should just be regarded as hints and not as definite ways to select between the different methods.

5.3.2 Results

Recall

To begin with, we report the *informativeness* scores, that is the ones obtained with Recall. The horizontal axis is the number of classes and the vertical axis the average Recall score. For each selection method, we try the three measures and the three discretization methods[9] (nine curves). The correspondence between figures and method is:

- *First found* in figure 5.1.
- *Shortest* in figure 5.2.
- *Similar* in figure 5.3.
- Summary of the best methods (the ones for which the training measure is recall, not surprisingly since no learning is involved) in figure 5.4.

[9]The last two letters in the configuration stand for the discretization process. K-means: km. Equal width: ew. Equal frequency: ef.

For these experiments, the statistical significance is 2.85% (that is, a difference of more than 2.85% between the scores of two methods is significant with a 95% confidence). We did not report the baseline score since the results are not obtained by an actual version of the system. The absolute quality of the summaries obtained with these *artificial* experiments does not make sense: there is no such thing as a perfect learner in real experiments. What we are looking for here are the relative differences from one configuration to another and the influence of the measure, discretization and selection process.

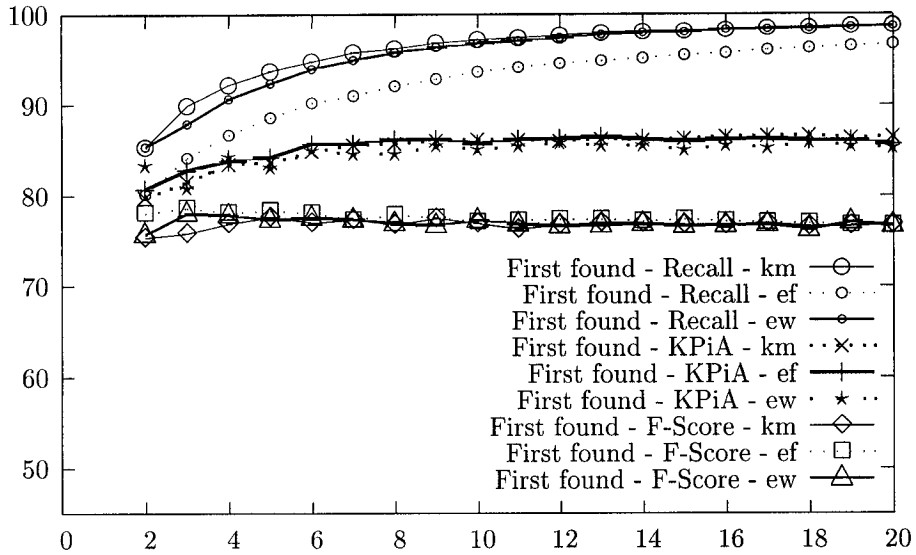

Figure 5.1: Recall Scores of the First found selection method.

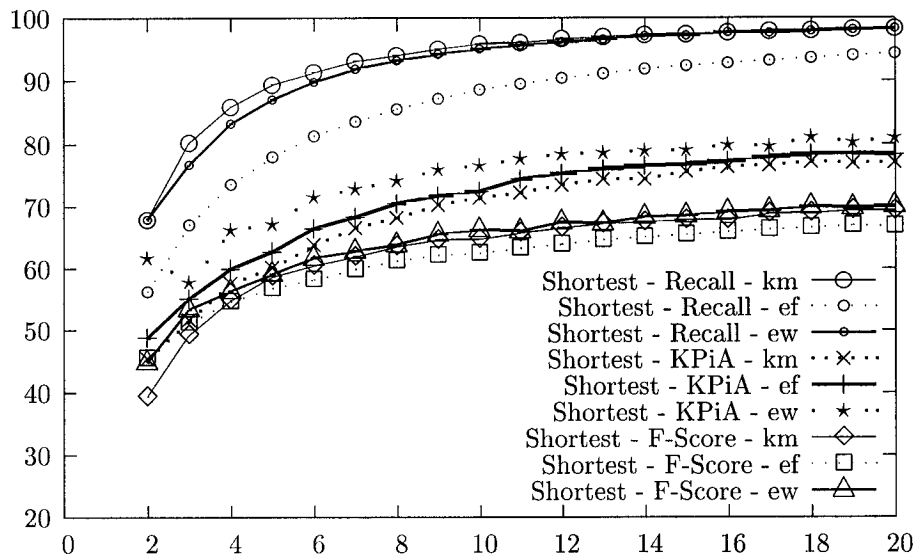

Figure 5.2: Recall Scores of the Shortest selection method.

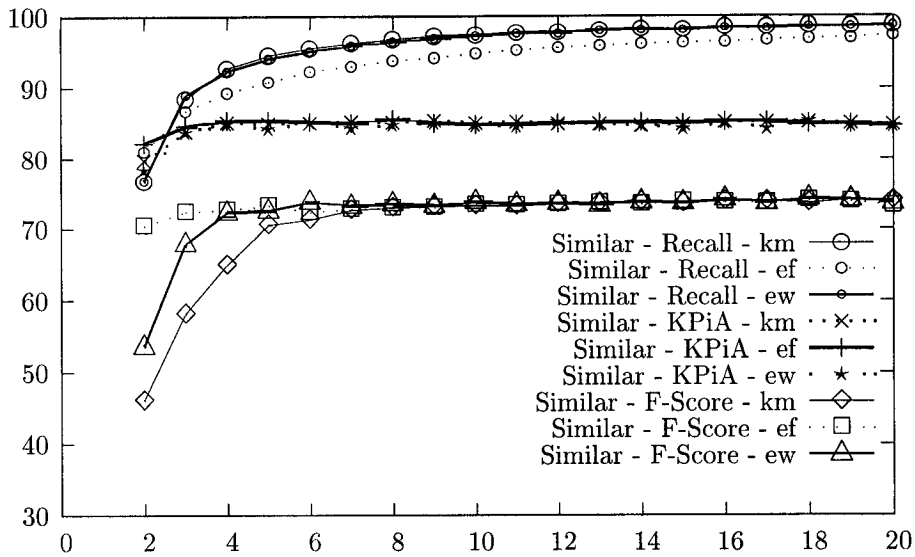

Figure 5.3: Recall Scores of the Similar selection method.

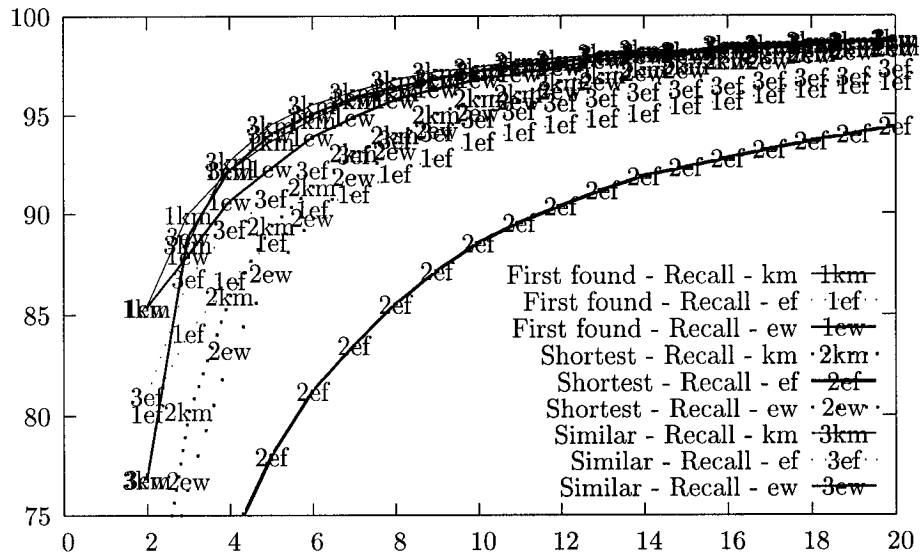

Figure 5.4: Recall Scores of the best methods.

The global shapes of the curves are quite alike in the first three figures. This shows that the influence of the number of classes is similar for the different configurations. The results with very few classes are significantly worse than the others. However, beyond a certain number (generally around 5), adding more classes brings almost no improvement. And, with a real learning algorithm, the over-fitting phenomena might even make the quality decrease, as said in section 5.2.2. So the ideal number of classes could be indeed around 5 (the current number used in the system) but as this issue is highly learner-dependent, we cannot say anything definite with these perfect learner-experiments.
The best results are obtained while training on Recall. Then come the KPiA configurations and the worst are the ones with F-Score. This means that the summaries having a high KPiA score are more informative than the ones having a high F-Score score. This is not surprising if we look at the way the measures are computed: F-Score puts more stress on brevity than KPiA and therefore tends to reward less verbosity[10].
For KPiA and F-Score, the differences produced by the choice of the discretization process are seldom significant. For some configurations, Equal width is slightly above and for some others, Equal frequency slightly below but, altogether, the results are too flat to choose one over another. On the contrary, for Recall, whatever the selection process is, we get better results with K-means and Equal width than with Equal frequency. This can be explained with the histograms (presented in figure 4.1 in the previous chapter). The Recall distribution is heavily oriented toward bad summaries. Hence with Equal width and K-means, the best class contains few examples but only very good ones. With Equal frequency, this best class is extended to less good summaries and thus the results are significantly less good.
For KPiA and F-Score, it seems that the selection process does not change much either. The Shortest method is below the other two, which produce similar results. This is not surprising either given that the Shortest method is made to favour brevity and hence Precision, not Recall.
In figure 5.4, we summarize the results for Recall. For the reasons already explained, all the configurations with either the Shortest selection method or the Equal Frequency discretization process, or both, perform less well. There are no significant differences between the others and, in particular, the configuration First found - Equal width, which is computationally the cheapest, seems to perform as well as the best ones. To sum up for informa-

[10]The Recall score does not take brevity into account whatsoever. So the summaries receiving the best Recall scores can also be the longest ones.

tiveness, it seems that we have no interest in looking for a more complicated selection method than First found and that the discretization process Equal width is sufficient too. However, as said in section 5.3.1, we have to remember that the choices of the discretization process, the number of classes and the measure are related to the learner[11].

F-Score

We now report on the F-Score measures, which are, as we already explained, the ones supposedly representative of the quality of the summary, as a trade-off between informativeness and brevity.
The configurations and the way they are presented are the same as for the Recall measure. The statistical significance is a bit lower: 2.75% with a 95% confidence.

[11] For example, some learners do not perform well on imbalanced datasets and thus if there are only 2 classes, bring better results discretizing with Equal frequency than with Equal width or K-means.

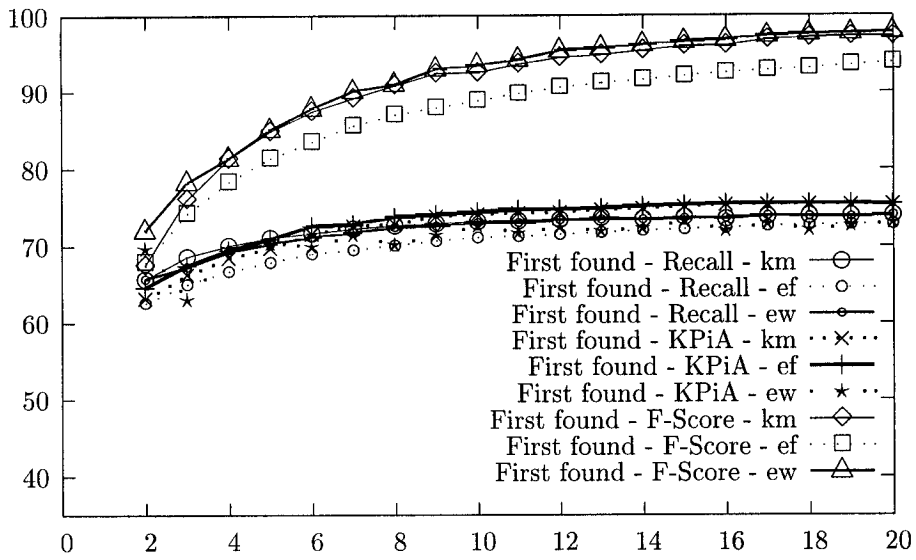

Figure 5.5: F-Score Scores of the First found selection method.

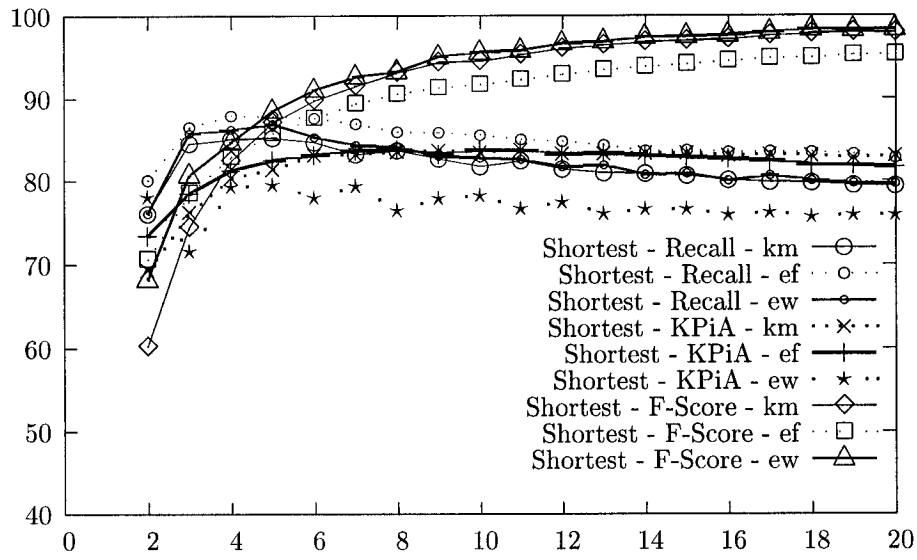

Figure 5.6: F-Score Scores of the Shortest selection method.

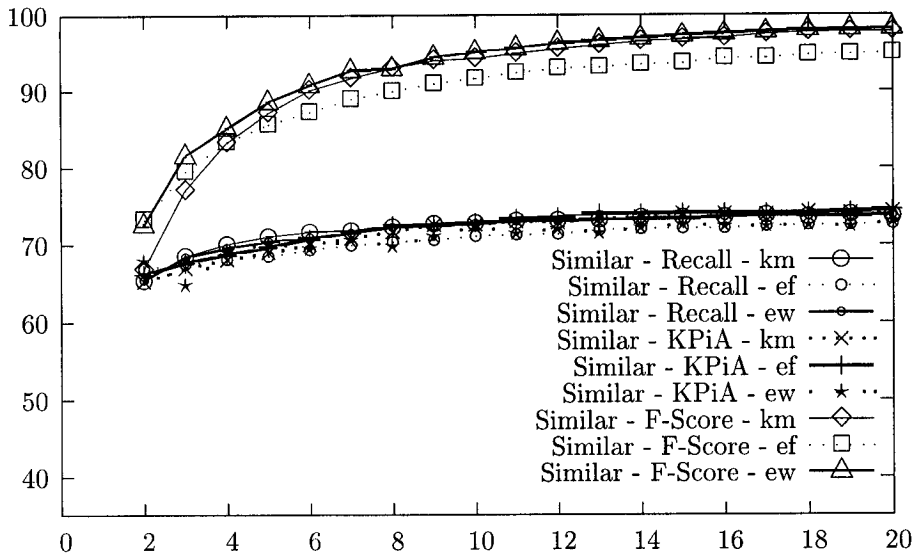

Figure 5.7: F-Score Scores of the Similar selection method.

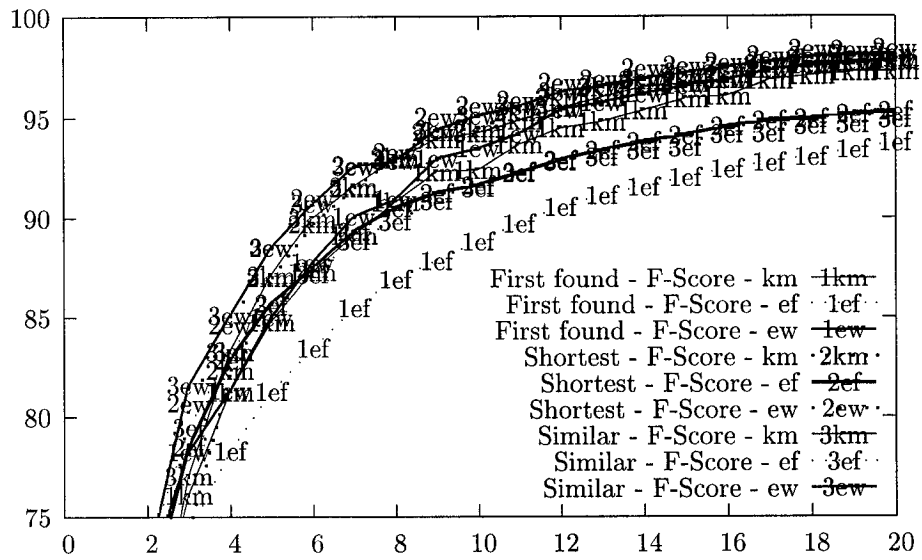

Figure 5.8: F-Score Scores of the best methods.

The easiest figures to interpret are the ones regarding the First found selection method (in figure 5.5) and the Similar selection method (in figure 5.7). The methods using F-Score as the reference measure significantly outperform the others[12], with K-means and Equal Width slightly better than Equal Frequency. The reason is the same as with Recall: the distribution of F-Score is biased toward bad summaries.
The results obtained with the Shortest selection method (in figure 5.6) are very interesting. For a small number of classes, some of the Recall and KPiA configurations even outperform the F-Score method. This validates one of the assumptions we had behind this Shortest method: it is more important in the learning process to focus on informativeness and, once we have the most informative summaries, we can pick the best summaries. And this will be all the more true if the F-Score function appears to be too difficult to learn. For KPiA and Recall, we observe neither the usual difference with Equal Frequency[13] nor the score increasing with the number of classes. It seems that the method of training on informativeness and selecting on length works only when the number of classes is small, that is when there are many examples to choose between. When the number of examples in the best class becomes larger, the selection process becomes less and less important. Beyond 6 classes, the F-Score methods are significantly better than the others. But we have to remember that, when using a real learner, over-fitting is more likely to occur for a large number of classes and therefore, the conclusion "the more, the better" obtained in this experiment might not be valid when we do not skip the learning step.
The best methods are reported in Figure 5.8. Even though they were competitive when the number of classes was small, we did not draw again on this graph the Recall and KPiA curves. This is because Figure 5.6 was clear enough and we just wanted to see if there was a way to discriminate between the best configurations for the largest numbers of classes. As it was the case with Recall, the only significant result is that the Equal Frequency methods are less good than the others.
To conclude, we obtained an interesting result that for a small number of classes (which may be the more realistic configuration with an actual learner), the shortest method used on KPiA and Recall gives promising results. Otherwise, there is no significant difference between all the F-Score methods with Equal Width/K-Means and First found/Similar/Shortest so

[12]Again, this is not surprising since there is no learning step: we are just selecting the examples with the highest F-Score in the input file.
[13]The best configuration even seems to be Recall - Equal Frequency.

we had better choose the less expensive one, which is Equal Width and First Found. But, we do not know whether these preliminary results would still be valid with the real learning component of the system, C5.0. This is what we are going to check in the next section.

5.4 Experiments with the real learner, C5.0

5.4.1 Adequate measure and discretization process

We now consider the current version of CALLISTO, in which the learning component is C5.0 [Quinlan, 1997]. As the previous experiments have shown that changing the selection process was not bringing consistent improvements, we first consider only the versions on the system with the First Found selection method. In section 5.4.2 and 5.4.3, we will try the Shortest and C5.0-confidence selection processes. In section 5.4.4 eventually, we will study the effect of C5.0 cost classification parameter on the learning process. For now, we keep the standard C5.0 algorithm, without cost classification and apply the First Found selection method. We change the discretization process[14], the number of classes and the measure used for training. The last change is especially important since the previous experiments, performed without learning, did not tell us how easy these measures were to learn. We now need to know if the three measures are equally learnable and if, therefore, the curves we get look like the ones we obtained for a perfect learner. The Recall scores are given in figure 5.9 and the F-Score scores are presented in figure 5.10. These results are presented along with the *leading text* baseline introduced in section 4.3.3. In these experiments, and in all the next ones, we note that, as already suggested in chapter 4, the F-Score baseline is much harder that the Recall one. We computed a global statistical significance for all the experiments with C5.0 in this chapter and got:

- 4.31% for Recall.
- 3.51% for F-Score.

[14]Note that for K-means, to save some time in the experiments, the discretization was performed on the whole dataset and not 10 times, after each training/test data division involved in the 10-fold cross validation process. This is not totally correct but we checked that this assumption was justified, comparing the values of the class obtained with global discretization and with discretization on 90% of the data only. It turns out that our dataset is very stable for K-means: the values are the same in 99% of the cases. Similar experiments with Equal Width and Equal Frequency show, on the contrary, that the assumption does not hold with these discretization processes. Therefore, we performed real 10-fold cross discretization for the other methods.

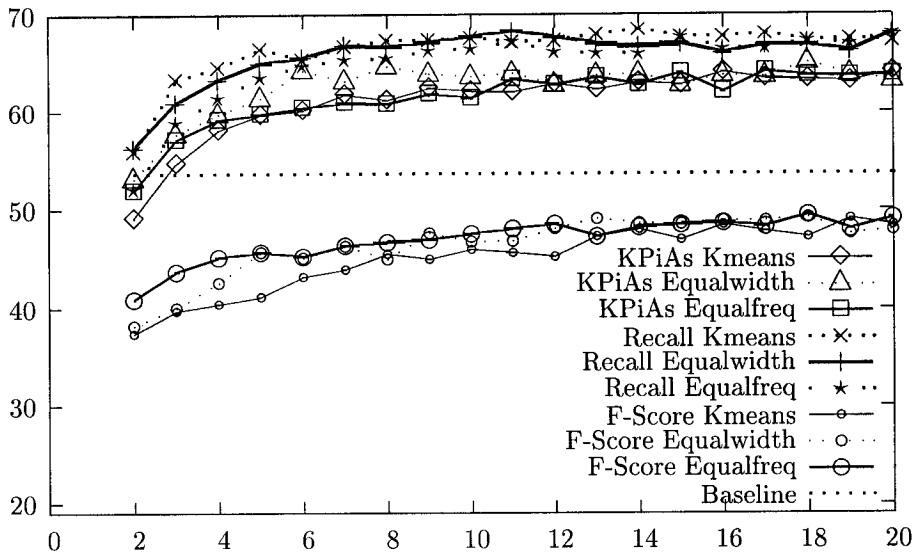

Figure 5.9: Recall Scores of CALLISTO (with First found).

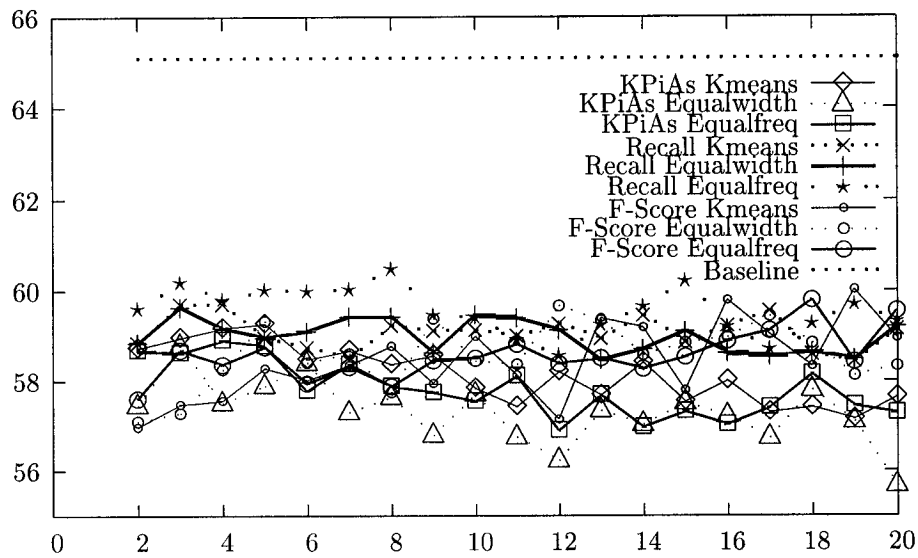

Figure 5.10: F-Score Scores of CALLISTO (with First found).

The Recall curves show that when trained on Recall, CALLISTO performs better than with the other two measures. This logical result shows that C5.0 can actually learn, at least partially, this concept of *informativeness*. The ranking of the different versions is similar to the one we obtained in the case of the perfect learner. The three Recall curves on top, followed by the three KPiA curves and the F-Score ones are the worst.
As for the discretization methods, it seems that, again, there are not many significant differences. Maybe the Equal Frequency method is a bit worse in the case of training on Recall and the Equal Width method a bit better for training on KPiA but all these minor variations are below the statistical significance threshold.
The F-Score curves are very flat. It is disappointing since it means that the summaries selected do not have better F-Score scores than average (57.9%), even when training on F-Score and, therefore, that the function is too difficult to be learned by C5.0. So the different results obtained on Recall just revealed that the dependencies between the text/system vectors and the Recall scores (or informativeness) were learnable but, the main rule actually found is that selecting longer summaries helps to improve Recall. This brute force technique does not help *summarizing* as the uniform results on F-Score show: while selecting more sentences, we are also selecting irrelevant material.
Now, we have seen that applying the shortest selection method with Recall or KPiA can help improve F-Score. The scores obtained are a bit worse than when the perfect learner was trained on F-Score but, since our *real* learner C5.0 does not learn the F-Score concept altogether, this idea is worth trying.

5.4.2 The shortest selection method with C5.0

The goal of these experiments is to see whether the promising result of figure 5.6, which was that combining a measure favouring informativeness (Recall or KPiA) and the Shortest selection method brings good F-Score scores, holds with a real learning algorithm. More informally, we would like to split the task of summarizing into identifying relevant extracts at the learning step and picking the shortest ones at the selection step. As it is the current discretization process and as we saw in the previous subsection that it does not make much of a difference, we only conducted these experiments with the K-means discretization process.
The Recall scores are presented in figures 5.11 and 5.12, along with the scores obtained with the current selection method: First Found.

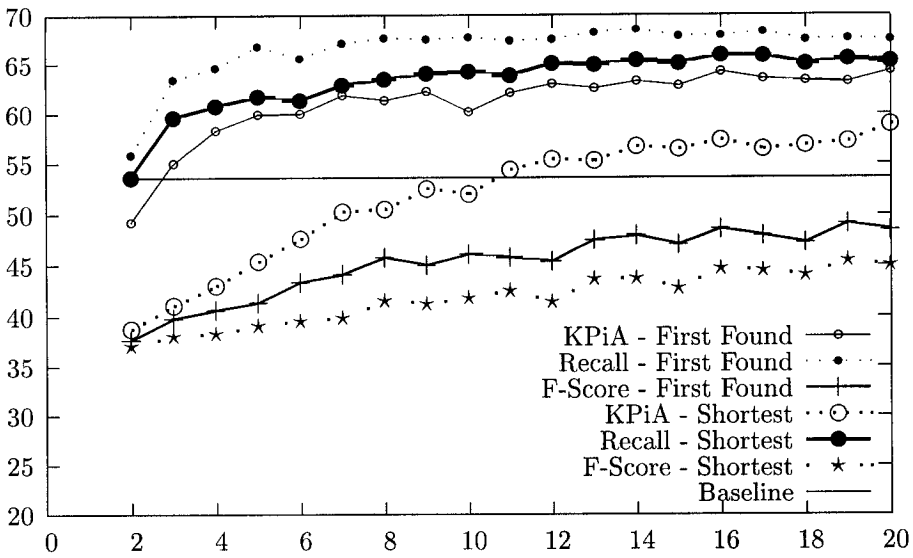

Figure 5.11: Recall Scores of CALLISTO (with K-means).

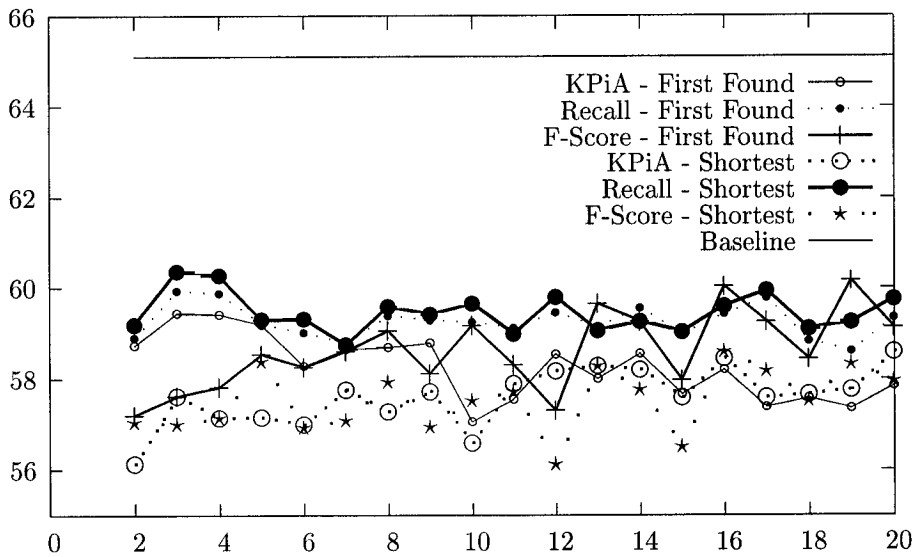

Figure 5.12: F-Score Scores of CALLISTO (with K-means).

Not surprisingly, the scores on Recall are all worse than with the First Found method. If we try to select shorter summaries instead of picking at random, we are likely to lose at least some information. That was already the case in the experiments with a perfect learner. It seems that the losses are not the same according to the measure chosen. Thus, whereas the differences between Recall (respectively F-Score) - First Found and Recall (respectively F-Score) - First Found are just slightly above the significant threshold (around 5%), the KPiA - First Found method outperforms the KPiA - Shortest method by 10% on average. This could indicate that selecting longer summaries is a very important rule learnt when training on KPiA whereas, maybe, C5.0 identifies some alternative rules while learning on Recall.
The results on F-Score show that our intuition was not totally wrong but that, unfortunately, it cannot work as well as when the learner is perfect. Indeed, for the smallest numbers of classes, the best configuration is Recall - Shortest but it does not outperform significantly Recall (or KPiA) - First Found. The conclusion is that the method can be useful but we would need a higher score on Recall[15]. However, It is interesting to note that, as with the perfect learner, the number of classes for which this method works best seems to be 3 or 4. With the other two measures, the method does not work and the Shortest configurations are worse than the First Found ones.

5.4.3 The C5.0-confidence selection method

The idea is now to trust the learner for the selection process as well. For each prediction, C5.0 also outputs a confidence. We use this number to choose the configuration we are going to pick. We select the one C5.0 predicts in the best class with the highest confidence. The results are presented along with the ones corresponding to the First Found selection method in figures 5.13 and 5.14.

[15]When skipping the learning step, the Recall score was around 80%. With C5.0, it is approximately 60%.

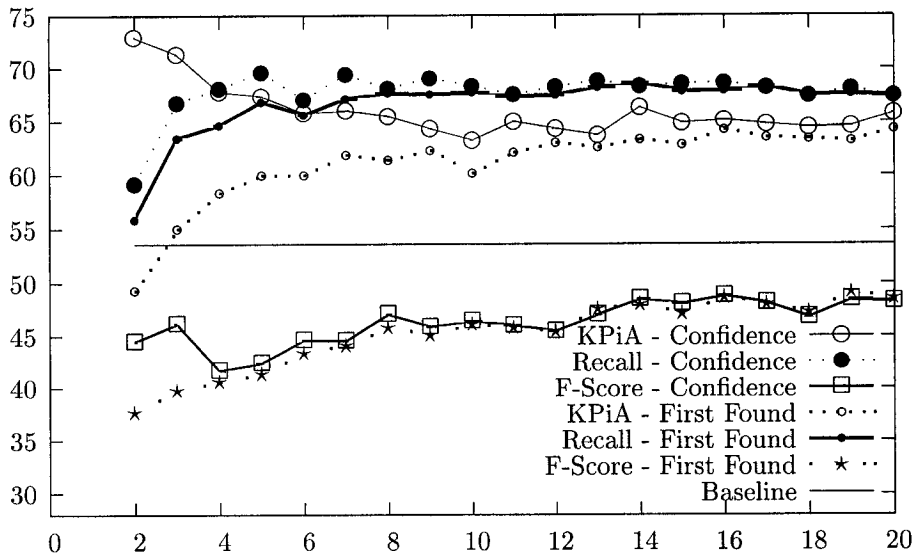

Figure 5.13: Recall Scores with the C5.0-confidence method (K-means).

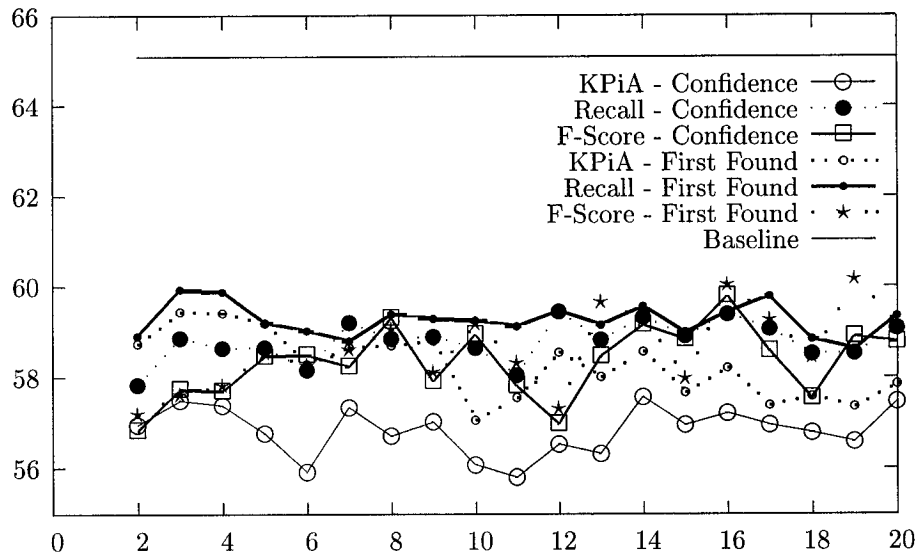

Figure 5.14: F-Score Scores with the C5.0-confidence method (K-means).

The method seems to work very well for KPiA - small number of classes. More generally, on Recall, using the confidence for the selection process seems to be helpful, especially when there are not too many classes[16]. However, these good results on Recall do not lead to any improvement on F-Score. This function seems definitely too difficult for C5.0 and the confidence rates it outputs do not help here. Altogether, there does not seem to be any selection method better than the others to improve F-Score. So keeping first found seems the most logical choice since it is the cheapest algorithm.

5.4.4 C5.0 with cost classification

As we select examples from the best class only, it is more important that the learner favours it in the learning process, even though it implies a larger error rate on the other classes. Now, C5.0 has a cost parameter by which we can specify that some errors (in our case the ones regarding the best class) are more important to avoid than others. We will now try to set the cost parameter so that missing an example from the best class would be k times more serious than for the other classes. To determine the best value for k, we ran a few experiments for k from 1 to 8 with all three different measures, 5-means discretization process (as in the current version of CALLISTO) and First found selection method. The results are presented in figure 5.15.

[16]For the largest numbers, we know that the selection process matters less, since there will be very few examples in the best classes.

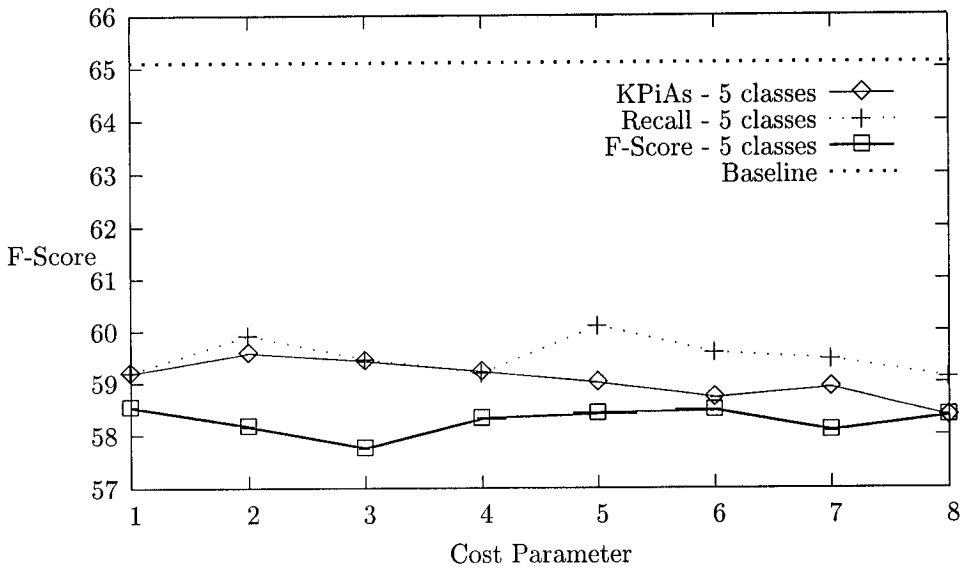

Figure 5.15: F-Score Scores with cost classification (5-means).

There does not seem to be significant differences in the overall performance for the different values of k we tried. Therefore, to investigate the influence of this parameter in more detail, we set it to 5 (a value which seems to give a good performance for Recall and F-Score) and conducted the same experiments with the 3 measures and changing the number of classes from 2 to 20 (on the horizontal axis), as in the previous experiments.
The results are given in figures 5.16 for Recall and 5.17 for F-Score. They show that there is in fact no significant differences between the regular performance of C5.0 (cost 1) and the one with cost classification. Based on these disappointing results, we decided that it was not worth investigating further the use of C5.0 cost parameter.

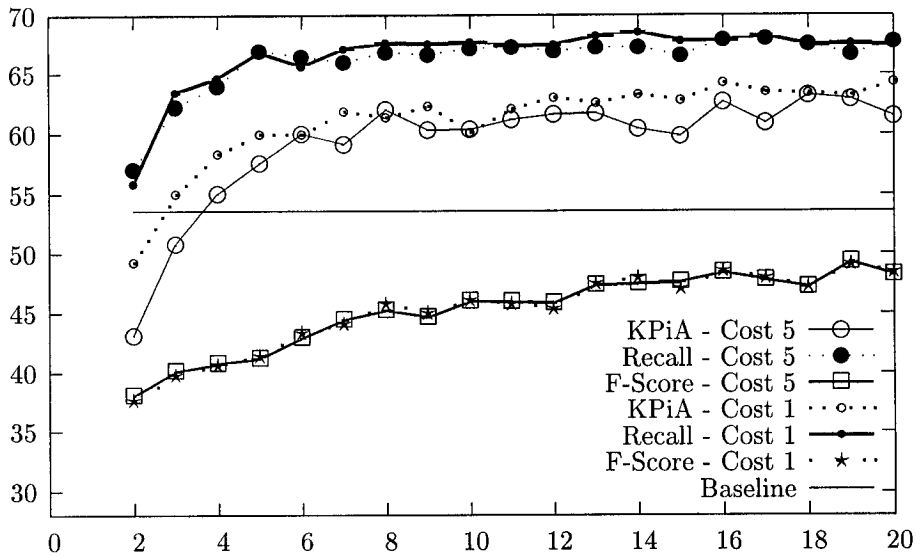

Figure 5.16: Recall Scores with cost classification (K-means).

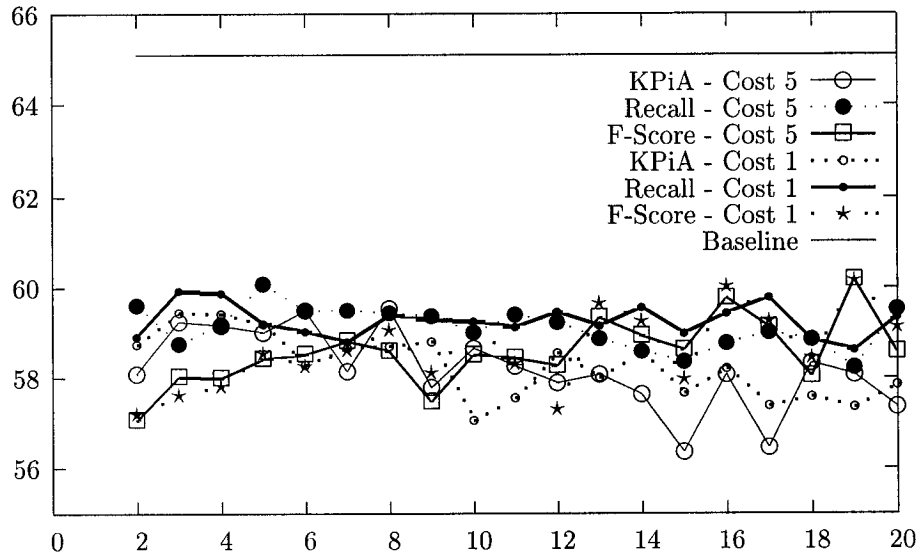

Figure 5.17: F-Score Scores with cost classification (K-means).

5.5 Conclusion

In this chapter, we have investigated different measures (KPiA, Recall, F-Score) with various discretization processes (K-means, Equal Width, Equal Frequency from 2 to 20 classes) and several selection methods (First Found, Shortest, Similar, C5.0-confidence). First, we have tried these modifications on an *ideal* version of the system without the learning component and then with the real CALLISTO including C5.0.
One first important conclusion is that the choice of the measure is highly learner-dependent and the experiments with a perfect learner did not help at all in this respect. However, they helped to confirm the intuition that learning Recall and applying the Shortest selection method could help to get good results on F-Score.
Other than that, it seems that the function mapping the text/systems vectors to their respective F-Score score is too difficult to learn for C5.0 whereas it manages to learn Recall more accurately. Changing the discretization process did not have much influence on the final results. On the real system, the Shortest selection method did not bring results as good as expected. We assume that it is because C5.0 does not have scores high enough in Recall. The use of cost classification with C5.0 did not help either, whichever measure was chosen. Now, we are going to try other learning algorithms to see if they perform better.

Chapter 6

Selecting an Adequate Learner

6.1 Introduction

The learning component is a crucial part of CALLISTO since it is responsible for predicting the right configuration to apply to each text. Therefore, the main idea underlying the system, which is that the setting to choose depends on the input text, is valuable only if we can efficiently take advantage of knowledge in the training data to predict choices on further texts.
In this chapter, we try different approaches from a wide range of Machine Learning algorithms. C5.0 [Quinlan, 1997] was chosen as a fast, efficient state-of-the-art program but it sometimes happens that others work better for a certain application and it could be the case for us. It is important to mention at that point (and this will be studied more deeply in the Naive Bayes case) that examining the error rate of the algorithm[1] is not enough to decide if it is applicable or not. Indeed we are only interested in a small part of the data: the examples belonging to the very best class, one of which representing the configuration we will eventually apply to the text. Therefore, a large number of examples could be misclassified without any consequences

[1]The error rate of a learning algorithm on a particular data is the number of times the algorithm predicts a class different from the actual one out of the total number of examples. Typically, the error rate of C5.0 on our DUC 2002 dataset is 30%.

for the final result[2]. All in all, only two types of errors are really hurtful: the *false best* (labeled as belonging to the best class whereas they are not) and the *missed best* (belonging to the best class but labeled otherwise). It is not clear whether the global error rate is approximately the same as the error rate on those two categories and therefore if the global error rate of the learning algorithm can be seen as representative of the quality of the summaries. Our framework will enable us to check this assumption.
First, we will consider applying the AdaBoost algorithm [Freund and Schapire, 1996] in C5.0. Then we will discuss the use of other classical learning algorithms: RIPPER [Cohen, 1995], Neural Networks [Zeller *et al.*, 1995], Naive Bayes [John and Langley, 1995], Nearest Neighbour (see Chapter 8 in [Mitchell, 1997]) and Support Vector Machine [Joachims, 1999]. These classification algorithms have been chosen as a representative state-of-the-art sample, which Mitchell deems "the key algorithms [...] that form the core of Machine Learning" [Mitchell, ibid.], page xv. Finally, the use of classification raises a number of issues such as the ones presented in the previous chapter: discretization, number of classes, selection method. We thought that trying regression algorithms could be a good idea as well and this is what we will talk about in the last section.

6.2 AdaBoost

Boosting is a technique of iterative learning which can be applied to any learning algorithm to improve its accuracy. The algorithm is described in [Freund and Schapire, 1996]. The idea is to modify the distribution of the examples in the dataset given to the learner to focus on the hardest to learn. First, the algorithm is run normally on the dataset. After this first iteration, the distribution of examples is changed, the weights being larger for the examples the learner did not classify correctly and smaller for the ones for which it was right. Thus, in the second iteration, the examples which were not properly dealt with in the first place are seen more often and there is a chance that the hypothesis output in the end classifies them better than the first one did. After a given number T of iterations (set arbitrarily), we therefore have T hypotheses output by the learning algorithm. The final hypothesis for a given example is a weighted vote over those T hypotheses, the weight of one hypothesis depending on its global accuracy. The better

[2]Consider for example that a combination text + configuration is labeled *medium* whereas it should have been labeled *bad*. This does not change anything for the remaining steps since we will only select the configurations among the *best* class.

a hypothesis is, the more important it is in the final vote.
The experiments with AdaBoost are very time-consuming, ten times more than the regular ones so we decided to limit the number of classes. As we found that the shortest selection process could be interesting in some cases in chapter 5, we decided to try it and we kept the original first found selection process too. Discretization is still done with K-means. The results are presented along with the ones obtained without Boosting in Figures 6.1 and 6.2.

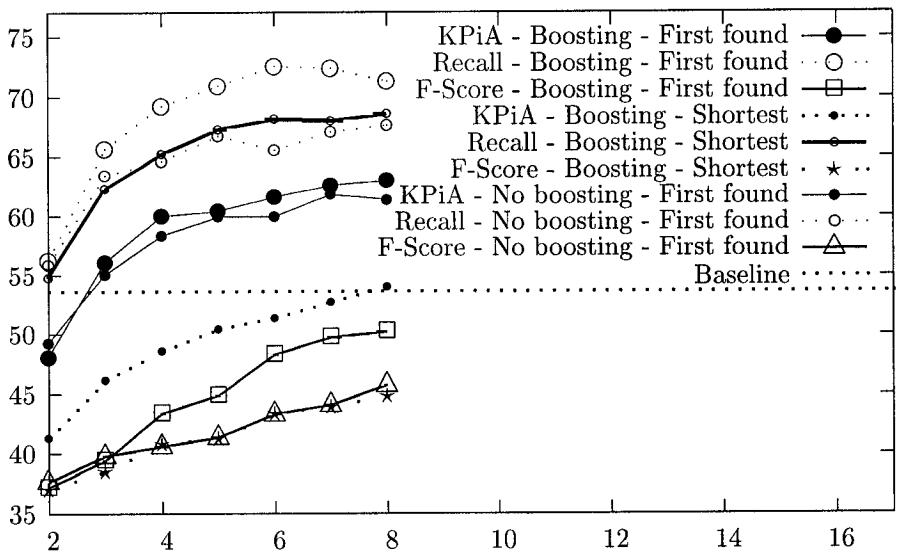

Figure 6.1: Recall scores with boosting method (K-means).

The results we obtained confirm two assumptions:

- AdaBoost does improve consistently the results C5.0 gets.
- Training with Recall and KPiA, even with Adaboost, does not lead to good F-Score scores, even with the shortest selection method and Adaboost-C5.0 is not more efficient when trained on F-Score than C5.0 alone.

Finally, the configuration with Recall-AdaBoost produces significantly better results on Recall, meaning that the learning step is more successful. However, we could not take advantage of this improvement to get better F-Score scores as well.

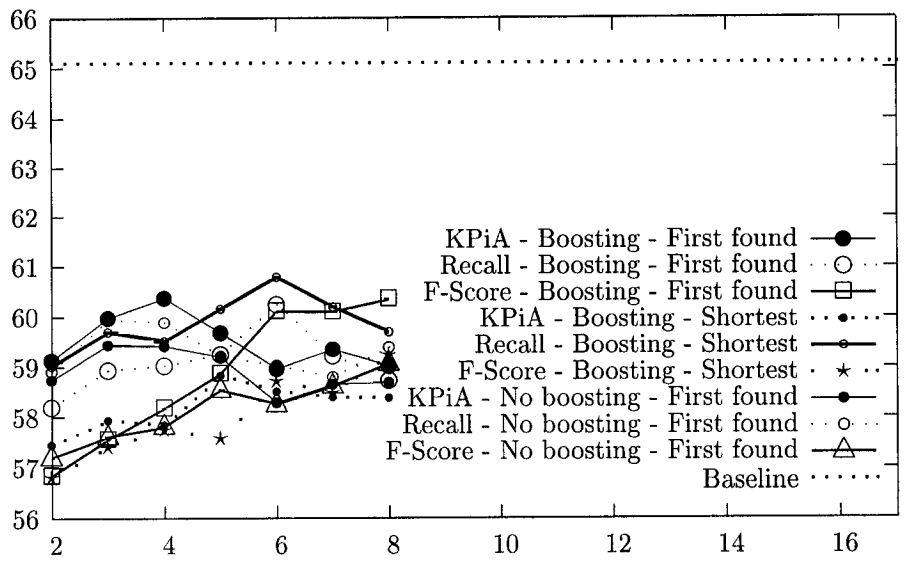

Figure 6.2: F-Score scores with boosting method (K-means).

The increase in Recall is less obvious when training on KPiA so improving the learning on KPiA is not totally equivalent to improving informativeness. Learning on F-Score still seems very useless and, more generally, all the results with F-Score measures bring the same disappointing conclusion: we are actually not learning summarization but just selecting more information with longer summaries. Average lengths we sampled on DUC dataset[3] reinforced this point:

- KPiA, 5 classes (base system): 7.42 sentences per extract
- Recall, 14 classes: 9.26 sentences per extract
- KPiA, Boosting, 20 classes: 8.74 sentences per extract
- Recall, Boosting, 6 classes: 10.01 sentences per extract

The problem remains: how can we improve *informativeness* and *brevity* at the same time?

[3]The average length of a text in this set is 34 sentences.

6.3 Neural Networks

Giving a detailed introduction to Neural Networks is far beyond the scope of this thesis. We will however present briefly the basic ideas involved in Neural Networks in general and in the implementation Stuttgart Neural Networks Simulator (SNNS) we used. For more information, see the on-line manual [Zeller *et al.*, 1995].
A Neural Network consists of *units*, linked with one another by weighted *connections*. Each unit has one or more input connections and an output connection. A function of the input determines what has to be sent to the output link. In the simplest model, the output is binary and a unit outputs either 0 or 1. In the latter case, we say the unit is *activated*. There are usually three layers of units:

- The input units receives as input the values of the attributes (in the classical Machine Learning meaning). That is why there is usually one unit per attribute.

- The hidden units constitute one or more layers of units. They receive data from the input units and feed the output unit. In the simplest classification tasks, those intermediate units may not be needed.

- The output unit corresponds to the class (still in the usual Machine Learning sense).

The goal in training such a network is to determine the weights in the activation function of each units which enable us to minimize the global error rate. An algorithm to do so is standard backpropagation [Zeller *et al.*, 1995] and that is the one we have used to apply SNNS to our dataset.
We have chosen SNNS because it is reasonably fast and gives good results on some classical Machine Learning problems[4]. However, we tried several sets of parameters in standard backpropagation (for the learning rate, the number of hidden units and the number of training periods) and none of them seemed to bring satisfactory results. The error rate was staying around 90% and even more troubling the lack of any improvement with learning. Based on this evidence, we concluded that this algorithm was not the best for our problem and decided to investigate others.

[4] In particular, Neural Networks have proven to be efficient for some pattern recognition problems.

6.4 RIPPER

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Cohen, 1995] uses techniques quite similar to the ones underlying C5.0 [Quinlan, 1997]. As the latter works well on our dataset, we assumed that it could be true as well for the former, which has, besides, been shown to do well on some text-based applications [Nastase and Szpakowicz, 2003].
The emphasis in the rule-learning algorithm RIPPER is on the Pruning step, that is, a set of rules is learned first on the *growing* data and then pruned from the *pruning data*[5]. Similarly to C5.0 algorithm, the rules are grown to maximize the *information gain* criterion. More precisely, the module to grow rules "begins with an empty conjunction of conditions, and considers adding to this any condition of the form $A_n = v$, $A_c \leq \theta$, or $A_c \geq \theta$, where A_n is a nominal attribute and v is a legal value for A_n, or A_c is a continuous variable and θ is some value for A_c that occurs in the training data." ([Cohen, 1995], p. 2). The information gain criterion used to select rules comes from the first-order learning algorithm FOIL [Quinlan and Cameron, 1993]. In the binary case, for condition C, if S is the set of training examples considered and if, for a set T, we denote T_+ the set of positive examples in T and $T(C)$ the number of examples in T for which C is true:

$$gain(C) = \frac{|T(C)|}{|T|}\left(-log_2\frac{|T_+|}{|T|} + log_2\frac{|T(C)_+|}{|T(C)|}\right)$$

After a rule is produced, it is pruned to cover as many examples as possible in the pruning data. The addition and deletion of rules in the initialization and optimization step are dictated by a *Minimum Description Length* principle, that is to minimize the number of bits needed to encode the rule set. See [Cohen, 1995] for further details.
Unfortunately, we have not been able to obtain any conclusive results with RIPPER on our dataset. The computation time was intractable and we did not have a chance to compare its performance to C5.0's. On an Ultra 10 Station at 300 MHz with a 128 Mb memory, the times needed to process datasets of various sizes were the following ones:

- 1000 examples: 8.30 seconds.
- 2000 examples: 21.69 seconds.

[5]The growing and pruning sets are usually randomly extracted from the training data. In his experiments, Cohen mentions the ratio of 2/3 of the training data to grow the rules and 1/3 to prune them.

- 5000 examples: 110.07 seconds.
- 30000 examples: 1611.39 seconds, that is approximately 27 minutes.
- 100000 examples: 8788.09 seconds, that is more than 2 hours and a half.

We would probably need more than 8 hours for the full dataset, which is three times larger. This illustrates the fact that not every learning algorithm works fine for CALLISTO and that the size of the dataset obliges us to limit our choices to the fastest ones.

6.5 Naive Bayes

6.5.1 Description of the algorithm

We have already discussed the Naive Bayes learning algorithm in Chapter 2, while talking about [Kupiec *et al.*, 1995]. A good description of the Naive Bayesian classifier can be found in [John and Langley, 1995]. This algorithm is very simple and outputs the most probable hypothesis (*Maximum A posteriori Hypothesis*) when "the attribute values are conditionally independent, given the classification of the instance" ([Mitchell, 1997], page 197). This assumption is almost never respected on real datasets. However, even in those cases, the Naive Bayesian classifier still outperforms other classical algorithms such as Neural Networks or Decision Trees. That is why we decided to try it on our dataset.
The main idea is to use the Bayes formula[6] to estimate the probability with which an unseen example belongs to a given class C_i, from its values on each feature F_j, the probabilities of these values being estimated from the training data:

$$P(C_i|F_1, F_2, \ldots, F_k) = \frac{\Pi_{j=1}^{k} P(F_j|C_i)P(C_i)}{\Pi_{j=1}^{k} P(F_j)}$$

Now, to determine to which class a given example belongs we have to compute this formula for every class. The product $\Pi_{j=1}^{k} P(F_j)$ is not computed since it is the same no matter what C_i is. Therefore the expression to evaluate is $\Pi_{j=1}^{k} P(F_j|C_i)P(C_i)$ and this is computed from the training data.

[6] Whenever $P(B) \neq 0$,

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

$P(C_i)$ is simply the number of training examples belonging to class C_i out of the total number of examples. If the attribute corresponding to F_j is discrete, $P(F_j|C_i)$ is the number of examples of class C_i with value F_j for that attribute. If it is continuous, it is generally assumed to be gaussian[7] and the value of the probability is then:

$$P(F = x|C_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

μ and σ are the respective average and standard deviation for that attribute of the population of examples of class C_i and this is calculated from the training data.
Meanwhile, we have implemented this basic Naive Bayes algorithm and tested it within the framework. An interesting property of this classifier is that it does not just predict a class, it also outputs a confidence in its prediction (the probability). This is particularly important for us regarding the selection process. It gives a classifier-dependent method, similar to C5.0-confidence, to pick a configuration among the best class instead of taking the first, shortest or most similar to the others. We can choose the configuration predicted with the highest confidence. This is the selection method we have applied in the experiments below.

6.5.2 Experiments

The discretization process and the measure chosen are highly dependent on the learner. Therefore, we have to test all of them again whatever the results obtained with C5.0 were. That is what we did. We report the results separately for each measure for the sake of readability. The Figures 6.3, 6.4 and 6.5 show the results for Recall and the Figures 6.6, 6.7 and 6.8 show the F-Score scores. The horizontal axis is, as usual, the number of classes. The statistical significance computed for these experiments is 3.42% for Recall and 2.59% for F-Score.

[7]This strong assumption of normality of continuous attributes within each class is not necessary if one proceeds to discretization of the continuous attributes prior to learning. But then, the problem is to choose an adequate discretization method [Dougherty *et al.*, 1995], reasonably cheap in computation time. The ones we tried, such as entropy-based techniques [Fayyad and Irani, 1993], did not work well for our dataset (too many cut-off points) and therefore we decided to stick to the gaussian hypothesis.

Recall

The results are very different from the ones we obtained with C5.0, much less flat. The Recall values are on the whole very large, the best being slightly above what we had with boosting. The best overall configuration for informativeness is Recall discretized with Equal Frequency. More generally, it seems that the Equal Frequency discretization which was not good for C5.0 seems more appropriate for Naive Bayes. It makes the classification depend only on the term $\Pi_{j=1}^{k} P(F_j|C_i)$ since $P(C_i)$ is the same whatever the class is.

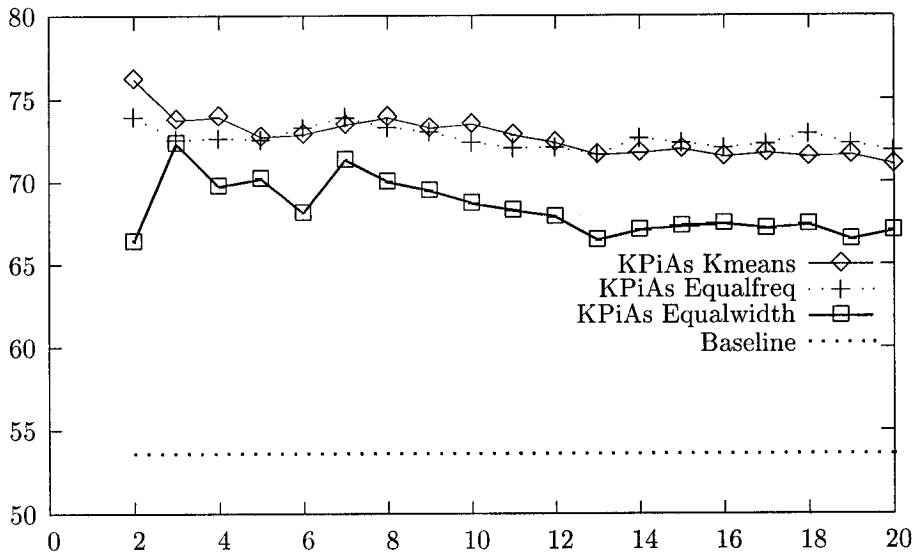

Figure 6.3: Recall Scores with Kpia measure.

For KPiA as the training measure, K-means performs as well as Equal Frequency, both being significantly better than Equal Width. For Recall, on the contrary, K-means is approximately equivalent to Equal Width, below Equal Frequency. For F-Score, the three discretization processes are more or less equivalent, except for the first three classes where Equal Frequency is much better. This may be due to the fact that with K-means and Equal Width, the problem is very unbalanced toward the *bad* classes. Note that, with F-Score in particular, the Recall scores are much better than what they were with C5.0 meaning that training on F-Score with Naive Bayes helps to improve the informativeness. This is a good sign regarding the *learnability* of F-Score.

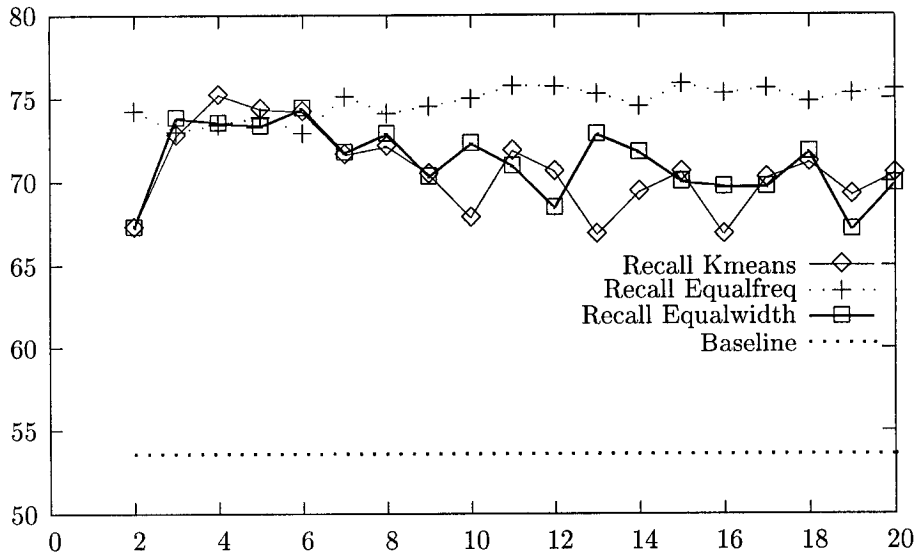

Figure 6.4: Recall Scores with Recall measure.

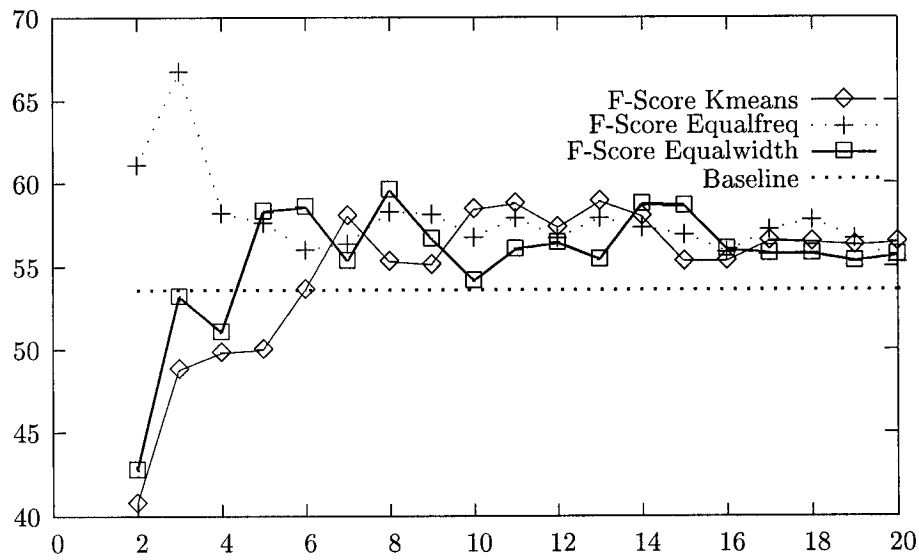

Figure 6.5: Recall Scores with F-Score measure.

F-Score

The results on F-Score are still below the baseline but better than the ones we get with C5.0. For KPiA, there is a contrast between the configurations corresponding to K-means and Equal Frequency and below 10 classes and the others. The Equal Width discretization process definitely has to be avoided with Naive Bayes but why a small number of classes is more desirable for the other two methods is not clear. It may be due to overfitting for large number of classes, a phenomenon that we have not observed with C5.0.

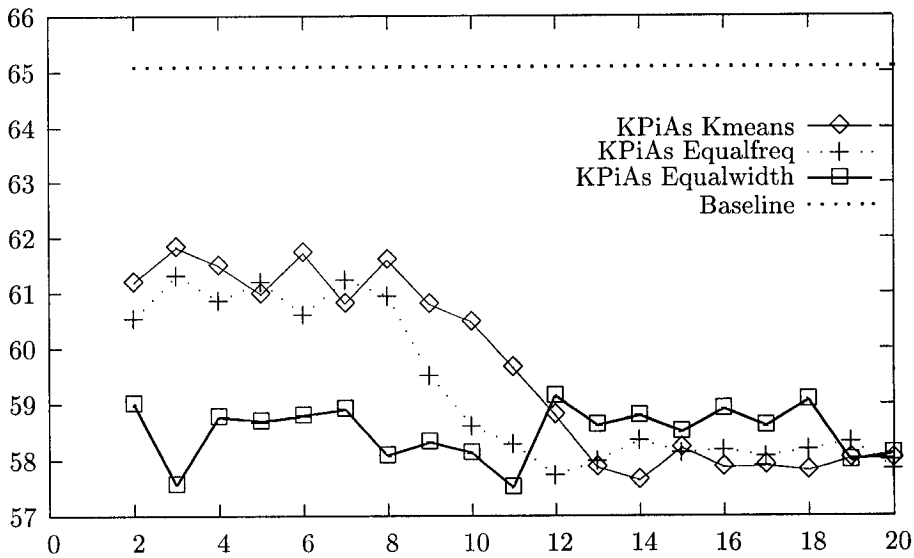

Figure 6.6: F-Score Scores with Kpia measure.

When training with Recall, the results are also better than with C5.0. For a small number of classes, there is no significant difference between the discretization methods but on the whole Equal Frequency seems to bring slightly better results, also outperforming all the KPiA methods and confirming the good Recall scores previously listed.
On F-Score, we get the best scores we had so far. It means that the F-Score concept is indeed learnable though obviously harder than Recall. The fact that the configurations with Equal Frequency and small number of classes are significantly better than the others is confirmed here, our best configuration being Naive Bayes trained on F-Score with 4 classes equally probable.

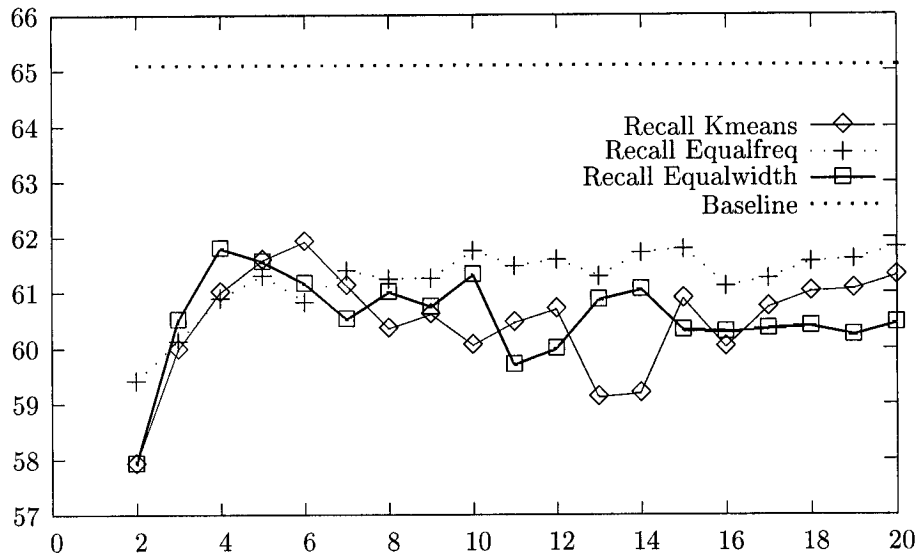

Figure 6.7: F-Score Scores with Recall measure.

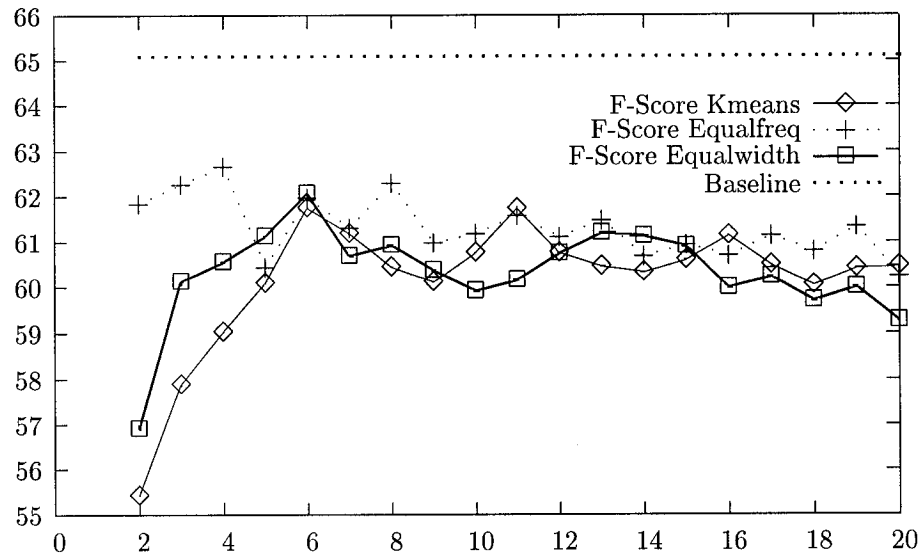

Figure 6.8: F-Score Scores with F-Score measure.

6.5.3 A bad learner but a good predictor?

The reason why Naive Bayes gives in the end much better results than C5.0 is not clear. The fact that we have a reliable selection method, being wholly part of the learning process may be one reason[8]. The previous results indeed showed that we had not found an optimal selection method yet, shortest, similar and C5.0-confidence not being significantly better than random, and that there was room for improvement at that point. To check this assumption, we tried Naive Bayes with F-Score and Equal Frequency with another selection method. The results in figures 6.9 and 6.10 show indeed that the performance drops significantly when we do not select the best configuration based on the probability (hence the title *most probable* in the figures).

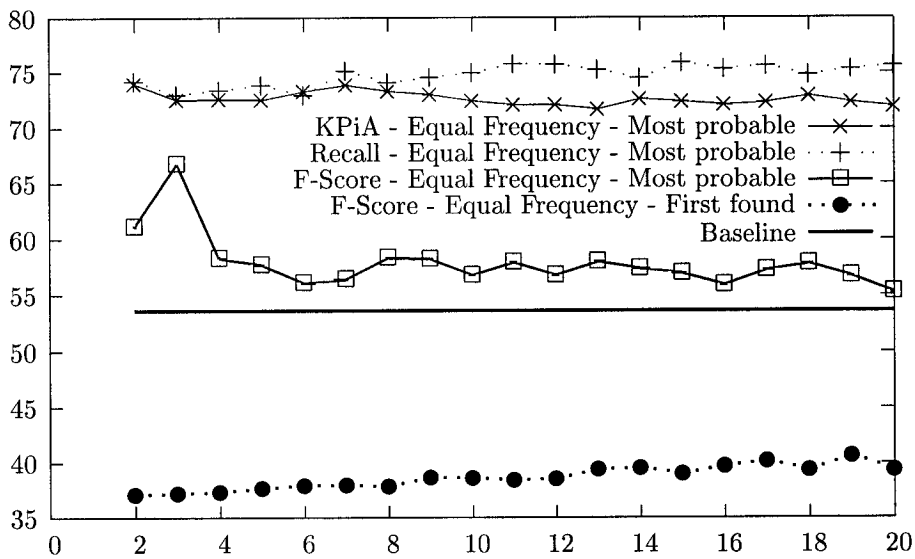

Figure 6.9: Comparison of Recall Scores with another selection method.

The fact that Naive Bayes helps produce better summaries than C5.0 is all the more surprising when we look at the accuracy of the classifiers on our dataset in 10-fold cross validation: C5.0 seems better than Naive Bayes. However, as we said at the beginning of this chapter, we are only interested in a small subset of the predictions and, with this selection method, we are only interested in the predictions Naive Bayes makes with high confidence.

[8]Even though C5.0-confidence was also classifier-dependent, the results of the experiments presented in chapter 5 seriously question its reliability.

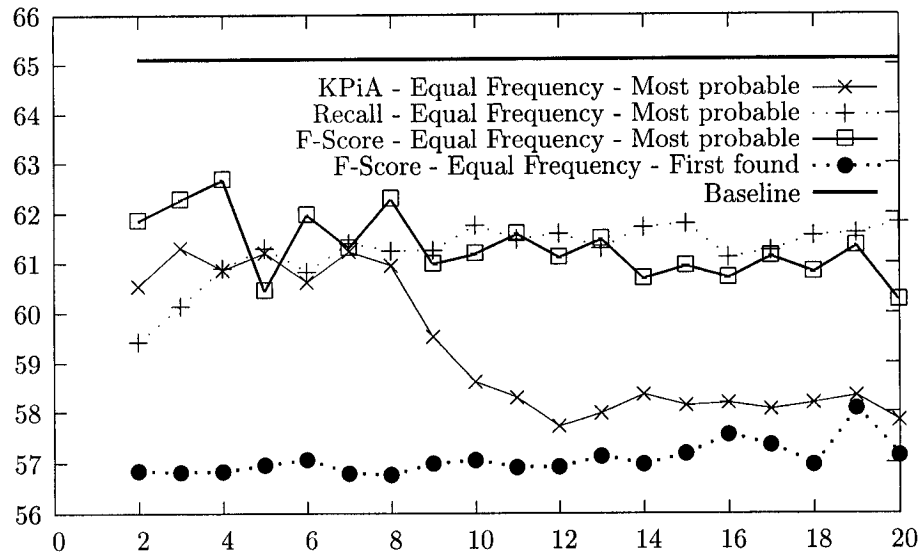

Figure 6.10: Comparison of F-Score Scores with another selection method.

It is possible that on those, Naive Bayes outperforms C5.0.
This hypothesis is in part confirmed by experiments as shown in figure 6.11. We compare the global error rates of the C5.0 (trained on F-Score, discretized with K-means) and Naive Bayes (trained on F-Score, discretized with Equal Frequency) classifiers with the error rates on the configurations actually selected (one per text). In the former case, we check the accuracy of the learners on *every* example in the training data[9] whereas in the latter case, we disregard all the configurations representing the configurations we did not choose to generate summaries[10]. This shows that, whereas the selection process for Naive Bayes helps improve the error rate and select the most reliable predictions, the effect is totally opposite with C5.0, whichever selection method (First Found or Confidence) is chosen. The accuracy becomes worse then, which means that C5.0 is worse on the best class, the one we are interested in, than on the others.
However, in particular for small numbers of classes, the error rate of C5.0 on the selected configurations is still better than the one of Naive Bayes,

[9]Therefore, we count 1 for every mistake and 0 for every correct prediction and divide the result by the number of summaries: 334210.

[10]Similarly, we count 1 for every mistake and 0 for every correct prediction but, in the end, we divide by 1120.

whereas we know that the summaries produced by the latter get larger F-Score scores. We believe that this is related to the selection process and, even when the prediction is wrong, the summary eventually selected may be better than with C5.0. Perhaps, the predictions getting higher confidence with Naive Bayes are good in some way whereas this minimal quality is not always guaranteed by C5.0. Besides, the error rates of C5.0 - F-Score - K-means are not totally meaningful. For example, for 2 classes, the classes are extremely unbalanced and the error rate is almost equivalent to the one we would get when predicting always the most probable class (bad). Therefore, the very good 90% accuracy does not mean that C5.0 is efficiently learning the F-Score function.
This analysis demonstrates the usefulness of our framework. Based just on the global error rates, we would have preferred C5.0, without necessarily having the idea to investigate further and examining the error rate on selection. And we would have missed the fact that the configurations with Naive Bayes are better than the ones with C5.0. With the framework however, we are able to judge configurations of the system on the overall qualities of the summary, i.e. on the final output of the system, and not on an intermediate measure inside CALLISTO, such as the error rate of the learning component. Thus the framework enabled us to make decision based on what we are really interested in (producing better summaries) and not on other considerations (reducing the error rate of the learning algorithm) for which we do not know how they impact on the final performance of the system.
However, to conclude with Naive Bayes, we have to say that the accuracy is a bit better on the configurations selected than on the whole but the absolute improvement is not great. And, on the best configuration F-Score - Equal Frequency - 4 classes, the error rate is surprisingly bad: 55%. Therefore, we believe that there is still a lot of room for improvement using the framework.

6.6 Nearest Neighbour

The K-Nearest Neighbour learning algorithm is described in [Mitchell, 1997], pp. 230-236. The basic idea is simple: when faced with a new example, try to look for the most similar example(s) seen so far and classify the new one the same way. When working with 1-Nearest Neighbour, we just look for the closest one. When $K > 1$, we proceed to a majority vote to determine the class.
Now, what does "closest" mean here? It implies a metric on the space of

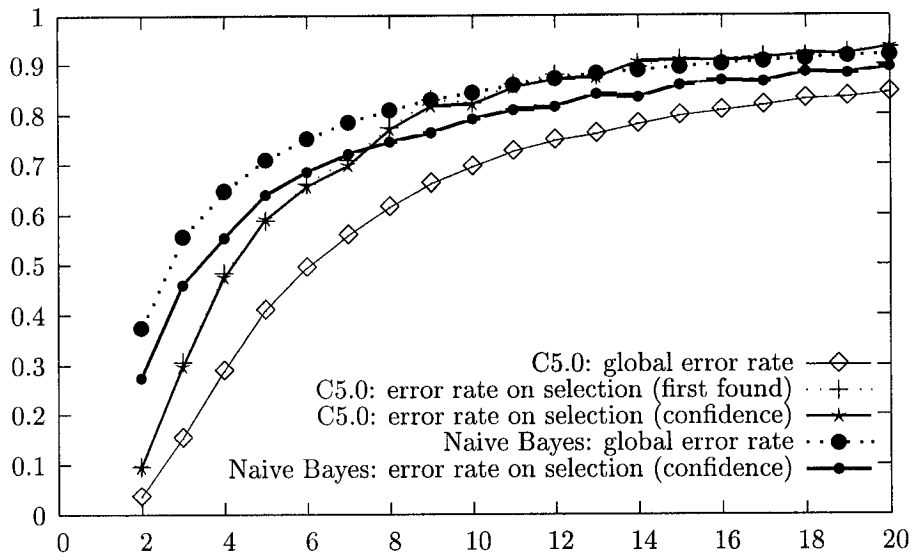

Figure 6.11: Accuracies of Naive Bayes and C5.0 trained on F-Score.

examples. Usually, the Euclidian distance is used. In our case, it will imply a prior normalization of the data to have every continuous attribute between 0 and 1, weighing the same in the final sum. Otherwise, we could have one attribute, and one only, for instance, in our case, the number of characters, counting in the distance, since its range is much larger than the others'[11]. We will apply the version of K-Nearest Neighbour to continuous-valued class to avoid the problems with discretization and selection methods. Let us denote x an example, $x_1, \ldots, x_K$ its K nearest neighbours (the ones with the smallest distances to x) and $d(x, x_j)$ ($j \in < 1, k >$) the classical Euclidian distance from x to x_j. The number $C(x)$ corresponding to the class of x is predicted by a weighted sum of the numbers $C(x_1), \ldots, C(x_K)$ (continuous classes), the weights corresponding to the proximity of the given neighbour

[11]Note that this problem occurs with many other learning algorithms, as soon as we have to compare numeric values of different attributes. That's why in general the normalization step is an implicit part of the pre-processing for learning, as it is the case for C5.0 for instance.

to x, as follows[12]:

$$C(x) = \begin{cases} C(x_j) & \text{if } \exists j \in < 1, k >, d(x, x_j) = 0; \\ \frac{\sum_{i=1}^{k} \frac{C(x_i)}{d(x,x_j)^2}}{\sum_{i=1}^{k} \frac{1}{d(x,x_j)^2}} & \text{otherwise.} \end{cases}$$

As usual, we produce 300 examples per text and choose the one receiving the largest predicted value for C. Ties are much more unlikely than in previous cases since the values of the class are continuous.
Unfortunately, the computation is practically intractable. There are $1+2+\ldots+334210$ distances to compute approximately (a bit less than that because examples from the same fold are not compared), that is around 55 billion! On an Ultra 10 Station at 300 MHz with a 128 Mb memory, computing 334210 distances takes 10 minutes so the total running time would be above 3 computation years... This problem seems inherent to the use of this algorithm on our dataset since we tried other well-known and more elaborate implementations such as the IBL package [Aha and Kibler, 1989] and they did not produce results in a reasonable amount of time. Since the bottleneck of the process is on the calculation of the distances between examples, there does not seem to be other solutions to this problem than using fewer examples.
Besides that, another problem, which often occurs with this algorithm, was foreseeable: *the curse of dimensionality* [Mitchell, 1997]. There is likely to be many irrelevant or, at best, redundant features in the text vector. So the text vector may weigh too much in comparison to the system vector (which is the one we are more interested in, in the end, since we want to predict an adequate configuration for every text)[13]. Therefore, leaving alone computation problems, this learning algorithm would probably not have been very efficient before getting significant results in selecting relevant features from the text vector (section 7.1).

6.7 Support Vector Machines

Once again, discussing in detail the framework and techniques behind Support Vector Machines is impossible in a small paragraph. We will therefore

[12]Note that there cannot be two examples j and k such that $d(x, x_j) = d(x, x_k) = 0$ since it would mean $d(x_j, x_k) = 0$ and therefore, the text and system vectors would be exactly the same. This would be inconsistent with the implemented procedure since each system vector is applied once and only once to every text.

[13]It is possible to fix this problem by adjusting weights on every attribute. The problem then would be to find the adequate set of weights yielding the best performance.

just state the main idea of Linear Support Vector Machines, which is to find hyperplanes best partitioning the data into distinct classes [Burges, 1998]. We follow the notation used in [Burges, ibid.]. If the number of attributes is n, then each example x can be seen as a vector $(x_1, \ldots, x_n)$ in $\mathbf{R}^n$ with a label y, the value of its class. We will consider only the binary case where $y \in \{-1, +1\}$. If the data is linearly separable, there exists some hyperplane H separating the positive from the negative examples, which is called a *separating hyperplane* and the equation of which is

$$\sum_{i=1}^{n} w_i x_i + b = 0 \text{ where } b \in \mathbf{R} \text{ and } \forall i \in < 1, n >, w_i \in \mathbf{R}$$

Then, we can assume that all the training data verify either $\sum_{i=1}^{n} w_i x_i + b \geq +1$ for the positive examples and $\sum_{i=1}^{n} w_i x_i + b \leq -1$ for the negative examples. Then it defines two particular separating hyperplanes: the hyperplane closest to the positive examples H_1, the equation of which is $\sum_{i=1}^{n} w_i x_i + b = +1$ and the hyperplane closest to the negative examples H_2, defined by $\sum_{i=1}^{n} w_i x_i + b = -1$. These hyperplanes contain the closest examples to H, called the *support vectors*. The problem is then to maximize the *margin*, that is the distances from H to its support vectors, which is also the distances from H to H_1 and H_2, which are equal to $\frac{1}{\sqrt{\sum_{i=1}^{n} w_i^2}}$. The problem of learning can be turned into an optimization problem: maximizing the margin, given some constraints. The general non-separable case is handled by introducing variables ξ_j covering the different error cases so that the training examples x_j would verify:

$$\sum_{i=1}^{n} w_i x_{ji} + b \geq +1 - \xi_j \text{ for } y_i = +1$$

and:

$$\sum_{i=1}^{n} w_i x_{ji} + b \leq -1 + \xi_j \text{ for } y_i = -1$$

for all j. This short presentation was just intended to give an idea of the Support Vector Machine framework. The full context is then given by switching to a Lagrangian formulation and solving the resulting optimization problems.

We used the implementation of Transductive Support Vector Machine by Joachims [Joachims, 1999]. It has been shown to be efficient for text categorization, where there may be a lot of attributes[14]. However, it also proved

[14]Sometimes, the frequency of each word is an attribute.

not to adapt well to the size of our dataset. For 10000 examples, it took half an hour on a Sparcv9 at 400MHz with 1 Gb memory. For 30000 examples (which is only one tenth of our dataset), we stopped the experiment after 10 hours.

6.8 Linear Regression

The experiments with C5.0 and Naive Bayes in particular illustrated several issues related to the discretization and selection process. One way to solve that is to skip this step and proceed to continuous learning as we did with K-nearest neighbour. The framework to do so is the one of statistical regression.
We have only considered linear regression. If we denote $(x_{i1}, x_{i2}, \ldots, x_{in})$ the n attributes of example i and y_i its class, can we find $(\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$ such that, for every example in the training data,

$$y_i = \beta_0 + \sum_{j=1}^{n} \beta_j x_{ij}$$

[Moore, 2001] describes a method to get the exact solution when it exists and which often works as well to give an approximate solution. Let us denote Z the following two-dimensional matrix (in our case 27x300000):

$$Z = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ & & \ldots & & \\ 1 & x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix}$$

and β and Y such as:

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \ldots \\ \beta_n \end{pmatrix} \text{ and } Y = \begin{pmatrix} y_0 \\ y_1 \\ \ldots \\ y_m \end{pmatrix}$$

Hence, we want to find β such that:

$$Z \times \beta = Y$$

This implies:

$$Z^T Z \beta = Z^T Y$$

Now, in our case, Z^TZ is symmetric positive-definite, hence invertible. Finally, we get:

$$\beta = (Z^TZ)^{-1}Z^TY$$

Therefore, our problem comes down to normalizing the elements of Z, inverting Z^TZ and multiplying $(Z^TZ)^{-1}$, Z^T and Y. Depending on the number of zeros and the kind of operations involved, matrix operations can be either tractable or very time-consuming. Consequently, we have implemented this procedure to check how time-consuming it was. The fact that Z has more than 300000 rows prevented us again from getting definite results. On a Sparcv9 at 400MHz with 1 Gb memory, the computation times were:

- 10000 examples: 5 minutes.
- 20000 examples: 15 minutes.
- 30000 examples: 35 minutes.
- 40000 examples: 1 hour.

Again, it seems impossible to get a result for more than 300000 examples. This illustrates how difficult it may be to find a learner appropriate to our case. In our study on statistical regression, we notice that the complexity of algorithms is always exponential either in the number of attributes or in the number of examples. Therefore, without reducing the number of attributes, it seems impossible to find an adequate regression algorithm. Nevertheless, after performing such a reduction, it could be interesting to come back to regression and perform a bigger study than this one, using a language dedicated to matrix calculus such as Matlab to speed up the experiments (we used Perl).

6.9 Conclusion

In this chapter, we have tried learning algorithms different from the one currently used in the system: C5.0. A summary of those is given in table 6.1.

It turns out that many of them could not be applied to our dataset because of the large number of examples (300000). Thus SNNS, Ripper, Nearest Neighbour, SVM and Linear Regression were intractable. Adaboost, albeit very time consuming as well, eventually gave some results but no significant improvements on F-Score.

On the contrary, a very simple Naive Bayes classifier was as fast as C5.0 and

produced better results than the latter, being in particular able to learn the F-Score function. However, the error rate of Naive Bayes on the task is not particularly good.

This surprising result shows the usefulness of our framework. Considering only intermediate criteria, such as the error rate of the learning algorithm, to take design decisions may lead to incorrect choices. The final quality of summaries, which our framework enables us to evaluate, indeed depends on many parameters in a non trivial way.

Algorithm	Description	Advantages	Drawbacks	In CALLISTO
C5.0	Decision Tree algorithm	Fast Efficient for numerous, diverse applications	Over-fitting frequent Prior feature discretization needed	Used in the first version
AdaBoost C5.0	Iterative Decision Tree algorithm	Generally improves C5.0 accuracy	Slower (wrapper)	Does not help to improve F-Score
Stuttgart Neural Networks Simulator	Neural Networks	Reasonably fast No feature discretization needed	Too many parameters not suitable for all problems	Does not improve with learning in our system
Ripper	Rule Learning algorithm	Human-readable rules The pruning step reduces the risk of over-fitting	Slow for large numbers of examples	Intractable
Naive Bayes	Bayesian Learning	Fast Simple to apply	Assumptions of independence and normality "Brute force" statistics	Gives better F-Score scores than C5.0
Nearest neighbour	Instance Based Learning	Simple No training process	Very slow at execution time for large datasets Highly sensitive to noise and irrelevant attributes Curse of dimensionality	Intractable
SVM Light	Transductive Support Vector Machine	Solid theoretical foundation Fast execution once trained Supports large numbers of attributes	Slow to train for a large number of examples	Intractable
Linear Regression	Regression	Simple No prior discretization needed for the class and attributes	Slow by the lack of matrix calculation engines Assumption of linearity unlikely to hold	Intractable

Chapter 7

Other experiments

The last two chapters have focused on the learning component of the system, either for the choice of the measure and related issues or to determine the best learning algorithm. We now show how the framework can be useful to solve other kinds of questions in CALLISTO, regarding the text and system vectors.

7.1 The Text Vector

7.1.1 Introduction

We need to characterize a text using a limited number of relevant attributes such as its length, the way it is organized, the number of meaningful units and so on. The features currently used as an input for the Machine Learning classifier are:

- The number of characters, words, sentences and paragraphs.
- The number of connectives, Proper Nouns/Acronyms.
- The number of content phrases, one, two, three and four-or-more-instance content phrases.
- The number of one, two, three and four-or-more-instance bigrams and content bigrams.

See chapter 3 (section 3.4.2) for more information about these attributes. The aim of this section is to consider adding new attributes to the text vector to improve CALLISTO's performance. In other words, we want to provide the classifier with more meaningful elements to find the best combination for every case. This is a difficult task: if we knew the circumstances in which one tool is better than another, we definitely would not need Machine Learning. So, this can be seen as a sort of guessing game: which features of the text are the most important in the choice of the most adequate system vector?
Eventually, after having added other attributes, we will try to detect irrelevant and redundant attributes using feature selection techniques.

7.1.2 New attributes

The attributes we computed follow.

- Average lengths: The first idea for new text attributes is that the numbers of characters, words, sentences and paragraphs are not relevant on their own but when we combine them. That is to say we could give to the learner the average lengths of a word, a sentence, a paragraph and it will probably help more than the above numbers, which are measuring the length of the document in various ways. Indeed, the average length of a word for instance can characterize the author's style and is, as a matter of fact, sometimes used as a readability indicator [Mani, 2001a].

- Maximal lengths: Another idea was to determine maximal length for the same units studied before (words, sentences and paragraphs). However, minimal and maximal lengths of a word do not make much sense since they are very local properties[1]. The attributes to test are maximal length of a sentence or of a paragraph.

- Rareness: It is a very natural idea to think that the more rare the words used by the author are, the more accurate they are and therefore, the more difficult the text will be to summarize. A list of rare words has been extracted from a frequency list established at the University of Ottawa [Copeck *et al.*, 1999] and we can count the number of rare words per text.

[1]That is to say that they are based on only one word in the text and do not give any insight into its overall organization. The other attributes we propose are rather related to statistical count with the hope that they are more meaningful.

- Pronouns: Another intuitive idea is that the more subjective an author is, the more personal pronouns he uses. This is the list of personal pronouns we have chosen to count[2]:

I	he	
me	him	ourselves
my	himself	ours
myself	his	they
mine	she	them
you	her	themselves
you-all	herself	their
your	hers	theirs
yourself	we	thee
yourselves	us	thou
yours	our	

- Definite and indefinite noun phrases: The intuition behind the suggestion of this attribute is less obvious. It is based on the idea that more general texts could use on the whole fewer definite articles. It is difficult to say whether this is important in the task of summarizing but it is worth testing.
  The selected words are:

Definite words	Others
the	a, an
this	each
that	every
these	some, any
those	much, many

- Marcu's connectives [Marcu, 1999]: Our last proposal actually gathers several older ideas about counting for example the number of definitions or the number of comparisons. The simplest way to do that is to count specific connectives. Now, these connectives are already counted on the whole in the system. We have decided to divide the list into several smaller ones, based on the meaning of the connectives. This

[2]This list is very debatable. For example, it may be argued that only first and second person pronouns are indicators of subjectivity and that "we" should be ignored in scientific texts. We decided to count all pronouns all the same, filtering this list could be part of future work.

work proves to be very difficult. The final grouping we got is probably not the best one (if a best one exists, which is not guaranteed given the subjectivity involved in this task) and could still contain many mistakes. The different kinds of connectives identified are: Addition, Cause, Comparison, Concession, Conclusion, Consequence, Miscellaneous, Opposition and Stress. The lists are provided in appendix C.

7.1.3 Feature Selection

First, we evaluated the effect of the new attributes using the variations of the learning rate. This has proven not satisfactory for two reasons:

- We have seen in chapter 6 that the error rate was not necessarily correlated with the final overall quality of the summaries.
- The variations in the error rate were very flat and did not enable us to establish which combinations were useful.

Therefore, we are faced with a classical Machine Learning problem: we have a large number of attributes, not knowing how many are relevant. A good state-of-the-art paper on Feature Selection is [Blum and Langley, 1997]. We still have a problem of computation time which makes all wrapper approaches (i.e. using the learning algorithm iteratively) intractable. We will therefore look at the best-known filtering approaches, which have proven to give good results on most datasets:

- FOCUS [Almuallim and Dietterich, 1991]
- Relief [Robnik-Sikonja and Kononenko, 1997]
- LVF [Liu and Setiono, 1996]

Unfortunately, FOCUS is similar to an exhaustive search and therefore impossible to apply to our dataset in a reasonable amount of time. Relief and LVF are not much more suitable since, as they include randomness, they are applicable to some small parts of our data but the significance of the results for few instances selected at random is very questionable.
Other simpler (wrapper) approaches just consist of Heuristic Searches as described in [Blum and Langley, 1997]:

- *Forward selection* starts from no attribute and adds each of them, one at a time, looking at the error rate for improvement. We can adapt this method with the framework, looking at the F-Score measure rather than the accuracy of the learner.

- *Backward elimination* exploits the same idea, except that it starts from all attributes together and removes them gradually. The order is absolutely essential and it is likely that Backward elimination will give different results from Forward selection, although it is impossible to predict before which one will output, in the end, the configuration with the best average F-Score.

7.1.4 Results

We considered only forward selection and backward elimination. We conducted a few experiments for forward selection with one of the version which was found to have acceptable results (C5.0 - Recall measure discretized by 5-means). But the F-Score scores, with one text attribute, presented in table 7.2, are still very flat, close to the one we have with no text attribute whatsoever (58.3%) and close to random selection (57.9%) as all the results obtained with C5.0 in chapter 5.
Therefore, we decided to conduct the same study with Naive Bayes. We chose the configuration which yielded the best results in chapter 6 (Equal Width - 4 classes). A few preliminary scores are presented in table 7.1. There is no significant difference when we add the 18 new attributes to the 19 existing ones. However, a surprising result is that removing the text vector altogether enables us to achieve consistent improvement and reach an F-Score score slightly above the baseline.
From this strange result, we applied the forward selection algorithm, in table 7.3, and found that every attribute causes a loss in the overall performance. We kept the one which caused the smallest loss (average length of a word) and tried to add the others, one by one, in the third column of the table. The performance dropped again. In experiments not reported here, we tried every possible pair of text attributes to see if one was working better. This hope proved futile: all the scores were very close to 63%. These results are extremely disappointing and mean that:

- We can obtain good results without characterizing the input text. That is to say that, on our corpus, one system vector (one configuration) works reasonably well (see chapter 8 for more detail).

- None of the attributes we tried characterized the text efficiently for the purpose of discriminating between system vectors.

We will comment more on these results in chapters 8 and 9.
Finally, we tried backward elimination (36 attributes). The results, given

in table 7.4, are extremely flat as were the ones with C5.0. It confirms that there is a lot of noise in this set of attributes, much more than there is relevant information. That is to say, very informally, that the text attributes are not related to the class, or in a way too complicated to identify for the learning algorithms we tried. In the end, they bring more misleading information and that is why we talk about *noise*. And it is in agreement with the sad conclusion above: in the experiments we did with the measures we chose and the learning algorithms available, we have not found any useful text attribute.

Best score with original text vector (19 text attributes): 62.66%
Score with no text vector (0 text attribute): 65.46%
Score with full text vector (37 text attributes): 62.41%
Baseline: 65.1%
Statistical significance: 1.65%

Table 7.1: Remarkable F-Score scores with Naive Bayes.

7.2 System vector

7.2.1 Segmenters and Key Phrase extractors

This section illustrates that we can use the framework to test a hypothesis rather than to improve the system. The assumption we would like to confirm is that every *black box* tool inside the system is useful and therefore, that the final quality drops significantly when we remove any of them. That is what Mani [Mani, 2001a], p. 239, calls an *ablation* study since it consists in removing one element of the system to evaluate its importance.
These are very simple experiments to conduct, we just need to remove from the training file the lines using a given tool and run the learning and evaluation process in cross validation. We used the configuration C5.0 - KPiA - 5-means - First found. The statistical significance for these experiments is 3.80% for Recall and 2.73% for F-Score. The results are presented in figures 7.1 and 7.2.

System vector +	F-Score
number of indefinite articles	58.87%
number of definite articles	59.29%
number of pronouns	58.15%
number of stress words	58.84%
number of opposition words	58.64%
number of misc. Marcu's words	57.70%
number of consequence words	58.11%
number of conclusion words	57.75%
number of concession words	58.64%
number of comparison words	58.90%
number of cause words	57.32%
number of addition words	58.62%
number of rare words	58.38%
maximal length of a paragraph	58.59%
maximal length of a sentence	58.32%
average length of a paragraph	57.97%
average length of a sentence	58.79%
average length of a word	58.52%
number of characters	59.27%
number of words	58.51%
number of sentences	58.28%
number of paragraphs	58.67%
number of (Marcu) connectives	59.07%
number of PNs, acronyms	58.74%
number of content phrases	58.57%
number of one-instance content phrases	59.16%
number of two-instance content phrases	58.38%
number of three-instance content phrases	57.47%
number of four-or-more instance content phrases	56.96%
number of one-instance content bigrams	58.35%
number of two-instance content bigrams	58.09%
number of three-instance content bigrams	57.14%
number of four-or-more instance content bigrams	57.26%
number of one-instance bigrams	59.01%
number of two-instance bigrams	58.66%
number of three-instance bigrams	58.04%
number of four-or-more instance bigrams	58.67%

Table 7.2: F-Score scores for forward selection with C5.0.

System vector +	1 text attribute	2 text attributes (+average length of a word)
number of indefinite articles	63.17%	63.16%
number of definite articles	63.55%	63.52%
number of pronouns	64.08%	64.16%
number of stress words	63.38%	63.16%
number of opposition words	63.65%	63.52%
number of misc. Marcu's words	63.60%	63.73%
number of consequence words	63.50%	63.52%
number of conclusion words	63.24%	63.22%
number of concession words	63.93%	63.52%
number of comparison words	63.68%	63.59%
number of cause words	63.66%	63.72%
number of addition words	63.39%	63.38%
number of rare words	63.94%	63.89%
maximal length of a paragraph	64.04%	63.90%
maximal length of a sentence	64.75%	64.56%
average length of a paragraph	65.05%	64.62%
average length of a sentence	65.13%	65.13%
average length of a word	**65.45%**	
number of characters	63.71%	63.61%
number of words	63.62%	63.43%
number of sentences	63.20%	62.94%
number of paragraphs	63.71%	63.66%
number of (Marcu) connectives	63.52%	63.67%
number of PNs, acronyms	63.70%	63.74%
number of content phrases	63.55%	63.30%
number of one-instance content phrases	63.29%	63.09%
number of two-instance content phrases	63.59%	63.60%
number of three-instance content phrases	63.56%	63.77%
number of four-or-more instance content phrases	63.29%	63.25%
number of one-instance content bigrams	63.61%	63.60%
number of two-instance content bigrams	63.70%	63.80%
number of three-instance content bigrams	63.76%	63.76%
number of four-or-more instance content bigrams	64.30%	64.10%
number of one-instance bigrams	62.91%	63.08%
number of two-instance bigrams	63.34%	63.46%
number of three-instance bigrams	63.95%	63.80%
number of four-or-more instance bigrams	64.19%	64.09%

Table 7.3: F-Score scores for forward selection with Naive Bayes.

System vector + Text vector Minus the following attribute:	F-Score
number of indefinite articles	62.55%
number of definite articles	62.39%
number of pronouns	62.43%
number of stress words	62.46%
number of opposition words	62.49%
number of misc. Marcu's words	62.38%
number of consequence words	62.37%
number of conclusion words	62.53%
number of concession words	62.38%
number of comparison words	62.47%
number of cause words	62.46%
number of addition words	62.44%
number of rare words	62.40%
maximal length of a paragraph	62.46%
maximal length of a sentence	62.42%
average length of a paragraph	62.43%
average length of a sentence	62.45%
average length of a word	62.45%
number of characters	62.31%
number of words	62.31%
number of sentences	62.42%
number of paragraphs	62.41%
number of (Marcu) connectives	62.47%
number of PNs, acronyms	62.35%
number of content phrases	62.40%
number of one-instance content phrases	62.43%
number of two-instance content phrases	62.43%
number of three-instance content phrases	62.52%
number of four-or-more instance content phrases	62.45%
number of one-instance content bigrams	62.37%
number of two-instance content bigrams	62.36%
number of three-instance content bigram	62.50%
number of four-or-more instance content bigrams	62.44%
number of one-instance bigrams	62.41%
number of two-instance bigrams	62.43%
number of three-instance bigrams	62.39%
number of four-or-more instance bigrams	62.39%

Table 7.4: F-Score scores for backward elimination with Naive Bayes.

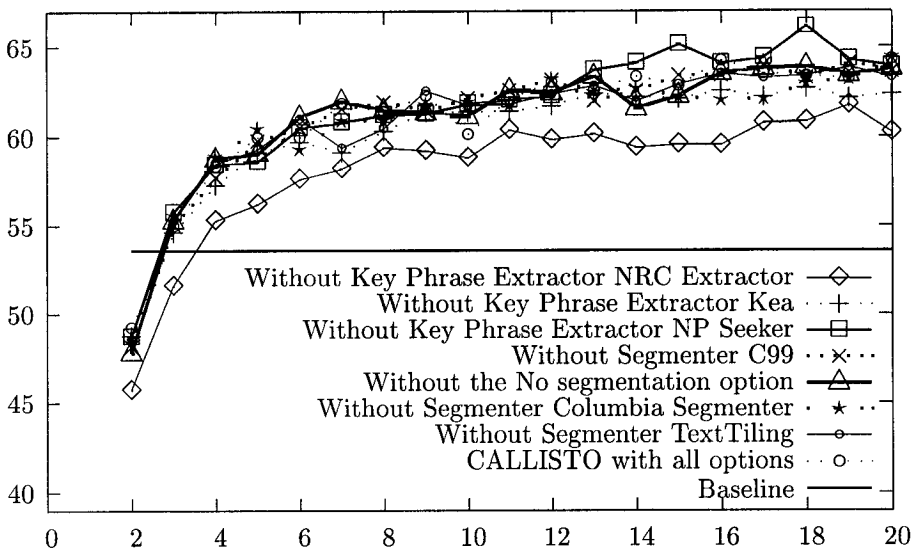

Figure 7.1: Recall scores for the ablation study.

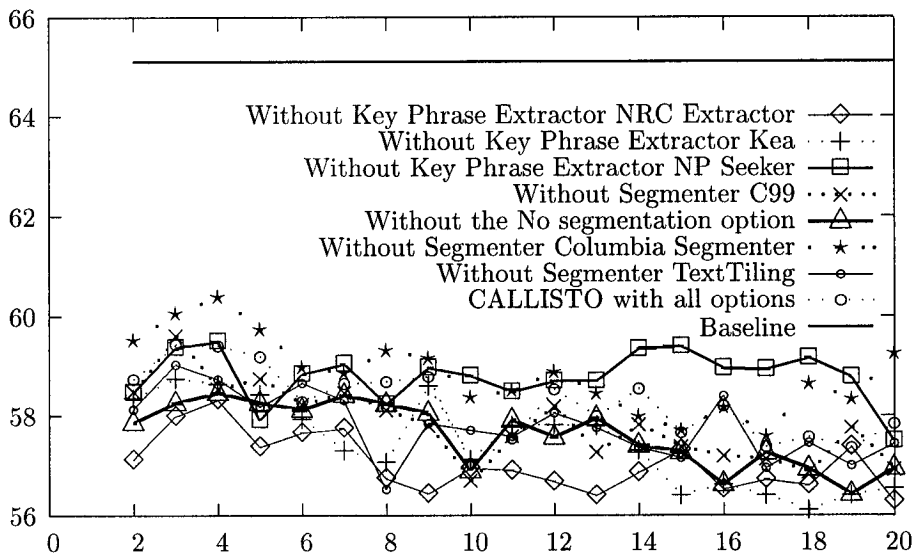

Figure 7.2: F-Score scores for the ablation study.

The results are quite flat which seems to indicate that none of the elements is really necessary and that, thanks to the joint effect of multiple modules for each task and of machine learning, CALLISTO manages to make up for the successive ablations by using alternative configurations. The full configuration is above most of the others for both the Recall scores and the F-Score scores.
A remarkable result is that dropping NP Seeker improves the overall performances, in particular for the largest number of classes. This is consistent with one of the first results noted in the first experiments with the system [Copeck *et al.*, 2002], that one of the first rules learnt by C5.0 is not to use NP Seeker. We can assume that for the largest number of classes, the learning process is worse and thus this rule is not as important as it should be. Another result we could not explain is that dropping Segmenter for the smallest numbers of classes seems to help as well.
On the other hand, Extractor is apparently very valuable for CALLISTO and removing it causes consistent losses in both Recall and F-Score. This result is also true for Kea, although less significantly. A last remark is that dropping the No segmentation option does not hurt much, which tends to confirm the initial assumption that a segmentation step is necessary.

7.2.2 Replacing target lengths with compression rates

One assumption after looking at the results of chapter 5 is that C5.0 can learn how to improve the Recall scores but not the F-Score scores. However, we know that they are related with each other and with the length of the summary. So the idea here is to try and replace the length of the summary (in number of sentences) in the system vector with the compression rate. This way, C5.0 might find the link between them and build rules yielding large recall scores at high compression rates, which is equivalent to having large F-Score scores.
The results are given in figures 7.3 and 7.4. For Recall, the curves with compression rate and with length of a summary are very close. One is sometimes slightly below, sometimes slightly above but these differences are not significant. And for F-Score, unfortunately, the results are still very flat and much lower than what we have with Naive Bayes. But adding the compression rate with Recall seems to help a bit, to confirm our original intuition. However the improvement is not significant enough to deserve further investigation.

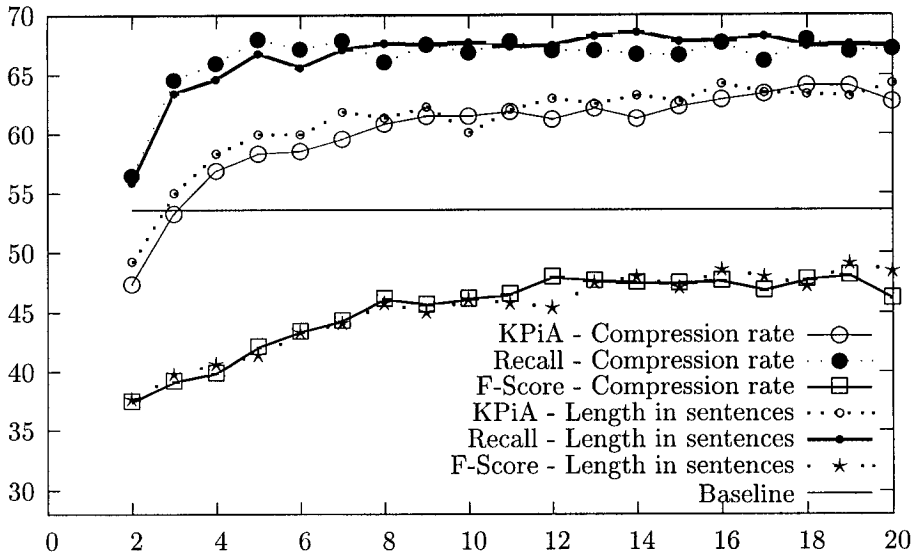

Figure 7.3: Recall scores with compression system attribute.

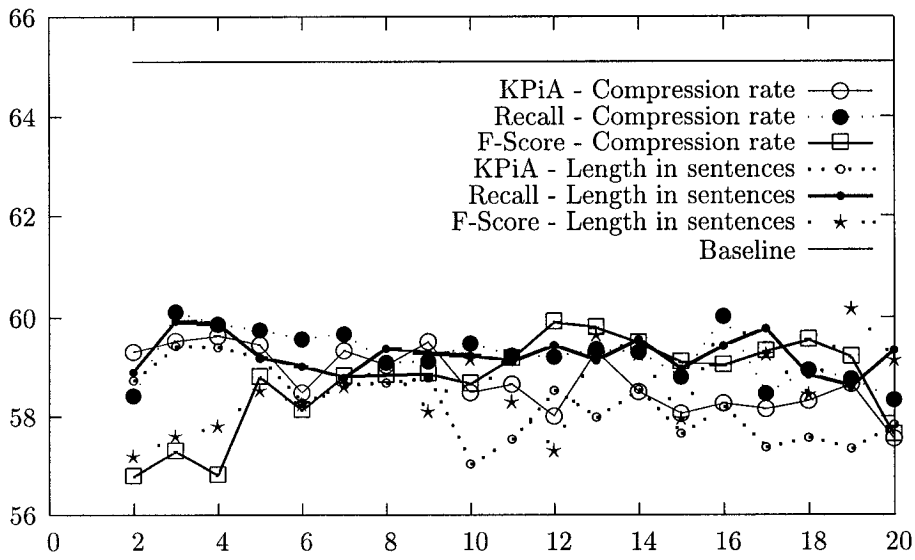

Figure 7.4: F-Score scores with compression system attribute.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis, we presented the CALLISTO summarization system and tried to modify it in several ways to improve it. We have judged the quality of the overall summaries produced and therefore the relevance of the changes with an evaluation framework. This framework, based on a measure proposed by Lin and Hovy, is fully automatic and can be applied to large corpora, which gives more significance to the results. It comes with a very demanding baseline obtained with the leading text extraction technique and a measure of the statistical significance of the results with 2-way ANOVA.
The following modifications of the system have been attempted.

- In chapter 5, we described the effects of changes of the measure used for training, the discretization process and the selection method among the best class. The conclusion is that the choice of the measure between KPiA, Recall and F-Score is highly learner-dependent. The discretization stage does not change much but little improvement can be in general achieved by adjusting the discretization (K-means, equal width intervals or equal frequency intervals) and the number of classes. The optimal choice is ultimately learner-dependent as well. There are almost no differences between the selection processes but we discovered a promising combination with Recall as a training measure and shortest for the selection process to improve the performance on F-Score.

- In chapter 6, we presented the results obtained with different learning algorithms. Most were intractable given the size of our training data.

However, we found that Naive Bayes produced good results and could learn how to get high F-Score scores more efficiently than C5.0. We think that this result is due in part to the fact that the selection process based on probabilities is better than the ones we investigated with C5.0.

- In chapter 7, the effects of adding or removing text attributes were studied. It turns out that none of them is really useful for the task and that the best performance can be achieved with Naive Bayes without text vector, that is applying the same configuration on every text. We also found with an ablation study that most of the segmenters and extractors were useful for the task.

The results from the feature selection study are very disappointing. They show that, so far, we have not found an adequate way to characterize the text since no text attributes, either alone or in pairs, were consequently improving the scores obtained by the system without text vector. We now want to understand more about the differences between the different versions of the system and study four remarkable configurations in more detail.

8.2 A study of remarkable configurations

The configurations chosen are presented, along with their respective normalized averages in Recall and F-Score in table 8.1.

	A	B	C	D
Learning algorithm	C5.0	C5.0	Naive Bayes	Naive Bayes
Measure for training	KPiA	Recall	F-Score	F-Score
Discretization	K-means	K-means	Equal Freq	Equal Freq
Number of classes	5	14	4	4
Number of text attributes	19	19	19	0
Recall	59.94%	68.55%	58.22%	52.96%
F-Score	59.19%	59.55%	62.66%	65.46%

Table 8.1: Scores and characteristics of the four configurations selected.

We chose these configurations because they were representative of the different findings we made throughout the thesis.

- Configuration A is the version of CALLISTO chosen as a starting point for this thesis, the one which entered DUC 2001 and DUC 2002.
- Configuration B is the best version we had for Recall, after having replaced the measure used for training and changed the number of classes in chapter 5. The F-Score score is not significantly different from A but the Recall score is.
- Configuration C was found when we tried another learning algorithm in chapter 6. It is the best with the Naive Bayes classifier, the text vector being the same as for configurations A and B. The Recall score is about the same as with configuration A and significantly lower than with configuration B. The F-Score score is better than with the two previous versions and the difference is slightly above the statistical significance threshold.
- Configuration D is the one resulting from the removal of the text vector (from Configuration C). It is almost a baseline version as we will explain later. The Recall score is significantly smaller than all the others whereas the F-Score score is the best one.

We have decided to look at the system vectors and, in particular, to count the ones used more than 50 times (4.4% of the texts) by one configuration. The results are presented in table 8.2 in the format already used in chapter 4 for each system vector: segmenter, key phrase extractor, matching, keycount, sentcount, hitcount[1].
It is straight-forward that Configurations A/B on the one hand and Configurations C/D on the other hand use very different configurations. Another experiment enables us to confirm that except these couples, the four configurations very rarely agree with one another. For a given text, no more than two configurations ever pick the same system vectors. A and B choose the same system vector 8.6% of the time whereas C and D agree on 58.2% of the cases. B and C select the same system vector on an insignificant 0.3% of the total number of texts. In all the other cases, the four configurations choose four different system vectors.
Configuration D chooses the same configuration (C99, with extractor, 5 keyphrases to extract and 5 sentences in the final summary) more than 99.5%

[1]Recall that, in all our experiments the values of matching and hitcount have been set to e and 2.

Configuration A	
s,e,e,3,8,2	21.7%
s,e,e,3,12,2	21.2%
s,e,e,3,3,2	15.5%
s,e,e,3,4,2	5.2%
s,e,e,3,5,2	4.7%
Configuration B	
s,e,e,3,12,2	19.7%
s,e,e,3,8,2	9.2%
Configuration C	
c,e,e,5,5,2	58.0%
c,e,e,8,12,2	17.8%
s,n,e,12,3,2	10.3%
n,n,e,3,12,2	8.0%
Configuration D	
c,e,e,5,5,2	99.6%

Table 8.2: System vectors used by the different configurations.

of the time. The remaining cases are the ones regarding texts containing fewer than 5 sentences for which this configuration is not applicable. They are not statistically significant. This regularity is logical since there is no text vector in Configuration D. Hence it only bases its choice on probability in the system vectors and makes the same choice whatever the text is. The fact that this version of the system obtains the best F-Score average is very discouraging. It tends to invalidate the CALLISTO methodology, at least on this dataset, showing that we perform better choosing the best parameters and applying them on every text. Even with a full text vector, the same configuration is chosen most of the time by Configuration C.
The other two configurations, with C5.0, rely more on Segmenter than on C99. In fact, Configuration A chooses the combination Segmenter/Extractor in 70% of the cases. That said, they are less definite in their choices than C and D[2]. B uses 81 different system vectors, against 51 for A, 9 for C and 3 for D. This illustrates the fact that the text vector is much more important for the configurations with C5.0 than for the ones with Naive Bayes. Unfortunately, their F-Score scores are worse.

[2]This is probably due to the fact that C5.0 can learn rules fitting more particular cases whereas Naive Bayes' probabilities are more general and care less about small phenomena.

To conclude this analysis, we can say that none of the configurations studied seems to illustrate satisfactorily the methodology which CALLISTO is based on. However, Configuration D provides an interesting baseline for future work.

8.3 Future Work

For the work in this thesis to be complete, a reliable experiment with human judges would have been necessary. Analyzing the quality of a few summaries produced by the four configurations considered in the previous sections would be interesting to challenge the conclusions brought by the framework.

Other than that, the most promising direction of future work seems to be to keep using the framework to find useful text features and validate CALLISTO's methodology. The best configuration we found achieves only around 66% of the best possible performance and there is still much room for improvement, even without adding other tools or changing parameters, provided that we find the right attributes to characterize a document. The experiments we conducted to find the best possible scores reveal that every configuration could indeed be useful and brings a better F-Score than the others a non negligible amount of times, as reported in table 8.3. And these limits can even be pushed further by adding other new and more powerful components to the system. Thanks to its methodology taking advantage of other tools, CALLISTO may fully benefit from any improvement in the fields of segmentation or key phrase extraction.

The reason why the ones we tried did not work may be numerous: it is possible that the news articles are too close to one another to be summarized differently and that we need a more diversified corpus to test the usefulness of CALLISTO. It is also possible that we need higher-level features such as the ones that could be obtained through text categorization techniques.

The question of the adequate learning algorithm is not solved either. In our case, Naive Bayes outperformed C5.0 but we showed that it did it only using brute force and taking advantage of the fact that the best configuration produces good summaries all the time. It is possible that, with a good text vector, C5.0 would be able to produce finer grain rules, more appropriate to every case. Besides, we have discarded many learners on the assumption that we wanted them to work on our full corpus. Some of them could be trained on small parts of it (10%) and could maybe give results as good as Naive Bayes and C5.0 trained on 1000 texts. The performance would prob-

Segmenter	Key Phrase extractor	Number of uses for maximal score
C99	NRC Extractor	220
No segmentation	NRC Extractor	146
Columbia Segmenter	NRC Extractor	118
No segmentation	Kea	98
C99	Kea	89
C99	NP Seeker	83
TextTiling	NRC Extractor	79
No segmentation	NP Seeker	78
Columbia Segmenter	Kea	68
TextTiling	NP Seeker	49
Columbia Segmenter	NP Seeker	47
TextTiling	Kea	46

Table 8.3: Number of times one given configuration outperforms all the others

ably drop with the number of training examples decreasing but we do not know how fast and we do not know either what is the smallest acceptable size for the training data. Using 10-fold cross validation was very expensive too. For future experiments, we concluded that the data was probably stable enough to switch for instance to 4-fold cross validation.
Another promising direction to go to could be to investigate regression algorithms. We tried one but not with the right tools and using software dedicated to matrix calculus such as Matlab could probably do much better. As C5.0 is usable on our dataset, Cubist, the regression tree algorithm from the same package, could probably perform well. It could also be interesting to investigate approaches from the theory of optimization, to avoid the discretization process.
Finally, there was one experiment too time-consuming to be finished: we tried to modify the values of *matching* and *hitcount* and compute all the resulting summaries. With more time or a smaller dataset, we could check the effect of these parameters.

Appendix A

KPiAs evaluation method

A.1 Algorithm

```
G is the gold standard.
S is the summary to test.
T is the original text.

// First part: Create the KPiA list
kpia = "";
listkpia = [];
foreach word in G
  foreach stopword
    if word != stopword
      concatenate kpia and word;
    else
      if kpia != ""
        push kpia in listkpiakpia;

k = number of KPiAs;

// Second part: Compare the summary
// with the list
int listkpiainsummary[k] = [0, ..., 0];
foreach word in S
  for(i = 0 .. k)
    if word == listkpia[i][0]
```

```
      if listkpia[i] matches the sequence
      in S beginning with word
        listkpiainsummary[i]++;
        break for;

score = 0;

// Third part: Compute the score
for(i = 0 .. k)
  if listkpia[i] >= 1
    score = score+1
            -0.5*(listkpia[i]-1);

// Fourth part: Normalization
m = number of sentences in T;
int scoresentence[m];
j = 0;
foreach sentence in T
  foreach word in sentence
    for(i = 0 .. k)
      if word == listkpia[i][0]
        if listkpia[i] matches
        the sequence in sentence
        beginning with word
           scoresentence[j]++;
           break for;
  j++;

p = number of sentences in S;
S' = p highest scoring sentences
     according to scoresentence[];

Apply parts 2 and 3 to S' to get
bestscore;

if bestscore < score
  return 1;
else
  return score/bestscore;
```

A.2 Complexity

The complexity of the algorithm is, if we denote as s the number of stop-words, N the number of words in the text and n the number of words in the summary to evaluate (which is approximately equal to the number of words in the reference summary):

- $n * s$ for the creation of the list.
- $n * k$ for the comparison of the summary to test with the list.
- k to compute the score.
- $N * k + n * k$ for the normalization of the score (computing the nearly best score).

On the whole, we get a complexity of $O((N + n) * k)$ which is, if we bound the number of KPiAs k by the number of words in the gold standard n, approximately $O(N * n)$ where N is the length of the original text and n the desired length of the resulting summary. Computing the score for the nearly best summary is thus the bottleneck in the process.

Appendix B

Lin and Hovy's evaluation method

B.1 Algorithm

```
G is the gold standard.
S is the summary to test.

// First part: Prepare the texts

G' = ();
foreach word in G
  foreach stopword
    if word == stopword
      break foreach;
    else if stopword is
         the last stop-word
           word = stemmed(word);
           push word in G';
G = reverse(G');

S' = ();
foreach word in S
  foreach stopword
    if word == stopword
      break foreach;
    else if stopword is
```

```
          the last stop-word
            word = stemmed(word);
            push word in S';
S = reverse(S');

// Second part: Unigrams

overlap = 0;
n = length(G);
m = length(S);
foreach word in S
  foreach gold in G
    if word == gold
      overlap++;
      remove gold from G;
      break foreach;

recallunigrams = overlap/n;
f-scoreunigrams = 2*overlap/(n+m);

// Third part: Bigrams

overlap = 0;
foreach pair of words in S
  foreach gold pair of words in G
    if pair == gold
      overlap++;
      remove gold from G;
      break foreach;

recallbigrams = overlap/(n-1);
f-scorebigrams = 2*overlap/(n+m-2);

recall = (recallunigrams+
          2*recallbigrams)/3;
f-score = (f-scoreunigrams+
          2*f-scorebigrams)/3;

return (recall,f-score);
```

B.2 Complexity

If we denote as n the length of the summary to evaluate (which is approximately the length of the reference summary as well), p the complexity of the stemmer and s the number of stop-words, we get the following complexity:

- $2 * n * s * p$ for filtering and stemming. That is $O(n)$ since p and s are constant not depending on the number of words in the summary.
- $n * n$ for unigram count.
- $(n-1) * (n-1)$ for bigram count.

On the whole, we get a complexity of $O(n^2)$.

Appendix C

Marcu's connectives

Addition

add to this
additionally
and again
already
also
also because
and also
not only
alternatively
and
and another
another time
besides
besides that
briefly
by the way
further
furthermore
in addition
in other respects
in other words
incidentally
moreover
next
next moment
next time
parenthetically
second
secondly
simultaneously
that reminds me
third
to add
what is more
with that
with this

Cause

actually
as far as
as if
back
because
because of
because of this
before
before long
before that
before then
ever since
for
for fear that
for one
for that
for that matter

for that reason
for this
for this reason
given
given that
if
if ever
if not
if only
if so
in fact
indeed
insofar
it is because
let us assume
let us consider
mainly
merely
merely because
on condition
on the assumption
on the bases
on the basis
on the grounds
on this basis
only because
only if
only when
perhaps
possibly
presumably
provided
provided that
providing that
seemingly
simply
simply because
since
suppose
suppose that
supposedly
to explain
under the circumstances
under these circumstances
when
where
whether
whether or not
which
which is why
which means
who
with regard to
with respect to

Comparison

analogously
as
as for
as such
as to
as well
by comparison
by the same
correspondingly
either
either case
either event
either way
equally
in comparison
in place of
in such a
in such an
in the same way
just as
like
likewise
or
or again

or else
similarly
such as

Concession

admittedly
all the same
arguably
at least
but also
but then
but then again
despite
despite this
even
even after
even before
even if
even so
even then
even though
even when
except
except that
except when
having said
however that may be
it may seem that
it stands to reason that
nevertheless
nonetheless
notwithstanding
now
now that
only
only after
the fact is
unless
up to now
up to this
yet

Conclusion

after all
all in all
anyhow
anyway
as a logical conclusion
at last
eventually
finally
in any case
in conclusion
in short
in sum
in the end
it can be concluded that
last
lastly
later
overall
summarizing
summing up
that is all
the end
to close
to conclude
to stop
to sum up
to summarize
ultimately

Consequence

accordingly
after
after a time
after that
after this
afterwards

then again
all this time
and then
as a consequence
as a corollary
as a matter of fact
as a result
as evidence
as long as
as soon as
as we will
consequently
following
for example
for instance
hence
in consequence
in doing
in doing so
in order to
in point of fact
in so doing
in that
in that case
in the event
in this case
in this connection
in this respect
in this way
in turn
in which
in which case
including
it follows
just then
on top of it
on which
particularly
particularly when
so
so far
so that
subsequently
such that
that is
that is how
that is to say
that is why
that way
the thing is
then
then again
thereafter
thereby
therefore
thereupon
thus
thus far
to illustrate
to the degree that
to the extent
to this end
to wit

Miscellaneous

never again
all right
at first
at once
at that moment
at that time
at the moment
at the outset
at the same time
at this date
at this moment
at this point
at this stage
at which
by that time

by the time
by then
each time
every time
everywhere
excuse me
fine
first
first of all
fortunately
from now on
from then on
here
heretofore
hitherto
in case
in general
in the beginning
in the case of
in the first place
in the hope that
initially
instantly
just
just before
lest
luckily
neither
no matter
okay
on account of
once
oops
originally
presently
regardless
some time
soon
suddenly
the first time
the last time
the latter
the moment
the next time
this time
to begin with
to comment
to interrupt
to open
to start with
too
true
unfortunately
until
until then
well
without
you know

Opposition

but
but also
although
apart from
as against
as though
aside from
by contrast
contrariwise
conversely
else
elsewhere
however
in contrast
whatever
whenever
whereas
whereby
wherein
whichever

while
whereupon
wherever
in spite of
instead
instead of
meanwhile
nor
not
not because
not only
not that
on a different note
on another
on balance
on one side
on the contrary
on the one hand
on the other hand
on the other side
other than
otherwise
though
whoever

Stress

above all
again
again and again
once again
as it happened
as it is
as it turned out
as we shall
at any rate
by
by all means
by and large
certainly
clearly
conceivably
considering
decidedly
definitely
doubtless
earlier
especially
essentially
evidently
formerly
I mean
in particular
in the meantime
in truth
incontestably
incontroversialy
indisputably
indubitably
it is only
largely
let us
moreover
most likely
more accurately
more importantly
more precisely
more specifically
more to the point
much as
much later
naturally
needless
no doubt
notably
obviously
of course
once again
once more
plainly
previously

put another way
quite likely
rather
returning to
speaking of
specifically
still
sure enough
surely
that done
the more
this means
to be sure
to clarify
to get back
to note
to repeat
undeniably
undoubtedly
unquestionably
very likely

Bibliography

[Aha and Kibler, 1989] Aha, D. W., and Kibler, D. Noise-tolerant instance-based learning algorithms. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 794-799. 1989

[Almuallim and Dietterich, 1991] Almuallim, H., and Dietterich, T. G. Learning with many irrelevant features. In Proceedings of the Ninth National Conference on Artificial Intelligence, 547-552. 1991.

[Aone *et al.*, 1997] Aone, C., Okurowski, M. E., Gorlinsky, J., and Larsen, B. A Trainable Summarizer with Knowledge Acquired from Robust NLP Techniques. 1997. In [Mani and Maybury, 1999].

[Baldwin *et al.*, 2000] Baldwin, N., Donaway, R., Hovy, E., Liddy, E., Mani, I., Marcu, D., McKeown, K., Mit-tal, V., Moens, M., Radev, D., Sparck Jones,K., Sundheim, B., Teufel, S., Weischedel,R., and White, M. An Evaluation Roadmap for Summarization Research. www-nlpir.nist.gov/projects/duc/roadmapping.html. 2000.

[Banko *et al.*, 1999] Banko, M., Mittal, V., Kantrowitz, M. and Goldstein, J. Generating Extraction-Based Summaries from Hand-Written One by Text Alignment. Submitted to the 1999 Pac. Rim Conf. on Comp. Linguistics. 1999.

[Barker and Cornacchia, 2000] Barker, K., and, Cornacchia, N. Using Noun Phrase Heads to Extract Document Keyphrases. Proceedings of the Thirteenth Canadian Conference on Artificial Intelligence (LNAI 1822). Montreal, 40-52. 2000.

[Blum and Langley, 1997] Blum, A.L., and Langley, P. Selection of Relevant Features and Examples in Machine Learning. Artificial Intelligence, 97:245–271. 1997.

[Boguraev and Kennedy, 1997] Boguraev, B., and Kennedy, C. Salience based content characterization of text documents. In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization. 1997. Reprinted in [Mani and Maybury, 1999].

[Boisen and Bates, 1992] Boisen, S. and, Bates, M. A practical methodology for the evaluation of spoken language systems. In Proceedings of the Third Conference on Applied Natural Language Processing. 162-169. 1992.

[Brandow *et al.*, 1995] Brandow, R., Mitze, K. and Rau., L. Automatic condensation of electronic publications by sentence selection. In Information Processing and Management, 31(5):675–685. 1995.

[Burges, 1998] Burges, C.J.C. A Tutorial on Support Vector Machines for Pattern Recognition. In Data Mining and Knowledge Discovery, 2(2):955-974. 1998.

[Choi, 2000] Choi, F. Advances in domain independent linear text segmentation. In Proceedings of ANLP/NAACL-00. 2000.

[Cohen, 1995] Cohen, W. Fast effective rule induction. In 12th International Conference on Machine Learning, Lake Tahoe, California. 1995.

[Copeck *et al.*, 1999] Copeck, T., Barker, K., Delisle, S., and Szpakowicz, S. More Alike than Not–An Analysis of Word Frequencies in Four General-Purpose Text Corpora. In Proceedings of the Fourth Conference of the Pacific Association for Computational Linguistics: 282-287. 1999.

[Copeck *et al.*, 2002] Copeck, T., Japkowicz, N., and Szpakowicz, S. Text Summarization as Controlled Search. in the Proceedings of the Fifteenth Conference of the Canadian Society for Computational Studies of Intelligence (AI'2002). 268-280. 2002.

[Damerau, 1979] Damerau, F. The Transformational Query Answering System (TQA) Operational Statistics. IBM Research Report RC7739. 1979.

[DeJong, 1982] DeJong, G. An Overview of the FRUMP System. In Strategies for Natural Language Processing, W.G.Lehnert & M.H.Ringle (Eds), Lawrence Erlbaum Associates. 149-176. 1982.

[Donaway *et al.*, 2000] Donaway, R.L., Drummey, K.W., and, Mather, L.A. A Comparison of Rankings Produced by Summarization Evaluation Measures In Proceedings of the Workshop on Automatic Summarization, 69-

78. New Brunswick, New Jersey: Association for Computational Linguistics. 2001.

[Dougherty *et al.*, 1995] Dougherty, J., Kohavi, R., and Sahami, M. Supervised and Unsupervised Discretization of Continuous Features. In International Conference on Machine Learning, 194-202. 1995.

[DUC, 2001 and 2002] Document Understanding Conference, National Institute of Standards and Technology http://duc.nist.gov/

[Edmundson, 1969] Edmundson, H.P. New methods in automatic abstracting. Journal of the ACM 16(2), 264–285. 1969. Reprinted in [Mani and Maybury, 1999].

[Falkedal, 1994] Falkedal, K. Evaluation methods for machine translation systems: An historical overview and critical account. ISSCO draft report, University of Geneva, Geneva. 1994.

[Fayyad and Irani, 1993] Fayyad, U.M., and Irani, K.B. Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-93), 1022-1029. 1993.

[Frank *et al.*, 1999] Frank, E., Paynter, G., Witten, I., Gutwin, C., and, Nevill-Manning, C. Domain-Specific Keyphrase Extraction. IJCAI:668-673. 1999.

[Freund and Schapire, 1996] Freund, Y. and, Schapire, R.E. Experiments with a New Boosting Algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, 148-156, Morgan Kaufmann. 1996.

[Galliers and Sparck Jones, 1995] Galliers, J. and Sparck Jones, K. Evaluating Natural Language Processing Systems. Lecture Notes in Artificial Intelligence. Springer, Berlin - Heidelberg - New York. 1995.

[Grishman and Sundheim, 1996] Grishman, R. and, Sundheim, B. Message understanding conference - 6: A brief history. In Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen. 1996.

[Hearst, 1994] Hearst, M. Multi-Paragraph Segmentation of Expository Text. Proceedings of the 32nd Meeting of the Association for Computational Linguistics, Los Cruces, NM. 1994.

[Holte, 1993] Holte, R.C. Very simple classification rules perform well on most commonly used datasets. In Machine Learning 11, 63-91. 1993

[Hull, 1993] Hull, D. Using statistical testing in the evaluation of retrieval experiments. In Proc. of SIGIR '93, 329-338. The Association for Computing Machinery. 1993.

[Jarke *et al.*, 1985] Jarke, M., Stohr, E.A., Turner, J.A., Vassiliou, Y., White, N.H. and, Michaelsen K. A Field Evaluation of Natural Language for Data Retrieval. In IEEE Transactions for Software Engineering, Vol.SE-11, No.1. 1985.

[Jing *et al.*, 1998] Jing, H., Barzilay, R., McKeown, K. and Elhadad, M. Summarization Evaluation Methods Experiments and Analysis. In AAAI Intelligent Text Summarization Workshop (Stanford, CA), pp. 60–68. 1998.

[Joachims, 1999] Joachims, T. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, MIT-Press. 1999.

[John and Langley, 1995] John, G., and Langley, P. Estimating continuous distributions in Bayesian classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 338-345. 1995.

[JTEC, 1989] JTEC Panel on Machine Translation. Japanese Techology Evaluation Centre. 1989.

[Kan *et al.*, 1998] Kan, M.-Y., Klavans, J., and, McKeown., K. Linear Segmentation and Segment Significance. In Proceedings of WVLC-6, pp. 197-205. 1998.

[Kanungo *et al.*, 2000] Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. The analysis of a simple k -means clustering algorithm. In Symposium on Computational Geometry, 100-109. 2000.

[Kupiec *et al.*, 1995] Kupiec, J.M., Pedersen J., and, Chen F. A Trainable Document Summarizer. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 68-73, Seattle, Washington, July 1995. Reprinted in [Mani and Maybury, 1999].

[Lin, 2001] Lin, C.-Y. Summary Evaluation Environment. http://www.isi.edu/ cyl/SEE. 2001.

[Lin and Hovy, 2002] Lin, C.-Y., and Hovy., E.H. Manual and Automatic Evaluations of Summaries. In Proceedings of the Workshop on Automatic Summarization post-conference workshop of ACL-02, Philadelphia, PA, U.S.A., July 11-13, 2002.

[Liu and Setiono, 1996] Liu, H., and Setiono, R. A Probabilistic Approach to Feature Selection. A Filter Solution. Machine Learning: ICML '96, 319-327. 1996.

[Luhn, 1958] Luhn, H.P. The automatic creation of literature abstracts. IBM Journal of Research and Development2 (2):159-195. 1958. Reprinted in [Mani and Maybury, 1999].

[Mac Queen, 1967] MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, (1), pp. 281-297. 1967.

[Mani, 2001a] Mani, I. Automatic Summarization. John Benjamins Pub Co. 2001.

[Mani, 2001b] Mani, I. Summarization Evaluation: An Overview. In Proceedings of the Second NTCIR Workshop on Research in Chinese & Japanese Text Retrieval and Text Summarization. 2001.

[Mani and Maybury, 1999] Mani, I. and Maybury, M.T. (eds.) Advances in Automatic Text Summarization. Cambridge, Massachusetts: MIT Press. 1999.

[Mani *et al.*, 1999] Mani, I., Gates, B. and Bloedorn, E. Improving Summaries by Revising Them. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, College Park, MD, pp. 558–565. 1999.

[Marcu, 1999] Marcu, D. Discourse trees are good indicators of importance in text. In [Mani and Maybury, 1999]. 1999.

[Minel *et al.*, 1997] Minel, J.-L., Nugier, S., and Piat G. How to appreciate the quality of automatic text summarization. ACL/EACL-97 summarization workshop. 1997.

[Mitchell, 1997] Mitchell, T. Machine Learning. McGraw Hill. 1997.

[Mooney, 1997] Mooney, R.J. Inductive logic programming for natural language processing. In Proceedings of the Sixth International Workshop on Inductive Logic Programming. 1997.

[Moore, 2001] Moore, A.W. Eight more Classic Machine Learning Algorithms. Class notes. School of Computer Science. Carnegie Mellon University. 2001.

[Morris *et al.*, 1992] Morris, A., Kasper, G., and Adams., D. The effects and limitations of automated text condensing on reading comprehension performance. Information Systems Research. 1992. Reprinted in [Mani and Maybury, 1999].

[Nagao, 1989] Nagao, M. A Japanese View on Machine Translation in Light of the Considerations and Recommendations Reported by ALPAC, U.S.A. (JEIDA Report). Japan Electronic Industry Development Association. 1989.

[Nastase and Szpakowicz, 2003] Nastase, V., and Szpakowicz, S. Exploring Noun-Modifier Semantic Relations. International Workshop on Computational Semantics. 2003.

[Oakes and Paice, 1999] Oakes, M., and Paice, C. The automatic generation of templates for automatic abstracting. In Proceedings of the information retrieval specialist group colloquium: British computer society. 1999.

[Paice, 1990] Paice, C. Constructing literature abstracts by computer: techniques and prospects. In Information Processing and Management. 1990.

[Papieni *et al.*, 2001] Papieni, K., Rouckos, S., Ward, T., and Zhu. W.-J. BLEU: a Method for Automatic Evaluation of Machine Translation. IBM Research Report RC22176(W0109-022). 2001.

[Pollock and Zamora, 1975] Pollock, J., and Zamora A. Automatic abstracting research at chemical abstracts service. In Journal of Chemical Information and Computer Science. 1975. Reprinted in [Mani and Maybury, 1999].

[Popescu-Belis, 1999] Popescu-Belis, A. Evaluation of Natural Language Processing Systems: A Model for Coherence Verification of Quality Measures A Blueprint for a General Infrastructure for Natural Language Processing Systems Evaluation Using Semi-Automatic Quantitative Black Box Approach in a Multilingual Environment. European project LE4-8340ELSE: Evaluation in Language and Speech Engineering. 1999.

[Quinlan, 1997] Quinlan, J.R. Data Mining Tools See5 and C5.0. http://www.rulequest.com/see5-info.html.

[Quinlan and Cameron, 1993] Quinlan, J.R., and Cameron-Jones, R.M. FOIL: A midterm report. In Machine Learning: ECML-93. 1993.

[Rath *et al.*, 1961] Rath, G.J., Resnick, A., and Savage T.R. The formation of abstracts by the selection of sentences. American Documentation, 12(2):139-143, April 1961. Reprinted in [Mani and Maybury, 1999].

[Rau, 1987] Rau, L. Knowledge organization and access in a conceptual information system . In Information Processing and Management. 1987.

[Robnik-Sikonja and Kononenko, 1997] Robnik-Sikonja, M., and Kononenko, I. An Adaptation of Relief for Attribute Estimation in Regression. In Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97), 296-304. 1997.

[Sparck Jones, 1998] Sparck Jones, K. Automatic summarising: factors and directions. In [Mani and Maybury, 1999].

[TIPSTER, 1998] Mani, I., Firmin, T., House, D., Chrzanowski, M., Klein, G., The TIPSTER SUMMAC Text Summarization Evaluation: Final Report. Hirschman, L., Sundheim, B., Obrst, L. MITRE Technical Report MTR 98W0000138. 1998.

[Torgo and Gama, 1997] Torgo, L., and Gama, J. Search-Based Class Discretization In European Conference on Machine Learning, 266-273. 1997.

[Turney, 2000] Turney, P. Learning Algorithms for Keyphrase Extraction. Information Retrieval, (2000), 2 (4), 303-336. 2000.

[Woods, 1973] Woods, W.A. Progress in NLU–An application to lunar geology. In AFIPS Conference Proceedings 441-450. 1973.

[Zeller *et al.*, 1995] Zell, A., Mamier, G., Vogt, M., Mache, N., Hbner, R., Dring, S., Herrmann, K.-U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeorgiou, A., Posselt, D., Schreiner, T., Kett, B., Clemente, G., and Wieland, J. Stuttgart Neural Network Simulator, User Manual, Version 4.1. University of Stuttgart, Institute for parallel and distributed high performance systems (IPVR). Report No. 6. 1995.