SUMMA

A Robust and Adaptable Summarization Tool

Horacio Saggion

Department of Computer Science
University of Sheffield
211 Portobello Street - Sheffield, England, UK
saggion@dcs.shef.ac.uk
Tel: +44-114-222-1947
Fax: +44-114-222-1810

ABSTRACT. *Over the last few years there has been substantial research on text summarization, but comparatively little research has been carried out on adaptable components that allow rapid development and evaluation of summarization solutions. This paper presents a set of adaptable summarization components together with well-established evaluation tools, all within the GATE paradigm. The toolkit includes resources for the computation of summarization features which are combined in order to provide functionalities for single-document, multi-document, query-based, and multi/cross-lingual summarization. The summarization tools have been successfully used in a number of applications including a fully-fledged information access system.*

RÉSUMÉ. *Au cours des dernières années il y a eu un nombre important de recherches au sujet du résumé automatique. Toutefois, il y a eu comparativement peu de recherche au sujet des ressources computationnelles et composantes qui peuvent être adaptées facilement pour le développement et l'évaluation des systèmes de résumé automatique. Ici on présente un ensemble de ressources spécifiquement développées pour le résumé automatique qui se basent sur la plateforme GATE. Les composantes sont utilisées pour calculer des traits indiquant la pertinence des phrases. Ces composantes sont combinées pour produire différents types de systèmes de résumé tels que résumé de document simple, résumé de document multiple, et résumé basé sur des topiques. Les ressources et algorithmes implémentés ont été utilisés pour développer plusieurs applications pour l'accès à l'information dans des systèmes d'information.*

KEYWORDS: *Summarization, Summarization Toolkit, Adaptable Summarization, SUMMA, GATE*

MOTS-CLÉS : *Résumé automatique, Systèmes de résumé automatique, Système générique de résumé automatique, SUMMA, GATE*

1. Introduction

Text summarization is an information access technology which, given a document or set of related documents, extracts the most important content from the sources and presents it in a well-formed and concise textual form. The ever-increasing availability of on-line textual databases and the development of the Web require effective tools and resources to assist humans in the creation of summaries and other types of document surrogates in order to facilitate information access. Recent years have witnessed many advances in information technology; however, summarization is still a field where significant improvements are required. For example, although automatic systems are outperforming simple baselines, their performance is still behind that of humans (Over *et al.*, 2007). Many summarization solutions have been produced in the long history of the field, from the seminal work of Luhn (Luhn, 1958) and Edmundson (Edmundson, 1969) to the recent advances presented in the special issue of *Information Processing & Management* on text summarization (Harman, 2007). To the best of our knowledge, very little research has been carried out on the construction of a standard set of tools universally available which can be used for creating or customizing summarization systems, carrying out experiments, and evaluating different possible configurations. Such a toolkit would ensure replicability, comparison of results, and sharing of language and processing resources across laboratories. The only exception to this has been the MEAD system (Radev *et al.*, 2004). In most cases, research on tools and software for natural language processing (NLP) has concentrated on tasks other than text summarization (see, for example, a recent issue of the *Natural Language Engineering* journal (Cunningham and Scott, 2004)).

Sets of tools and platforms for NLP have been produced and are well documented, notably the General Architecture for Text Engineering, the GATE platform (Cunningham *et al.*, 2002), which has been used in many natural language processing tasks, notably information extraction. However, GATE in itself does not provide modules for text summarization or its rather idiosyncratic evaluation methods. In this paper we describe our experience in the creation of a set of standard summarization tools – the SUMMA toolkit – under the GATE platform. We also describe practical applications developed with the toolkit, but the focus of the paper is on the toolkit itself. Because the implementation of SUMMA is based on GATE, many functionalities are inherited and resources reused to create fully-fledged summarization applications. There are many advantages in having a set of summarization components integrated in the GATE platform, including rapid development or testing of research ideas, evaluation, replicability of results, education, and training.

This article is structured as follows: first, in Section 2, we give a brief overview of automatic text summarization. This is followed in Section 3 by a very brief description of the GATE platform and the implementation of the SUMMA toolkit as a set of GATE components. Section 4 overviews summarization evaluation, and Section 5 describes the SUMMA content-based evaluation framework. In Section 6 we show how summarization systems can be trained using the GATE and SUMMA tools, while in Section 7 we describe a number of real applications developed with the tool. In Sec-

tion 8 we present related work, and Section 9 closes with conclusions and prospects for further work.

2. Automatic Text Summarization

According to Mani (2001), the goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the system's user in a condensed form which is sensitive to the user's task. There are two main problems in text summarization: one is the problem of selecting the most relevant information from a source document or documents, the second is how to express that key information in the final summary. A compression parameter is usually specified as part of the process. This can be an absolute number of words to be produced or a percentage of the original text. Sometimes the task of content selection is too open (e.g., "select the most important content of the document") and may hinder the development of satisfactory summarization solutions. Usually selection of information depends not only on the document to be summarized but also on contextual factors such as the audience (e.g., does the reader need background information?) or the tasks the user/reader has to carry out with the summary (e.g., indexing, classification, question answering). When minimal requirements are specified in a summarization task (e.g., summarize this document in 200 words), we talk about generic summarization, meaning that the perspective of the author of the document would be taken into account when deciding on what content to select from the source.

Most attention has been paid to the problem of what to extract from the source document; however, research also has been carried out on how to create well-formed summaries.

Approaches to the first problem have tried to come up with a list of relevant features that are believed to indicate the relevance of a sentence in a document or set of documents (see Mani (2001)). These features can be used alone or in combination to produce sentence (or paragraph) relevance scores which in turn are used to rank sentences/paragraphs and select the top-ranked ones (up to a certain compression) as the resulting summary content (e.g., an extract). For example, in cases where features are used to learn a selection function, the scoring is some sort of probability which indicates the likelihood that the sentence belongs to a summary. Various interpretations exist of how to compute the features.

The simplest strategy for presenting the summary is a concatenation strategy which outputs the selected sentences in order of occurrence in the document. In multi-document summarization settings particular attention has to be paid to the way in which information is to be presented (Barzilay *et al.*, 2001; Lapata, 2003). The generation of extracts has the obvious disadvantages of producing fragmentary texts exhibiting problems of cohesion and coherence (Paice, 1990), but there are no robust and generic solutions to produce non-extractive multi-sentence summaries from open-domain texts.

Some features investigated in the literature include the use of frequency information (term frequency or tf) combined with inverted term frequency (or idf) from corpus statistics. This implements the idea that the more frequent a term is in a document – and at the same time more discriminative – the more relevant it is and as a consequence it is worth selecting sentences containing the term (Luhn, 1958).

The position of a sentence in the document has also informed content selection strategies (Edmundson, 1969; Lin and Hovy, 1997). News wire and similar texts have a pyramidal discourse structure where relevant/new information (the key events) is usually expressed in the leading paragraph, and therefore sentences from the leading paragraph are considered relevant. In scientific papers, sections such as the introduction and the conclusion report on the objectives and findings of the research, so sentences contained in those sections can be considered as important.

Because titles of articles, especially scientific ones, sometimes state the theme or subject dealt with in the article, one may consider that sentences containing terms from the title are relevant for a summary (Edmundson, 1969).

The presence of specific formulaic expressions, especially in the scientific and legal domains, has also been considered a useful cue to find information relevant for a summary (Paice, 1981). Formulaic expressions such as "This paper presents," "Our results indicate," "We conclude that" are used by paper authors to explicitly introduce key information. In fact, this is also a strategy used by professional abstractors to produce abstracts, a fact acknowledged in abstract writing studies (Cremmins, 1982; Endres-Niggemeyer *et al.*, 1995).

In the context of creating a summary of multiple documents, one feature usually exploited, and proven to work well, is the proximity in content of a sentence to the centroid of the set of documents to be summarized (Radev *et al.*, 2000; Saggion and Gaizauskas, 2004), which is a set of words in a cluster of documents considered statistically important.

These are only a few of the indicators which may be used by a summarization algorithm in the selection of sentences or passages for a summary; these features together with new functionalities have been implemented in the SUMMA toolkit (a full list of resources is given in the Appendix). They are continuously used by the research community and thus they seem to be necessary in order to create baselines and explore new research ideas. These are simple to compute, reliable, and have been the focus of attention of a number of studies in human summarization (Endres-Niggemeyer, 2000).

3. SUMMA Implementation

The components have been developed at the NLP Group in the University of Sheffield and are implemented as a set of GATE processing and language resources (i.e., algorithms and data). The first set of components was described in

(Saggion, 2002), and since then the tool has been transformed to cope with new summarization challenges and to address real summarization tasks, as will be described later.

3.1. *GATE*

The GATE platform is a framework for the development and deployment of language processing technology on a large scale (Cunningham *et al.*, 2002) and in recent years has been extended to cope with new challenges posed by the information society, notably semantic access to information (Bontcheva *et al.*, 2004). The GATE architecture is based on components, reusable pieces of software that may be deployed in a variety of contexts. GATE components may be implemented by a variety of programming languages, but they are all represented as Java classes. GATE provides three types of resources: Language Resources (LRs), which collectively refer to data; Processing Resources (PRs), which refer to algorithms; and Visualisation Resources (VRs), which represent visualization and editing components. The set of resources integrated with GATE is known as CREOLE: a Collection of REusable Objects for Language Engineering. A *creole.xml* file contains the specification of each of the resources available to GATE in a given plug-in.

GATE can be used to process documents in different formats including plain text, HTML, XML, RTF, and SGML. The documents in GATE are language resources that contain text and one or more annotation sets which are used to encode different types of information. Another language resource is a GATE corpus, which is a set of GATE documents. In order to create applications in GATE or in a program that uses the GATE library, one has to combine processing resources into an application. There are two ways to do this in GATE. An application can be created using a pipeline that groups processing resources together and executes them in sequence. Another way to create an application is by using a corpus pipeline, which also groups a set of resources together, but runs over a corpus. A corpus pipeline iterates over each document in the corpus, applying each processing resource in turn.

Document annotations are generally created and/or updated by PRs during text processing (e.g., application execution). Each annotation belongs to an annotation set and has a type, a pair of offsets (the span of text one wants to annotate), and a set of features and values that are used to encode various types of information. Features (or attribute names) are generally strings. Attributes and values can be specified in an annotation schema, which facilitates validation and input during manual annotation. Programmatic access to the annotation sets, annotations, features, and values is possible through the GATE Application Program Interface (API). Some typical components of GATE are a tokenizer (which produces token annotations), a sentence splitter (which produces sentence annotations), a part-of-speech tagger (which adds a part of speech tag to each token), and a named entity recognition module (which identifies different types of named entities and creates annotations for each type). Most native GATE components are exploited in one form or another in the SUMMA package.

3.2. *SUMMA Toolkit*

The summarization toolkit uses some of the default GATE components and also uses GATE language resources to store computed values. Algorithms for the computation of summarization features and evaluation methods have been implemented as GATE processing resources, and therefore methods for initialization and execution over corpora have been implemented. Processing resources in SUMMA have parameters that can be modified by the user (e.g., summary compression, scoring values, features to be combined).

Language resources are also used, notably in the creation of sentence extracts. The system uses an annotation set to indicate (as annotations) the sentences selected for a summary, and if required creates a stand-alone GATE document to store the extracted text; in the case of multidocument summarization a multidocument summary is also created as a GATE document.

The sentence annotations produced by the GATE sentence splitter are used to store values computed by SUMMA and can be exported for use in statistical or symbolic learning.

Annotation sets containing the extract can be exported to a repository on disk for further use or can be exported together with the GATE documents in order to produce highlighted texts. Summarization components can be combined by the user to produce an application (pipeline or corpus pipeline of processing resources). Therefore, the tool provides an easy way for the development, testing, and deployment of customizable summarization solutions. The resources are specified in a CREOLE file so that they can be created, combined, and executed in GATE as any other resource. Because SUMMA uses language resources from GATE, visualization of results is possible within GATE.

Processing resources compute numerical features for each sentence in the input document, which indicate how relevant the information in the sentence is for the feature. The computed values are combined – using a specific processing resource – in a linear formula to obtain a score for each sentence, which is used as the basis for sentence selection. The formula can be edited by the user in a flexible way: features can be removed or added and weights can be adjusted. Sentences are ranked based on their score and top-ranked sentences selected to produce an extract, i.e., annotated in an annotation set; if specified, a GATE document containing the summary is also created. The SUMMA tool can be used in the GATE user interface or standalone by importing in a Java program the SUMMA class library.

The features can also be used in order to train a sentence classification program and the trained system used to produce summaries for unseen documents (this is detailed below).

An example summary obtained with the tool can be seen in Figure 1. The Figure shows the GATE graphical user interface, where a number of documents (left-hand side of the figure) have been analyzed by SUMMA components. One of the documents

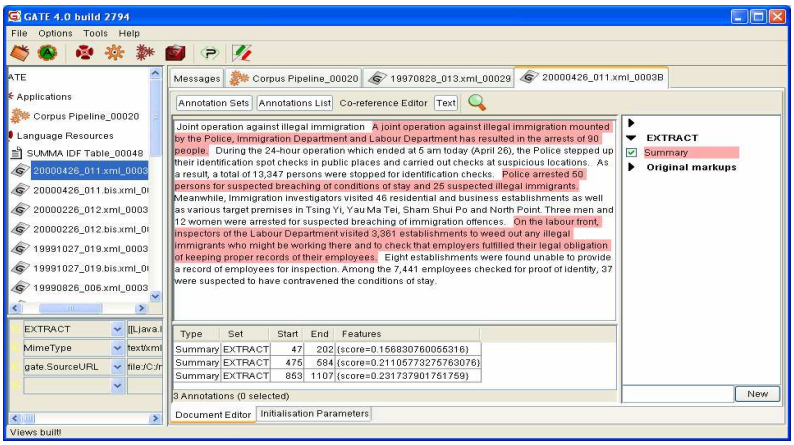

Figure 1. *GATE GUI showing extract computed by SUMMA*

is being displayed in the GATE GUI. The text of the document (top middle panel) has a number of sentences which have been highlighted or annotated as summary sentences. The score associated to each of the annotated sentences can be seen in the bottom middle panel: the score is represented by the feature 'score=...'. The annotated sentences (which are called 'Summary') are GATE annotations which are stored in an annotation set called 'EXTRACT' (see right-hand side panel).

These sentences can be exported to a text file using a component provided with the summarization toolkit.

A corpus statistic module computes token statistics including term frequency – the number of times each term occurs in the document (tf). Token statistics are stored in the token annotations produced by the GATE tokenizer. The vector space model (Salton, 1988) has been implemented and is used to create vector representations of different text fragments – usually sentences but also the full document. Each vector contains, for each term occurring in the text fragment, the value tf*idf (term frequency * inverted document frequency). The inverted document frequency of a given term is the number of documents in a collection containing the term; these values are found in a language resource which implements an inverted document frequency table. The table can be created from a file on disk or computed on the fly for the current corpus to be summarized. With the latter option, the values can then be saved for future use. Statistics can also be computed for units other than tokens (e.g., named entities) and used to associate frequency information to them.

A term frequency resource computes the sum of the tf*idf of all terms in each sentence (note that because frequent terms such as 'the' have nearly zero idf values, their contributions to the term frequency feature are minimal). These values are normalized to yield numbers between 0 and 1.

A content analysis module is used to compute the similarity between two text fragments in the document represented in the vector space. The measure of similarity is the cosine of the angle between the two vectors given by the *Cosine* formula:

$$Cosine(X, Y) = \frac{\sum X_i * Y_i}{\sqrt{\sum (X_i)^2} * \sqrt{\sum (Y_i)^2}}$$

where X and Y are text representations based on a vector space model implemented in SUMMA and X_i and Y_i are the weights of the i term in the representation. There are various ways in which SUMMA uses this similarity computation; one is to compute the similarity of the title of the document to each sentence (title method), another one is to compute the similarity of each sentence to a particular user query (query-based method), yet another is to compute the similarity of each sentence to the first sentence of the document; finally for each sentence the similarities forward (with following sentence) and backward (with previous sentence) can also be computed.

The sentence position module computes two features for each sentence: the absolute position of the sentence in the document and the relative position of the sentence in the paragraph. The absolute position of sentence i receives value i^{-1} while the paragraph feature receives a value which depends on the sentence being in the beginning, middle or end of paragraph. These values are parameters of the system and can be modified by the user.

For a cluster of related documents, the system computes the centroid of the set of document vectors in the cluster. The centroid is a vector of terms and values which is in the center of the cluster. The value of each term in the centroid is the average of the values of the terms in the vectors created for each document.

The similarity of each sentence in the cluster to the centroid is also computed using the procedure stated above. This value is stored as a sentence feature and used during sentence scoring in multi-document summarization tasks (Saggion and Gaizauskas, 2004). Note that the processing resource implementing this algorithm requires a set of documents (i.e., GATE corpus) as a parameter.

Figure 2 shows the document annotated with different summarization features and the score produced by the summarization toolkit. The Figure shows again the GATE GUI; here a document which has been processed with SUMMA tools is presented. These sentences have been enriched with features (*paragraph_score*, *position_score*, *query_sim*, etc.) and values computed by different components. These features are stored as sentence features and used to produce the sentence final score (feature score).

The final score for a sentence is computed using the following formula:

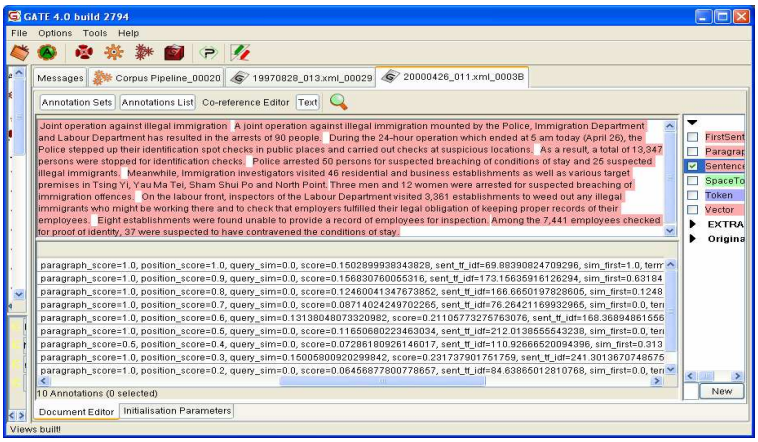

Figure 2. *GATE GUI showing sentence features computed by SUMMA.*

$$\sum_{i=1}^{n} feature_i * weight_i$$

where $feature_i$ are features computed by SUMMA and the weights ($weight_i$) are obtained experimentally. These components can be changed by the user and constitute parameters of the summarization process (the summarizer comes with pre-established weights). Feature names are constants specified in the SUMMA resources.

3.3. *Language Resources*

In text summarization, corpus statistics information is often used since the distribution of terms in documents together with the distribution of terms in a collection can be used as indicators of term-hood and therefore of content relevance. An inverted document frequency (IDF) table provides information about the distribution of terms in a document collection. In SUMMA, this is implemented as a language resource, created from a text file which records information about the corpus size and the number of documents each term appears in. IDF tables are used to compute statistics over documents to be summarized by the statistics module. Although tables can be created by any external program, they can also be created within SUMMA: given a tokenized corpus, a table can be created on the fly and stored for later processing, or the table can be used after creation by the statistics module.

An n-gram computation algorithm has also been implemented which creates n-grams annotations for the documents, and these are stored in annotation sets. These n-grams are used in SUMMA to support content-based evaluation as well as to calculate similarity between text fragments, thus supporting redundancy detection during multi-document summarization.

3.4. *Multi-document Summarization in SUMMA*

SUMMA implements multi-document summarization in a similar way to single document summarization by computing a score for each sentence in the input document cluster. Multi-document features are usually combined with single-document features to produce the sentence scores. The multi-document feature used in SUMMA is the centroid feature, which is very effective for this task. In addition to scoring sentences in multiple documents, a method has been implemented for measuring the similarity between sentences and to detect redundancy – a specific requirement in multi-document summarization (see Section 5.1). Another requirement of multi-document summarization is a method to create the final summary, i.e., how to order the resulting set of selected sentences. Two methods are provided: one concatenates the sentences according to the scores they had obtained; and the second depends on temporal and positional order, i.e., it takes into account the date of the document (stored as a feature in the document) and the position of each sentence within the document. Thus information in earlier (or later) documents is presented first (and sentences in order of occurrence). A GATE document is created with the selected sentences, which are annotated with features describing the document they came from.

3.5. *Ready-made Summarization Applications in SUMMA*

GATE comes with an information extraction application called ANNIE (A Nearly New Information Extraction System), which provides a pipeline of processing resources; the ANNIE system can be created with default or user-specified parameters and constitutes a typical pipeline for named entity recognition and tracking. This system allows users not only to understand how an application can be created within the platform, but also to rapidly test it using a new document sample. It has the added advantage of providing a baseline information extraction system other systems will be compared to.

In the same spirit as ANNIE, SUMMA implements three different summarization applications: a single-document summarization system, a query-based summarization system, and a centroid-based multi-document summarization system. All three systems are created from a list of components loaded by the system and sequenced in a pipeline. In the case of a query-based summarization system, a runtime parameter represents the query which is implemented as a string. In a typical situation, a set of documents matching a query is retrieved from Google or Yahoo! and then summarized with the application; all this can be easily implemented in the GATE platform thanks

to the availability of plug-ins for Web access. Components are loaded with default parameters which the user can modify.

4. Summarization Evaluation

Evaluation of summaries is a complex issue, and in the last decade much research has been devoted to this problem. While research in text summarization has always presented evaluation results of one type or another, ten years ago the first system-independent evaluation took place: the SUMMAC evaluation (Mani *et al.*, 1998). This event was very significant for the research community, because for the first time systems were compared and measured using the same yardstick.

Two types of evaluation are generally considered in summarization (Sparck Jones and Galliers, 1995). In an intrinsic evaluation the summaries produced are evaluated in terms of whether they contain the main topics of the source and whether they are acceptable texts. Variables measured can be the proportion of key information covered or precision/recall statistics as well as grammaticality and coherence scores. The content of an automatic summary can be compared to the content of a human-produced summary in terms of word or "propositional" overlap. In an extrinsic evaluation, the summaries are evaluated in a concrete task seeking to verify if the summaries are instruments which could be used instead of full documents in specific situations. Variables measured can be accuracy in performing a task and time to complete the task. While extrinsic evaluation is very attractive from the point of view of information access, it is also very time-consuming and costly, so its application has been limited. In recent years large-scale evaluation of summarization systems has been carried out under the auspices of the National Institute for Standards and Technology with the Document Understanding Conferences programme (Over *et al.*, 2007) (and now by the Text Analysis Conferences - TAC)[1]. The approach used is mainly intrinsic, although some efforts to simulate specific information access tasks have also been undertaken.

In the context of single document-summarization, the traditional metrics used for evaluating sentence extraction systems are precision and recall (Firmin and Chrzanowski, 1999). Precision is the ratio of the number of summary-sentences identified by the system to the number of sentences; recall is the ratio of summary-sentences identified by the system to the number of true summary sentences. Precision and recall are currently being questioned by researchers because they fail to measure content coverage, since they only take into account the identity of a sentence and not its content. Nonetheless, in spite of their limitations, they still serve as the basis for current "content-based" fine-grained metrics adopted by the research community.

Current summarization evaluation methods rely on content-based metrics, which compare the content of an automatic summary to the content of a human summary and produce a similarity score. A high score indicates that the automatic summary is

1. `http://www.nist.gov/tac/`

close in content to the human summary (e.g., a sentence such as "Three people were killed in the blast" and "In the blast three were killed" should be considered close in content in spite of their differences). Some metrics used are word or lemma overlap, cosine similarity (which treats documents as vectors of terms), and longest common subsequence between sentences (which takes into account the number of minimal transformations to transform one sentence into another).

In GATE, evaluation metrics used in information extraction contexts, such as precision, recall, and f-score, are available and can be used to evaluate summaries created by SUMMA. Because of the way in which summaries are created – as summary annotations in a particular annotation set – the computation of precision, recall, etc. is quite simple using the annotation Diff package in GATE. In order to compare summaries produced by different methods, including human extracts, using the GATE evaluation framework, it is sufficient to produce different annotation sets, one per summary to be compared.

5. Content-based Evaluation Modules in SUMMA

In the SUMMA package, a number of evaluation metrics based on vocabulary overlap have been implemented as processing resources which output evaluation reports.

5.1. *Content-based metrics*

In our research at the Johns Hopkins Workshop in Language Engineering (Saggion *et al.*, 2002a), we investigated measures for content evaluation based on the notion of vocabulary overlap. They were developed to reduce the problems with precision and recall. As they are completely automatic, they overcome the problems of task-based evaluations and can be used in both extractive and non-extractive summarization, single and multi-document summarization. Past research has shown that content-based evaluation can be carried out in an automatic or semi-automatic fashion (Donaway *et al.*, 2000; Paice and Oakes, 1999). Content-based similarity measures are functions that take as arguments two text representations and compute a real value in the interval $[0..1]$; the value 1 means that the two texts are closely related while the value 0 means that the two texts are quite different. SUMMA implements various measures; for example the *Cosine* metric is used to measure the similarity between two texts in the vector space and used to compare system with ideal summaries. Also implemented is a unit overlap metric, which is computed using the following formula:

$$overlap(X, Y) = \frac{\|X \cap Y\|}{\|X\| + \|Y\| - \|X \cap Y\|}$$

where X and Y are text representations based on sets. Here $\|S\|$ is the size of set S. These sets contains n-grams from the document. A weighted n-gram metric is also computed using the following formula:

$$NgramSim(T_1, T_2, n) = \sum_{k=1}^{n} w_k * \frac{|grams(T_1,k) \cap grams(T_2,k)|}{|grams(T_1,k) \cup grams(T_2,k)|}$$

where n means that n-grams 1, 2, ... n are to be considered, $grams(T, k)$ is the set of k-grams of fragment T, and w_k is the weight associated with the k-gram similarity of two sets where the weights are uniform and computed as follows:

$$w_i = \frac{i}{\sum_{j=1,2,...n} j} \qquad \text{for } i = 1, 2, ..., n$$

5.2. *BLEU and ROUGE Evaluation*

BLEU is an intrinsic evaluation measure used in Machine Translation which compares the content of a machine translation against an "ideal" translation. The metric has been found to correlate closely with human judgments, being reliable even when run on different documents and against different numbers of model references. It is based on a "weighted average of similar length phrase matches" (n-grams), it is sensitive to longer n-grams (the baseline being the use of up to 4-grams), and it also includes a brevity penalty factor for penalizing translations which are shorter than the "gold standard" translations (Papineni *et al.*, 2001). In Lin and Hovy (2002) BLEU has been applied to summarization. The authors note the unstable and unreliable nature of manual evaluation and the low agreement levels between humans on the contents of a reference summary. In a number of experiments we have carried out using BLEU in the evaluation of summarization systems (Pastra and Saggion, 2003), we have found that using a single reference affects the reliability of the results provided by the metric, so multiple references should generally be used. We have also found that estimating the average rank of a summarization system might yield consistent and reliable results even with a single reference summary and therefore compensate for lack of multiple reference summaries. We have implemented BLEU in the SUMMA package as a processing resource that given as parameter a summary or extract, an n-gram parameter (e.g., 1, 2, 3, 4), and a set of reference summaries (in a corpus), computes the IBM Blue score between each document and each reference summary and produces a report which can then be used for further statistical testing. The exact formula used for computing the BLEU score is as follows:

$$Bleu(S, R) = K(S, R) * e^{Bleu_1(S,R)}$$

$$Bleu_1(S, R) = \sum_{i=1,2,...n} w_i * \lg(\frac{|(S_i \bigcap R_i)|}{|S_i|})$$

$$K(S,R) = \begin{cases} 1 & \text{if } |S| > |R| \\ e^{(1-\frac{|R|}{|S|})} & \text{otherwise} \end{cases}$$

$$w_i = \frac{i}{\sum_{j=1,2,...n} j} \qquad \text{for } i = 1, 2, ..., n$$

where S and R are the system and reference sets. S_i and R_i are the "bags" of i-grams for system and reference. n is a parameter of the implementation.

ROUGE (Lin and Hovy, 2003) is a recall metric which also considers n-grams as units for comparing automatic and human summaries. We have implemented a language resource which computes ROUGE-n values, where n is a parameter of the resource. The formula we use is the one specified in Lin (2004).

$$\textit{ROUGE-n(Summary,Ideals)} = \frac{\sum_{\text{I}\in\text{Ideals}} \sum_{\text{n-gram}\in\text{Summary}} \text{count}_{\text{match}}(\text{n-gram}))}{\sum_{\text{I}\in\text{Ideals}} \sum \text{count(n-gram)}}$$

where Summary is the summary to be evaluated, Ideals is the set of ideal (human) summaries, $\text{count}_{\text{match}}$ is the number of common n-grams in I and Summary, and count is the number of n-grams in the ideal summaries. The evaluation produces a report comparing each system summary against a set of ideal summaries, and this information can be used to compute confidence intervals for system ranking and clustering. Some ROUGE configurations have been shown to correlate with rankings based on content produced by humans. Future versions of SUMMA will include the whole ROUGE battery (e.g., ROUGE-LCS, ROUGE-W).

6. Learning Framework for Summarization

SUMMA follows the Edmundsonian summarization paradigm (Mani, 2001), in which features are combined with specific weights in order to score sentences; however, SUMMA can be used for learning summarization systems in a way similar to the one introduced in (Kupiec *et al.*, 1995) and depicted in Figure 3. In this framework a set of documents and summaries is used to create an aligned corpus (A, B, and C) – this can be achieved using aligned techniques described in Jing and McKeown (1999); Marcu (1999) – or a set of documents with extracts used if available (C); thus one knows for every sentence in the document whether it is a summary sentence or not (this is the target information to be learned). A program is run over the corpus to

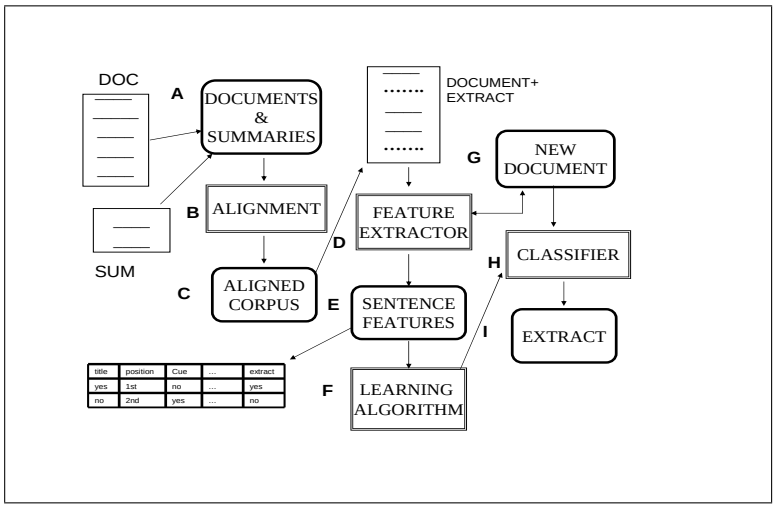

Figure 3. *Framework for Learning Sentence Extraction Systems.*

compute summarization features (D and E). The corpus is used to train a classifier (F and H). Finally, a new document (G) is processed by the feature extraction program (D) and fed into the classifier (H) to obtain a summary (I).

Using GATE and SUMMA functionalities, this paradigm can be easily instantiated following these steps:

1) A corpus of documents with annotated extracts is loaded in GATE. This can be instantiated by using the SummBank corpus, where each sentence in a document cluster contains relevance judgments assigned by humans (see Saggion *at al.* (2002b)). There are two possibilities when using relevance judgments: one possibility is to use the judgment itself as the target (or class) of the learning program, the other is to create extracts at certain compression rates (e.g., 10, 20, 30 percent) and consider as a feature the relevance judgment (e.g., a sentence belongs to the extract if its relevance is greater than a threshold), and in this case yes/no judgements are used for learning. This implements step C.

2) The corpus is processed using SUMMA tools in order to produce summarization features; these features can be used as they are, or can be transformed into discrete values. This implements steps D and E.

3) One of the machine learning algorithms available in GATE – Support Vector Machines or Naïve Bayes Classifiers – is invoked for a sentence classification task. The features to be used and the class to be learned have to be specified in a configuration file. This implements steps F and H.

4) The classifier is saved and used for new documents which will have features computed by SUMMA. This implements sequence G, H, and I.

Note that because SUMMA computes summarization features as GATE annotations, the communication between SUMMA and the learning component is facilitated by GATE. Additionally, the computed features may be exported for use in any other application outside GATE. Thus we can see that the SUMMA system provides all necessary components to follow the learning paradigm, and they are all integrated into one platform.

7. Practical Applications and Multilinguality

The SUMMA summarization toolkit has been used in two fully-implemented information access systems as well as in international evaluations on text summarization, demonstrating the flexibility and adaptability of the toolkit. Here we give a brief description of these applications. The multilingual capabilities of the system inherited from GATE made possible the use of SUMMA to provide cross-lingual extracts from a large multi-lingual corpus in the Clarity project (Demetriou *et al.*, 2004). In addition to the English summarization system, adaptations were undertaken to produce summaries for Latvian, Swedish, and Finnish. The adaptation for each language consisted in the use of specific tokenizers and sentence splitters implemented in GATE and statistical tables for each language; all summarization-specific modules were used without the need for language-specific adaptation.

In the Cubreporter information access system (Gaizauskas *et al.*, 2007), the tool was also used to provide single and multi-document summarization functionalities. The Press Association news archive (over 8 million news stories) was single-document summarized using SUMMA in order to provide access points during document retrieval. Multi-document summaries computed off-line were provided for all sets of related documents in the text collection. In addition, an on-line multi-document summarization function was made available to the user in order to summarize sets of documents selected during system interaction.

Customized summarization solutions were also used for the Document Understanding Conferences (Saggion and Gaizauskas, 2004; Saggion, 2005), where our centroid-based multi-document summarization system performed very well in the competition, and for the multilingual summarization evaluation competition (MSE 2005), where summarization in Arabic was carried out (Saggion, 2006), again using the multilingual capabilities that GATE provided, including tokenization and sentence recognition in Arabic. The tool was also used to support experiments on clustering for cross-source coreference (Saggion, 2008). In addition to these practical applications, SUMMA has also been used for training and NLP courses to demonstrate text summarization concepts.

8. Related Work

There have been a number of summarization components described in the literature including SUMMARIST (Hovy and Lin, 1999), SweSum (Dalianis *et al.*, 2004), FEMSum (Fuentes *et al.*, 2006), and ContextO (Crispino and Couto, 2004). An implementation of text summarization using the GATE platform was produced by (Lal and Ruger, 2002), but it has not been made available or developed any further.

The SUMMARIST system was one of the first tools to implement and combine a number of methods for feature computation, including the position method (Lin and Hovy, 1997), cue-phrase method, term frequency method, and the topic signatures method (Lin, 1998). To the best of our knowledge, apart from extensions to cope with multiple languages, the system has not been further developed or made freely available.

The SweSum system is being developed with the challenges of processing Scandinavian languages, although it also produces summaries for English. It is worth noting that SUMMA has been adapted to Scandinavian languages for the CLARITY project. Summarization methods implemented in SweSum are the position method (absolute and relative), the identification and scoring of sentences based on the presence of named entities or numerical data, and the scoring of sentences based on the presence of typographical features (e.g., headlines). As in SUMMA, features are combined with appropriate weights in a simple formula or used in a decision tree trained over manually created extracts. It is thus similar to SUMMA; however, in spite of offering a Web interface for summarizing Web pages, it has not been made publicly available.

FEMsum (Flexible Eclectic Multi-task Summarizer) is an architecture for automatic text summarization which allows researchers to investigate text summarization in different contexts. In the same spirit as SUMMA, it has been used to create different systems for different summarization tasks (e.g., for the DUC evaluations), achieving some degree of success. The specification of the architecture includes three main components: a relevance information detector, which ranks sentences or units; a content extractor, a set of linguistic analyzers including named entity detection; and a summary composer. Unlike SUMMA, whose components are specified as GATE resources and use annotations as an underlying communication mechanism, it is unclear how composition and communication between components are achieved in FEMsum.

ContextO is based on work on "contextual exploration" (Minel *et al.*, 2001). The platform integrates text summarization with a visualization and navigation module, allowing users to create "dynamic" summaries in the sense that the type of information to extract, the compression rate, and the modes of exploration of the summary can be specified. The summary text is coupled with the source document; thus the user has access to the context of specific pieces of information. Semantic information is identified by hand-coded contextual rules that are fired when specific linguistic markers are found in the document. SUMMA is different from ContextO in a number of ways. ContextO concentrates on semantic information, while SUMMA implements generic summarization features. ContextO requires an extensive rule-base to detect

relevant information, SUMMA relies on a more light-weight approach to summarization. Where presentation of information is concerned, as in ContextO, SUMMA also offers summaries with access to the context of the selected sentences.

Closer to SUMMA is MEAD (Radev *et al.*, 2004), a platform developed at the University of Michigan[2] which provides a set of tools implemented in Perl to create summarization applications. As in SUMMA, MEAD implements a number of feature-computation programs (position, centroid, and query similarity), which can be combined with appropriate weights. All communication in MEAD is based on an internal XML-based schema – documents and configuration files for different tasks; this is not necessary in SUMMA. SUMMA has some advantages over MEAD, because it can make use of a number of linguistic processors available in GATE without the need for complicated conversion from one format to another. Additionally, SUMMA implementation in Java makes it usable across different platforms. As in SUMMA, evaluation functionalities such as precision, recall, and the BLEU statistics are available. MEAD has also been used in a number of applications including NewsInEssence (Radev *et al.*, 2001).

9. Conclusions and Perspectives

Summaries provide readers with abbreviated descriptions of the contents of full documents. They are used in a variety of situations, from deciding whether or not to read a document to the extraction of key information from text. While text summarization has been on the natural language processing research agenda for a long time, it has, in the past decade, gained impetus due to the increasing availability of documents in digital format and the need to filter massive amounts of information. Automatic text summarization is required in order to provide users with access points to relevant information in private (Intranets) and public (the Web) text collections through the generation of document surrogates.

This paper has presented the SUMMA toolkit, a set of summarization components based on the well known GATE platform. Although this paper has mainly concentrated on the toolkit itself, it has also shown how SUMMA was used in various information access scenarios.

SUMMA makes available well-established summarization methods through the implementation and integration of algorithms and evaluation resources in a widely used architecture, therefore facilitating the development, testing, and deployment of fully-fledged summarization solutions.

We believe this could facilitate replicability of colleagues' ideas, implementation of new ones, and evaluation in a well-established framework. Because of its integration with GATE, a framework for training and deployments is inherited.

2. The author of this paper has also been involved in the development of MEAD.

The SUMMA tool will continue to be extended in order to cope with new challenges, for example to support semantic-based summarization as well as summarization of opinions. We are currently investigating the use of machine learning techniques for rhetorical classification in technical articles. We are also studying supervised approaches to the identification of opinions and sentiments in text, and we plan to make these functionalities available in a future version of SUMMA in order to study the problem of opinion summarization. Additionally, evaluation metrics will be extended to cope with evaluation requirements imposed by international evaluation programs.

Acknowledgments

We are indebted to the three anonymous reviewers for their extensive comments, suggestions, and recommendations that helped improve this work. We thank Christopher Brewster and Adam Funk who helped us improve the quality of the paper.

Appendix

The software is distributed[3] as a *tar* file which contains a number of components. The distribution includes a *jar* file, *creole.xml* file, and a *resources* directory containing an IDF table. The *creole.xml* file specifies each of the components which are available when loading the SUMMA plug-in into GATE. One example of a processing resource specification found in the SUMMA *creole* file is given below:

```
<RESOURCE>
  <NAME>SUMMA Statistics</NAME>
  <CLASS>sumgate.resources.frequency.TermFrequency</CLASS>
  <JAR>sumgate.jar</JAR>
  <PARAMETER NAME="document"
    COMMENT="The document to be analysed"
    RUNTIME="true">gate.corpora.DocumentImpl</PARAMETER>
   <PARAMETER NAME="table"
    COMMENT="The IDF table"
    RUNTIME="true">
    sumgate.resources.frequency.InvertedTable
   </PARAMETER>
   <PARAMETER NAME="leadPercent"
    COMMENT="The percent of lead text"
    DEFAULT="10.0"
    RUNTIME="false">java.lang.Double</PARAMETER>
</RESOURCE>
```

3. http://www.dcs.shef.ac.uk/ saggion/summa/default.html

This is the resource in charge of computing statistics for each "token" in the input document. Parameters are the document to be used, the IDF table to be used (which is a language resource), and an indication of what proportion of the document is to be considered the leading part of the document.

The components distributed with the current version of SUMMA include:

– A basic document analyzer;

– Statistic computation module and vector space model representation;

– Absolute position method;

– Relative position method;

– Centroid computation and sentence-centroid similarity computation;

– Term frequency method;

– Title method (similarity to title);

– First sentence method (similarity to first sentence);

– Query method (similarity to a query);

– Cue-phrase method;

– Named entity method (relevance of named entities in sentence);

– Feature combination;

– N-gram computation;

– Content-based evaluation metrics;

– Summary writer.

Documentation on how to produce a summarization application and dependencies between components (precedence) is given in a user guide distributed with SUMMA. The only mandatory annotations required by SUMMA are sentences and tokens that can be created by available GATE components (i.e., the tokenizer and sentence splitter). Implementing new summarization features in SUMMA is done in the same way as implementing a GATE processing resource; however, in order to comply with the way in which SUMMA operates, the computed feature should be stored as a sentence feature.

10. References

Barzilay R., Elhadad N., McKeown K., "Sentence Ordering in Multidocument Summarization", *Proceedings of the 1st Human Language Technology Conference*, San Diego, California, USA, 2001.

Bontcheva K., Tablan V., Maynard D., Cunningham H., "Evolving GATE to Meet New Challenges in Language Engineering", *Natural Language Engineering*, vol. 10, n° 3/4, p. 349-373, 2004.

Cremmins E. T., *The Art of Abstracting*, ISI PRESS, 1982.

Crispino G., Couto J., "Construction automatique de résumés : Une approche dynamique", *TAL. Traitement automatique des langues*, vol. 45, n° 1, p. 95-120, 2004.

Cunningham H., Maynard D., Bontcheva K., Tablan V., "GATE: A framework and graphical development environment for robust NLP tools and applications", *ACL 2002*, 2002.

Cunningham H., Scott D., "Software Architecture for Language Engineering", *Natural Language Engineering*, vol. 10, n° 3/4, p. 205-209, 2004.

Dalianis H., Hassel M., de Smedt K., Liseth A., Lech T., Wedekind J., "Porting and evaluation of automatic summarization", *Nordisk Sprogteknologi*, p. 107-121, 2004.

Demetriou G., Skadina I., H. K., Karlgren J., Deksne D., Petrelli D., Hansen P., Gaizauskas R., Sanderson M., "Cross Lingual Document Retrieval, Categorisation and Navigation Based on Distributed Services", *Proceedings of the Baltic HLT Conference 2004*, 2004.

Donaway R. L., Drummey K. W., Mather L. A., "A comparison of rankings produced by summarization evaluation measures", *NAACL-ANLP 2000 Workshop on Automatic Summarization*, Association for Computational Linguistics, Morristown, NJ, USA, p. 69-78, 2000.

Edmundson H., "New Methods in Automatic Extracting", *Journal of the Association for Computing Machinery*, vol. 16, n° 2, p. 264-285, April, 1969.

Endres-Niggemeyer B., "SimSum: an empirically founded simulation of summarizing", *Information Processing & Management*, vol. 36, p. 659-682, 2000.

Endres-Niggemeyer B., Maier E., Sigel A., "How to Implement a Naturalistic Model of Abstracting: Four Core Working Steps of an Expert Abstractor", *Information Processing & Management*, vol. 31, n° 5, p. 631-674, 1995.

Firmin T., Chrzanowski M. J., "An Evaluation of Automatic Text Summarization Systems", *in* I. Mani, M. Maybury (eds), *Advances in Automatic Text Summarization*, The MIT Press, p. 325-336, 1999.

Fuentes M., Rodríguez H., Turmo J., Ferrés D., "FEMsum at DUC 2006: Semantic-based approach integrated in a Flexible Eclectic Multitask Summarizer Architecture", *Proceedings of HLT-NAACL Workshop (DUC 2006)*, New York City, USA, June 3-8, 2006.

Gaizauskas R., Saggion H., Barker E., "Information Access and Natural Language Processing: A Stimulating Dialogue", *in* K. Ahmad, C. Brewster, M. Stevenson (eds), *Words and Intelligence II – Essays in Honor of Yorick Wilks*, Springer, 2007.

Harman D., "Editorial: Text Summarization Special Issue", *Information Processing & Management*, vol. 43, n° 6, p. 1446-1448, 2007.

Hovy E., Lin C.-Y., "Automated Text Summarization in SUMMARIST", *in* I. Mani, M. Maybury (eds), *Advances in Automatic Text Summarization*, The MIT Press, p. 81-94, 1999.

Jing H., McKeown K., "The Decomposition of Human-Written Summary Sentences", *in* M. Hearst, F. Gey, R. Tong (eds), *Proceedings of SIGIR'99 – 22nd International Conference on Research and Development in Information Retrieval*, University of California, Berkeley, p. 129-136, August, 1999.

Kupiec J., Pedersen J., Chen F., "A Trainable Document Summarizer", *Proc. of the 18th ACM-SIGIR Conference*, Seattle, Washington, United States, p. 68-73, 1995.

Lal P., Ruger S., "Extract-based Summarization with Simplification", *Proceedings of the ACL Workshop on Text Summarisation (DUC)*, Gaitherdburg,Md., USA, 2002.

Lapata M., "Probabilistic Text Structuring: Experiments with Sentence Ordering", *Proceedings of the 41st Meeting of the Association of Computational Linguistics*, Sapporo, Japan, p. 545-552, 2003.

Lin C., Hovy E., "Identifying Topics by Position", *Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, p. 283-290, 31 March-3 April, 1997.

Lin C.-Y., "Assembly of Topic Extraction Modules in SUMMARIST", *Intelligent Text Summarization. Papers from the 1998 AAAI Spring Symposium. Technical Report SS-98-06*, The AAAI Press, Standford (CA), USA, p. 53-59, March 23-25, 1998.

Lin C.-Y., "ROUGE: A Package for Automatic Evaluation of Summaries", *Proceedings of the Workshop on Text Summarization*, ACL, Barcelona, Spain, p. 74-81, 2004.

Lin C.-Y., Hovy E., "Manual and automatic evaluation of summaries", *Proceedings of the ACL-02 Workshop on Automatic Summarization*, Association for Computational Linguistics, Morristown, NJ, USA, p. 45-51, 2002.

Lin C.-Y., Hovy E., "Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics", *Proceedings of HLT-NAACL 2003*, Association for Computational Linguistics, Morristown, NJ, USA, p. 71-78, 2003.

Luhn H. P., "The Automatic Creation of Literature Abstracts", *IBM Journal of Research Development*, vol. 2, n° 2, p. 159-165, 1958.

Mani I., *Automatic Text Summarization*, John Benjamins Publishing Company, 2001.

Mani I., House D., Klein G., Hirshman L., Obrst L., Firmin T., Chrzanowski M., Sundheim B., The TIPSTER SUMMAC Text Summarization Evaluation, Technical report, The Mitre Corporation, 1998.

Marcu D., "The automatic construction of large-scale corpora for summarization research", *in* M. Hearst, F. Gey, R. Tong (eds), *Proceedings of SIGIR'99. 22nd International Conference on Research and Development in Information Retrieval*, University of California, Berkeley, p. 137-144, August, 1999.

Minel J.-L., Jean-Pierre Desclés J.-M., Cartier E., Crispino G., Ben Hazez S., Jackiewicz A., "Résumé automatique par giltrage sémantique d'informations dans des textes", *TAL. Traitement automatique des langues*, vol. 20, n° 3, p. 369-395, 2001.

Over P., Dang H., Harman D., "DUC in context", *Information Processing & Management*, vol. 43, n° 6, p. 1506-1520, 2007.

Paice C. D., "The Automatic Generation of Literary Abstracts: An Approach Based on Identification of Self-indicating Phrases", *in* O. Norman, S. Robertson, C. van Rijsbergen, P. Williams (eds), *Information Retrieval Research*, Butterworth, 1981.

Paice C. D., "Constructing Literature Abstracts by Computer: Technics and Prospects", *Information Processing & Management*, vol. 26, n° 1, p. 171-186, 1990.

Paice C. D., Oakes M. P., A Concept-Based Method for Automatic Abstracting, Technical Report n° 27, Library and Information Commission, 1999.

Papineni K., Roukos S., Ward T., Zhu W., Bleu: a Method for Automatic Evaluation of Machine Translation, Technical Report n° RC22176 (W0 109-022), IBM Research Division, 2001.

Pastra K., Saggion H., "Colouring Summaries Bleu", *EACL workshop on the Reuse of Evaluation Resources -EACL Conference 2003*, Budapest, Hungary, 2003.

Radev D., Allison T., Blair-Goldensohn S., Blitzer J., Çelebi A., Dimitrov S., Drabek E., Hakim A., Lam W., Liu D., Otterbacher J., Qi H., Saggion H., Teufel S., Topper M., Winkel A., Zhu Z., "MEAD – a platform for multidocument multilingual text summarization", *Proceedings of LREC 2004*, Lisbon, Portugal, May, 2004.

Radev D., Blair-Goldensohn S., Zhang Z., Raghavan R., "NewsInEssence: a system for domain-independent, real-time news clustering and multi-document summarization", *HLT '01: Proceedings of the First International Conference on Human Language Technology Research*, Association for Computational Linguistics, Morristown, NJ, USA, p. 1-4, 2001.

Radev D. R., Jing H., Budzikowska M., "Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies", *ANLP/NAACL Workshop on Summarization*, Seattle, WA, April, 2000.

Saggion H., "Shallow-based Robust Summarization", *Automatic Summarization: Solutions and Perspectives*, Paris, France, December 14, 2002.

Saggion H., "Topic-based Summarization at DUC", *Proceedings of DUC 2005*, Vancouver, Canada, October 9-10, 2005.

Saggion H., "Multilingual Multidocument Summarization: Tools and Evaluation", *Proceedings of LREC 2006*, Genoa, Italy, 2006.

Saggion H., "Experiments on Semantic-based Clustering for Cross-document Coreference", *Proceedings of the 3rd International Join Conference on Natural Language Processing*, AFNLP, Hyderabad, India, January 8-10, 2008.

Saggion H., Gaizauskas R., "Multi-document summarization by cluster/profile relevance and redundancy removal", *Proceedings of the Document Understanding Conference 2004*, NIST, Boston, USA, May 6-7, 2004.

Saggion H., Radev D., Teufel S., Lam W., "Meta-evaluation of Summaries in a Cross-lingual Environment using Content-based Metrics", *Proceedings of COLING 2002*, Taipei, Taiwan, p. 849-855, August 24-September 1, 2002a.

Saggion H., Radev D., Teufel S., Wai L., Strassel S., "Developing Infrastructure for the Evaluation of Single and Multi-document Summarization Systems in a Cross-lingual Environment", *LREC 2002*, Las Palmas, Gran Canaria, Spain, p. 747-754, 2002b.

Salton G., *Automatic Text Processing*, Addison-Wesley Publishing Company, 1988.

Sparck Jones K., Galliers J. R., *Evaluating Natural Language Processing Systems: An Analysis and Review*, n° 1083 in *Lecture Notes in Artificial Intelligence*, Springer, 1995.