Automatic generation of term definitions using multidocument summarisation from the web

Rafael Torralbo[1], **Enrique Alfonseca**[1], **Antonio Moreno-Sandoval**[2] and **José María Guirao**[3]

[1]Department of Computer Science
Universidad Autónoma de Madrid
e70415@estudiante.uam.es
Enrique.Alfonseca@uam.es

[2]Department of Linguistics
Universidad Autónoma de Madrid
antonio.msandoval@uam.es

[3]Department of Computer Science
Universidad de Granada
jmguirao@ugr.es

Abstract

This paper describes a new Multi-Document Summarisation architecture with which it is possible to collect and summarise documents about domain-specific terms, and produces a summary about each of the terms. It has applications in computational terminology, allowing us to provide a definition for each of the terms identified. In particular, the module will be integrated in an on-line educational system so students can use it to obtain automatically new information about the particular terminology used in their fields of study. An evaluation with computer science students shows that they appreciated the generated summaries, and most of them agreed in that it would be a useful tool in e-learning environments.

1 Introduction

Multi-Document Summarisation (MDS) is a task consisting in condensing the information from several documents in a single document with the most relevant information. The architectures and techniques used in MDS can vary depending on the characteristics of the source documents (genre, media, language, etc.) and the summary (audience, function, etc). In this work, we address the problem of applying MDS to the automatic generation of summaries from various documents from the web, with the final aim of using them in an educational system.

The purpose of the proposed system would be the following: given a list of domain-specific terms, to generate a definition of each of them. It can be thought of as a complement of Term Identification (Cabré *et al.* 01) procedures, so they are able to provide a small definition of each of the new terms identified. The application that we have in mind for the system is in the field of e-learning: quite often, students find in textbooks many domain-dependent terms that are simply introduced without much detail. We consider that a specialist module in the system capable of finding automatically information about those terms and generating a summary page would be useful for the students' learning process.

In our approach, we have used the World Wide Web as the source of documents, so the module can be applied to every domain for which there is information in the web. Given that this source is not fully reliable, as it may contain inexact or erroneous information, the students are always advised that the summary pages cannot be taken as perfect or authoritative. They are also able to access the web pages from which the summary has been generated. This may be useful if they want to discover who is the author of one of the source documents from which a dubious assertion has been extracted.

This module might have other applications, such as generating automatically reference answers to be used by programs for Computer-Assisted Assessment of free-text answers (Valenti *et al.* 03).

This paper is structured as follows: Section 2 describes related work both in the field of automatic MDS and the application of NLP techniques in educational systems. Next, Section 3 describes the procedure applied in this work, and Section 4 describes the evaluation performed and the results obtained. Finally, Section 5 summarises the conclusions and describes open lines for improving this work.

2 Related work

2.1 Multidocument summarisation

According to (Mani 01), Multi-Document Summarisation (MDS) systems usually share five steps:

1. **Identification** of the elements to be extracted from the collection.
2. **Matching** instances of these elements across the texts, to find related elements mentioned in different documents.
3. **Filtering** the matched elements, to keep the most salient ones.
4. **Compacting** them, by aggregating and generalising the information they cite.
5. **Presenting** the results, for instance, with Natural Language Generation (NLG) or with visualisation methods.

Systems usually differ in the approach chosen for some step.

Unit identification In MDS, it is common to take each sentence as a single element. All the same, some approaches work with clauses, paragraphs or documents. The compression rate is usually why different units are chosen: if it is very large, then there is generally more need to use units smaller than sentences; and, if it is small, paragraphs can be considered units.

Unit matching A very common procedure to match units from different documents consists of using a bag-of-words procedure, by characterising each unit with the set of words contained in it. The vectors can include, together with the words, their frequencies. These can also be transformed into weights, using functions such as tf·idf, χ^2 or Student's t-score. The cosine similarity, calculated by considering the two sets of words as vectors in an N-dimensional space (where N is the size of the vocabulary) is one of the most commonly used similarity functions in Information Retrieval (Salton 89). Other functions used are the scalar product or the Jaccard coefficient. Other common extension to VSM is the dimensionality reduction performed by means of Latent Semantic Analysis (LSA) (Deerwester *et al.* 90), which has also been applied in MDS systems (Ando *et al.* 00).

Unit filtering Using the similarity metrics from the Vector Space Model, it is possible to cluster the units, so those with a high degree of salient-vocabulary overlapping will be grouped together (Angheluta *et al.* 04; Erkan & Radev 04; Saggion & Gaizauskas 04). Small clusters with few representatives can also be considered not very important and can be discarded. Some approaches also try to improve the quality of the clusters by filtering the units whose similarity with all the others inside the cluster is not above a threshold (Blair-Goldensohn *et al.* 04).

Other procedure is to score the units using a cohesion-based weighting metric (Mani & Bloedorn 99). Possible cohesion relationships are identity relationships between words, synonymy, proximity, coreference and hyperonymy. Sentences whose words have many relationships with words from other units will receive a higher score, and will be selected. Yet another possibilities consist in giving higher weights to the NPs that appear in long coreference chains (Witte *et al.* 04), and in creating a graph of terms (or events), and next apply a graph-scoring algorithm (Vanderwende *et al.* 04; Erkan & Radev 04), such as Pagerank (Brin & Page 98).

In order to select the most relevant units, there are other heuristics which are also used in *single-document summarisation*, such as unit position and length, or calculating how many terms from the headline appear in every unit. (Nobata & Sekine 04) divides the documents in two groups according to the term distributions, and applies the heuristic based on the unit position just in those groups which appear to contain the key units at the beginning. In MDS, (Witte *et al.* 04) rank NPs in all the documents based on the length of cross-document coreference chains, and also give highest weights to the NPs that appear in the first units. Finally, the units with the highest-ranking NPs are selected for the summary.

Concerning the size of the units, although most systems mainly work with sentences, some of them also filter clauses and phrases in this step. Common heuristics are to remove relative clauses and appositives (Blair-Goldensohn *et al.* 04; Conroy *et al.* 04).

Unit compacting If the units have been grouped in clusters in the previous step, it can be expected that the units which are in the same cluster contain repetitive information and, thus, it should be possible to choose just one from each cluster so as to generate the summary. The unit chosen is usually the one closest to the centroid of the cluster (Blair-Goldensohn *et al.* 04).

(Barzilay *et al.* 99) uses a more sophisticated approach, by parsing all the units in each cluster with a syntactic analyser, and matching the parse trees with each other. In this matching, they use paraphrasing rules (e.g. transforming passive verbs into active verbs) to discover whether they are *compatible* units. Finally, the units that matched can be merged together with a syntax-based generation procedure.

Results generation and presentation In many cases, systems select units from the documents and put them together. At most, they perform small modifications to them, e.g. by removing relative clauses and appositions, normalising personal names, or removing dangling conjunctions. A few approaches, however, either transform the texts into a logical form (Vanderwende *et al.* 04), or apply Information Extraction procedures to fill in templates from the text (Harabagiu & Maiorano 02). In these cases, it is possible to use a Natural Language Generation system to write the summary from the extracted information.

2.2 Applying NLP techniques to e-learning

NLP techniques have been applied to e-learning applications since the sixties. The Project Essay Grader (Page 66) was probably the first system that appeared to *automatically score open-ended questions* written by students. It was later extended with part-of-speech tagging and syntactic analysis (Page 94). In recent years, there has been an upsurge of research in the field of free-text Computer Assisted Assessment (Valenti *et al.* 03), with systems such as E-rater (Burstein *et al.* 98), C-rater (Burstein *et al.* 01), Automark (Mitchell *et al.* 02), BETSY (Rudner & Liang 02) or Atenea (Alfonseca & Pérez 04).

Other common applications of NLP to e-learning are systems to generate, automatically, multi-choice questions (Mitkov & An-Ha 03; Liu *et al.* 05), or to teach and correct the grammar of sentences (Virvou *et al.* 00). Natural Language Generation and Text Summarisation have also been applied to on-line information systems, an application which might as well be considered educational (Milosavljevic *et al.* 98; Oberlander *et al.* 98; Alfonseca & Rodríguez 03b; Alfonseca & Rodríguez 03a).

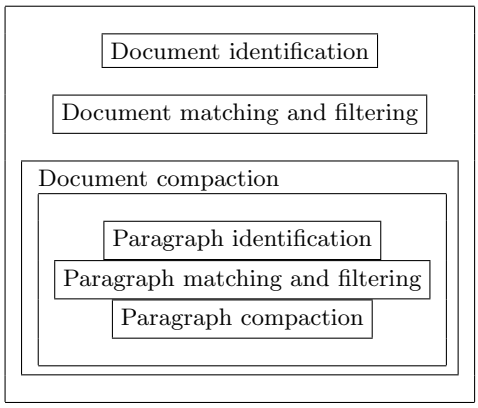

Figure 1: Architecture of the system.

3 Procedure

The architecture designed for this task includes the five common steps for MDS: identification, matching, filtering and compacting the units, and presenting the results. The main difference with respect to other approaches is that most of the systems reported generally assume that the set of documents from which to summarise are all relevant with respect to the topic. That is the case, for instance, in the Document Understanding Conferences. However, in our situation, if we download documents from Internet about a term, we are never going to be sure that the documents really refer about our term, used with the desired meaning. Therefore, the five mentioned steps have been duplicated in a two-tier architecture:

1. Firstly, we apply the identification, matching and filtering steps using as textual unit whole documents. The purpose of these steps is to keep just the documents which are most relevant about the term.
2. Secondly, once we have filtered the set of documents, they have to be compacted. This step consists of applying the five steps again, but using now, as unit, a paragraph. In this second tier, the system identifies paragraph boundaries, decides which paragraphs seem more relevant, and merges the information from them.

Figure 1 displays the architecture of the system. The following sections describe each step in detail.

3.1 1st tier: document-based summarisation

Input data We assume that the glossary is a list of terms. Although this is not a requirement of the system, in the ideal case, the glossary contains, for each term, a very brief definition. This glossary could be either written by hand by the user, obtained automatically with a search of glossaries about the desired domain in the web, or obtained with an automatic Term Identification procedure. The glossary used in our experiments has been obtained with a search about Operating Systems in the web. Figure 2 shows the first six words in the list used, and their brief descriptions.

Once the system receives the glossary, it analyses each definition applying tokenisation, sentence splitting, part-of-speech tagging, stemming and partial parsing with the *wraetlic* tools (Alfonseca 03). Furthermore, the system identifies whether the meaning of an acronym is written in the definition, by checking that there is a sequence of words starting with capital letters that correspond to the letters of the term.

Document retrieval With this information, the system is now able to construct a query for the Google search engine, to retrieve a set of documents for each of the terms. To build the query, we always include, as compulsory keywords, the name of the term to be defined, followed by the verb "is" (to favour those pages in which the term is defined). Additionally, the query is extended with the following optional keywords:

- If no definitions are available, other terms from the list can be added as optional keywords in the query, trying to guide the search engine towards pages whose domain is the same as the glossary.
- If the meaning of the acronyms are available, and the term is an acronym, the first optional keywords to be added are the words in the acronym, followed as well by the verb *to be*. For instance, if the term is *ASCII*, the optional keyword will be *American Standard Code for Information Interchange is*.
- Finally, if brief definitions are available, the remaining optional words are the rest of the words in the definition, excluding closed-class words.

Note that Google just considers the first ten optional keywords, so those that are afterwards will be ignored. In several experiments, we have found that we obtain the best results if we start with the nouns from the term definition, followed by the verbs, and we leave the adjectives and the adverbs until the end.

Once we have constructed the query, we ask Google to return the first 100 URLs that answer it. In order to maximise the variability of the results, we have added the additional restriction that all the pages must be retrieved from different hosts. In this way, we continue asking for more results until we have collected 100 pages from different hosts, or until there are no more results.

The web pages are next transformed into XML format, in which the only format that is preserved is the paragraph boundaries. This format will be suitable to be processed with the aforementioned linguistic processing tools for morphological and syntactic analysis.

Document filtering Next, the system filters the results provided by the search engine. In our experiments, a manual evaluation has disclosed that less than 30% of the retrieved pages really verse about the term that we want to define. Therefore, it will be useful if we can automatically remove, from the set of documents, the lest relevant ones.

The procedure followed is the following: every docu-

ANSI	American National Standardisation Institute
ASCII	American Standard Code for Information Interchange - a table converting numeric values into human readable characters.
API	Application Programming Interface - the set of routines/functions made available to a program developer.
ATA	AT Attachment - also known as IDE.
ATAPI	ATA Packet Interface - minor extension to IDE to control additional device types.
BASIC	Beginners All-purpose Symbolic Instruction Code - a high-level interpreted programming language which is very easy to learn.

Figure 2: Sample words at the beginning of the glossary used in the experiments.

ment is represented in the VSM as the vector of words contained in it with their frequencies. Using the small definition from the glossary, we shall generate another vector that represents the term. If the glossary does not contain definitions, other terms from the same glossary can be used to calculate this vector. Finally, we calculate the cosine distance between the vector of each document and the vector of the definition. Only the documents with the highest scores are kept. Section 4 discusses the recall/precision curves obtained in this step.

It should be born in mind that, at this step, we can take advantage of the fact that the web contains a high level of redundancy, which implies that many of the retrieved web pages contain repeated information. This means that we can afford to lose some relevant data, because we can be quite confident that it will appear in some other document. In other words, it is more important to obtain a good precision at this step than to try to maximise the recall, because we can assume that we can afford to lose some relevant (but redundant) documents.

3.2 2nd tier: paragraph-based summarisation

Paragraph identification and filtering Once the most relevant documents have been filtered, the next steps would be the document compaction. At this point, the second tier of the architecture starts functioning, now processing the paragraphs.

All the documents have been processed, and they are annotated with paragraph boundaries. At the beginning of the second tier, the paragraphs have to be analysed, and those which are not judged relevant will be filtered out. Web pages usually contain navigation instructions, menus, titles and foot-pages which will not be relevant at all. A manual observation of the web pages showed that there are a few lexico-syntactic patterns with which we can identify most of the paragraphs that define the terms, such as when the term appears in the subject position of the verb *to be*. These kind of lexico-syntactic heuristics and patterns have also been used previously for generating definitions and biographies in other MDS systems (Alfonseca & Rodríguez 03a; Lacatusu *et al.* 04; Erkan & Radev 04). This procedure has been complemented with the heuristic that a unit that is in between two relevant units will also contain relevant information. Therefore, the algorithm for filtering the paragraphs is the following:

1. Annotate all the paragraphs in which the term to be defined appears either as the syntactic subject or as the agent, in the case of passive sentences.
2. Filter in all the paragraphs p that
   - Either p has the term as the subject of *to be* in a sentence.
   - Either there is one paragraph p_1 before p and one paragraph p_2 after p for which the term is the subject in a sentence, and there are less than three paragraphs in between p_1 and p_2..

In this way, if there are just one or two paragraphs in between two that have been considered relevant, they will be filtered in as well.

The filtered paragraphs are next cleaned, to remove sentences that are usually incorrect or uninformative: interrogative phrases, unfinished sentences (e.g. those that end with a preposition or a comma), or sentences that start with lowercase.

Summary generation After filtering the paragraphs, it is necessary to compact them, by integrating the information from all the paragraphs in a single summary. In this step, we have assumed that all the paragraphs are internally well structured, and written in a coherent way. Therefore, we would like to retain as much as possible the inner structure of all the paragraphs. This means that if one sentence s_1 is written before other sentence s_2 inside an original paragraph, we would like to preserve that ordering between the two sentences in the final summary, because there is a correct discursive ordering in which s_1 should come before.

To aggregate the paragraphs while satisfying this constraint, we have devised an algorithm in which, initially, we take one paragraph p as an initial model summary, and we proceed by alternating sentences from the other paragraphs, in the same order in which they were written, in between the sentences p, in an incremental way.

The following algorithm merges the sentences from several paragraphs in single summaries:

1. For each paragraph p_k,
   (a) Initialise the target summary Sum_k as an empty text.
   (b) Let $p = p_k$.
   (c) Remove the first sentence s from p, and add

it at the end of Sum_k.

(d) Calculate the similarity between s and the first sentence of all the paragraphs. It is calculated using the VSM, and the similarity metric used is the size of the intersection of the two vectors of words. Note that, now, the first sentence of p is the sentence that was right after s.

(e) Let p be the paragraph whose first sentence maximises the similarity, and go back to step (c) with that paragraph. If the best similarity is 0, stop.

2. At this point we have k different summaries, depending on the paragraph that we took as starting point. It should be possible to allow the user to choose which one to read, but the summary recommended by the system is the one that maximises the number of sentences.

It can be seen that, in the summary, all the sentences that stem from the same paragraph appear in the same order.

Last of all, we perform a few procedures to try to improve the readability of the generated summaries:

- If a sentence starts with a conjunction, and the previous sentence does not come from the same paragraph, remove the conjunction, because it is probably dangling.
- If two sentences have more than 75% of their vocabulary overlapping, remove the shortest one, because they are probably redundant.
- If the term can be written in several ways (e.g. *ASCII* and *American Standard Code for Information Interchange*), substitute all but the first appearances of the longest by the shortest. In this way, the meaning of the acronyms will only appear the first time.
- If the subject of a sentence is the same as the subject of the previous sentence, substitute it for the pronoun *it*.

Figure 3 shows an example of two generated summaries.

4 Evaluation and results

Figure 4 shows the recall/precision curves of the document filtering step. For the first ten terms, all the documents downloaded (678 in total) have been classified manually as relevant and irrelevant. Out of them, 200 (29.5%) were judged relevant. As it is well known, there is always a trade-off between precision and recall. In our case, we have chosen to keep the upper 37.5% of the documents whose similarity with the term definition is above the average similarity.

The quality of the generated summaries has been evaluated by hand, using a set of quality questions developed at NIST by the organisers of the Document Understanding Competition-2005[1]. There are

[1]Obtained from Hoa T. Dang via the DUC e-mail list

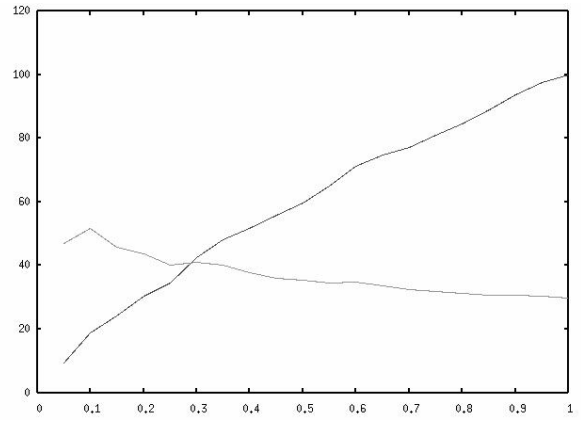

Figure 4: Recall and precision depending on the percentage of documents retained.

five features of the summary that are evaluated independently:

- Grammaticality: the summary should not contain ungrammatical sentences or format instructions which make it difficult to read it.
- Non-redundancy: there should not be unnecessary repetitions, such as repeated sentences or personal names.
- Referential clarity: it should be easy to identify the referent of the pronouns and noun phrases.
- Focus: the summaries should be centred on the term to define.
- Structure and coherence: the summaries should be well structured and organised.

Eight Computer Science students, all of them with a background in Operating Systems, participated in the evaluation. They were divided into two groups, and each member in each group was assigned six random summaries. In this way, each summary was scored by two different judges. Each student had to read each summary and provide a score, in a scale from 1 to 5, to the five features. In total, 24 summaries were evaluated twice.

Table 1 displays the results obtained. As expected, *grammaticality* received the highest score, with a mean value of 4.17. This is because the summaries consist of sentences extracted from web pages, which are expected to be written by hand, grammatically. It came as a surprise that referential clarity also received a high score, 3.96. Given that the summaries are finally produced by merging together several paragraphs, this can be due to the fact that, in most of the cases, the paragraphs are self-contained, and they do not contain references to previous paragraphs that may have been filtered out.

Slightly lower values were obtained in the features of *non-redundancy* and *focus*. Although we dedicated some effort to eliminating very similar sentences and substituting repeated proper names by pronouns, there are, however, long sentences that only have some part in common. We should compare clauses rather

ASCII The American Standard Code for Information Interchange is a set of integers running from 0 to 127 (decimal) that imply character interpretation by the display and other system (s) of computers. It is a standard way of representing characters on many computer systems. ASCII is a standard way of representing text as numbers. It is the most common format for text files in computers and on the Internet. It actually was developed for teletype machines and predates computers. This leads to a minor headache in using it. In an ASCII file, each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined. The range 0..127 can be covered by setting the bits in a 7-bit binary digit, hence the set is sometimes referred to as a 7-bit ASCII. ASCII is the original 7-bit character set, originally designed for American English. The term ASCII file "is often used as a synonym for plain text file" (which is actually based on ISO Latin 1, not ASCII), a file without any special formatting, which can be viewed using UNIX system utilities such as cat (1), more (1), and vi (1). When referring to ASCII, many people erroneously include all the characters in the IBM PC extended character set. This set consists of 254 letters, technical symbols, graphics block characters, and 32 control characters that you do not usually see. The characters with numbers above the original 128 ASCII characters are sometimes referred to as higher-order ASCII characters, but an ASCII file is a file containing only the 128 original ASCII characters. It is currently described by the ECMA-6 standard. ASCII was described by the American National Standards Institute document ANSI X3.4-1986. It was also described by ISO 646: 1991 (with localization for currency symbols). The full ASCII set is given in the table below as the first 128 elements. Languages that can be written adequately with the characters in ASCII include English, Hawaiian, Indonesian, Swahili and some Native American languages.

CPU The CPU is responsible for the actual execution of the instructions contained in the programs. During execution these programs and the data they operate on are usually stored in memory . In the data processing field, CPU is well known to be the center of the computer system in that it governs all information transfers between itself and memory and between memory and the input/output devices. It is the component of the computer system that habitually is used to describe what 'type' of computer you have (i.e. a Pentium 166 MMX System, a Celeron 566 System, a Pentium III 800 System). The CPU is the brain of the computer. It is responsible for executing the machine code instruction set that controls the system. As known, CPU also executes all instructions and determines the orderly sequence of instruction execution. It fetches these instructions from memory and these instructions control every aspect of the computer's operations. Intermediate results are saved and later read from memory. Many people make the mistake of assuming that the CPU is the most important component in the system because of this habit.

Figure 3: Generated summaries for the terms ASCII and CPU.

	Grammaticality	Non redundancy	Referential clarity	Focus	Structure
Mean judges group 1	4.25	3.63	3.83	3.58	3.46
Mean judges group 2	4.08	3.67	4.08	3.71	3.42
Mean	**4.17**	**3.65**	**3.96**	**3.67**	**3.44**
Standard deviation	0.97	1.16	1.07	1.23	1.15
0.95 Confidence interval	[3.64,4.69]	[3.02,4.27]	[3.38,4.53]	[3.04,4.29]	[2.78,4.09]

Table 1: Results obtained in the manual evaluation of the summaries according to the criteria proposed by the NIST organisers of the DUC-2005 competition.

than sentences, for instance, to remove subordinate clauses, to decrease the degree of redundancy. On the other hand, the scores obtained for the feature *focus*, as can be seen, are those with the highest standard deviation. This is because the focus has received both very high scores, and very low scores. We have identified the following difficult cases:

- For some terms, such as *basic*, *handle* and *process*, the filtering steps were not completely successful, and pages which were not relevant entered the last step. Therefore, the summaries generated contain, respectively, sentences about simple things, about door handles, and about legal processes.
- In other cases, the summary did not focus on the term defined, but on differences with other related terms. For instance, the summary about *paging* only discusses the differences between paging and swapping, and the summary about *SCSI* only describes it with respect to *IDE*.
- Finally, for very common words, such as *API* and *CD-ROM*, together with general pages describing those terms, many pages about particular APIs or CD-ROMs were also selected.

Finally, as expected, the feature which received the lowest score was the *structure* of the summaries. The score was higher or equal to 4 in 26 judgings, which indicates that the procedure used to merge the paragraphs succeeds in producing a well-organised paragraph more than half of the times. However, some other times, the summaries have received low scores, specially when very different paragraphs in summaries that were not well focused were merged together.

Educational value The student were also asked to answer whether they would be willing to use this feature in an educational system. All of them agreed that the students should be warned that the information had been collected automatically from an unreliable source and might contain errors. The answers obtained, in a score of 1 to 5, were three 3's, four 4's and one 5, with an average value of 3.75. Therefore, we can say that most of them value the use of this facility in an e-learning system.

5 Conclusions and future work

In this work, we present a new architecture for Multi-Document Summarisation, with an application for automatically generating descriptions of domain-dependent terms obtained from a glossary. Given that the focus of the original documents is not guaranteed when we collect them from the Internet, the summarisation has been divided into two selection steps: initially, the documents are matched with the brief definitions or with the other terms from the glossary to filter out the most dissimilar ones; and, in a second step, the paragraphs from those documents are filtered out using some heuristics. Finally, the information from those paragraphs is put together in the final summary. The evaluation performed with eight students from a Computer Science MSc course shows that the summaries were considered better than average in all their features, with specially good results on grammaticality and referential clarity.

We envision the following lines of future work: (a) to

optimise the document filtering step, for instance, by combining it with an automatic clustering of the documents, so the precision/recall curve improves. This will probably improve the focus of the generated summaries; (b) to combine it with NLG techniques or discourse planners, using information from other paragraphs that we are currently filtering out; (c) to improve the procedure for eliminating the redundancies, so we do not eliminate sentences but portions of sentences which may be repeated; and (d) to integrate the system both in our e-learning environment Tangow (Carro *et al.* 99) and with a Term Identification module, so it provides the definitions of the terms found in course-ware materials and domain-dependent texts.

References

(Alfonseca & Pérez 04) E. Alfonseca and D. Pérez. Automatic assessment of short questions with a BLEU-inspired algorithm and shallow nlp. In *Advances in Natural Language Processing*, volume 3230 of *Lecture Notes in Computer Science*, pages 25–35. Springer Verlag, Berlin-Heidelberg, 2004.

(Alfonseca & Rodríguez 03a) E. Alfonseca and P. Rodríguez. Extending an on-line information site with accurate domain-dependent extracts from the world wide web. In *Semantic Web for Web Learning workshop, CAiSE'2003*, 2003.

(Alfonseca & Rodríguez 03b) E. Alfonseca and P. Rodríguez. Modelling users' interests and needs for an adaptive on-line information system. In *User Modelling 2003*, volume 2702 of *Lecture Notes in Artificial Intelligence*, pages 76–80. Springer, 2003.

(Alfonseca 03) E. Alfonseca. Wraetlic user guide version 1.0. http://www.eps.uam.es/~ealfon/download.html, 2003.

(Ando *et al.* 00) R. K. Ando, B. K. Boguraev, R. J. Byrd, and M. S. Neff. Multi-document summarization by vsualizing topical content. In *Proceedings of the workshop on automatic summarization*, pages 79–88, 2000.

(Angheluta *et al.* 04) R. Angheluta, R. Mitra, X. Jing, and M.-F. Moens. K. u. leuven summarization system at DUC 2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Barzilay *et al.* 99) R. Barzilay, K. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 550–557, 1999.

(Blair-Goldensohn *et al.* 04) S. Blair-Goldensohn, D. Evans, V. V. Hatzivassiloglou, K. McKeown, A. Nenkova, R. Passonneau, B. Schiffman, A. Schlaikjer, A. Siddharthan, and S. Siegelman. Columbia university at duc-2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Brin & Page 98) S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

(Burstein *et al.* 98) J. Burstein, K. Kukich, S. Wolff, C. Lu, M. Chodorow, L. Bradenharder, and M. Dee Harris. Automated scoring using a hybrid feature identification technique. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, Montreal, Quebec, Canada, 1998. The Association of Computational Linguistics.

(Burstein *et al.* 01) J. Burstein, C. Leacock, and R. Swartz. Automated evaluation of essays and short answers. In *Proceedings of the 5th International Computer Asssited Assessment Conference*, Loughborough, U.K., 2001.

(Cabré *et al.* 01) M. T. Cabré, R. Estopá, and J. Vivaldi. Automatic term detection: a review of current systems. In *Recent advances in computational terminology, volume 2 of Natural Language Processing*, pages 53–87. John Benjamins, 2001.

(Carro *et al.* 99) R. M. Carro, E. Pulido, and P. Rodríguez. Dynamic generation of adaptive internet-based courses. *Journal of Network and Computer Applications*, 22:249–257, 1999.

(Conroy *et al.* 04) J. M. Conroy, J. D. Schlesinger, J.Goldstein, and D. P. O'Leary. Left-brain/right-brain multi-document summarization. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Deerwester *et al.* 90) Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

(Erkan & Radev 04) G. Erkan and D. R. Radev. The university of michigan at duc 2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Harabagiu & Maiorano 02) S. Harabagiu and S: Maiorano. Multi-document summarization with GISTEXTER. In *Proceedings of the Third Language Resources and Evaluation Conference (LREC-2002)*, Las Palmas, 2002.

(Lacatusu *et al.* 04) F. Lacatusu, A. Hickl, S. Harabagiu, and L. Nezda. Lite-GISTexter at DUC2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Liu *et al.* 05) C.-L. Liu, C.-H. Wang, and Z.-M. Gao. Using lexical constraints for corpus-based generation of multiple-choice cloze items. In *Proceedings of The Second Workshop on Building Educational Applications Using Natural Language Processing*, 2005.

(Mani & Bloedorn 99) I. Mani and E. Bloedorn. Summarising similarities and differences among related documents. *Information Retrieval*, 1(1):35–67, 1999.

(Mani 01) I. Mani. *Automatic Summarization*. John Benjamins Publishing Company, 2001.

(Milosavljevic *et al.* 98) M. Milosavljevic, R. Dale, S. J. Green, C. Paris, and S. Williams. Virtual museums on the information superhighway: Prospects and potholes. In *Proceedings of CIDOC'98, the Annual Conference of the International Committee for Documentation of the International Council of Museums*, Melbourne, Australia, 1998.

(Mitchell *et al.* 02) T. Mitchell, T. Russell, P. Broomhead, and N. Aldridge. Towards robust computerised marking of free-text responses. In *Proceedings of the 6th International Computer-Assissted Conference*, pages 233–249, Loughborough, U.K., 2002.

(Mitkov & An-Ha 03) R. Mitkov and L. An-Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 2003 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22, Edmonton, Canada, 2003.

(Nobata & Sekine 04) C. Nobata and S. Sekine. CRL/NYU summarization system at DUC-2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Oberlander *et al.* 98) J. Oberlander, M. O'Donell, C. Mellish, and A. Knott. Conversation in the museum: experiments in dynamic hypermedia with the intelligent labeling explorer. *The new review of multimedia and hypermedia*, 4:11–32, 1998.

(Page 66) E.B. Page. The imminence of grading essays by computer. *Phi Delta Kappan*, 1966.

(Page 94) E.B. Page. Computer grading of student prose, using modern concepts and software. *Journal of Experimental Education*, 2(62):127–142, 1994.

(Rudner & Liang 02) L.M. Rudner and T. Liang. Automated essay scoring using bayes' theorem. In *Proceedings of the annual meeting of the National Council on Measurement in Education*, 2002.

(Saggion & Gaizauskas 04) H. Saggion and R. Gaizauskas. Multi-document summarization by cluster/profile relevance and redundancy removal. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Salton 89) G. Salton. *Automatic text processing*. Addison-Wesley, 1989.

(Valenti *et al.* 03) S. Valenti, F. Neri, and A. Cucchiarelli. An overview of current research on automated essay grading. *Journal of Information Technology Education*, 2:319–330, 2003.

(Vanderwende *et al.* 04) L. Vanderwende, M. Banko, and A. Menezes. Event-centric summary generation. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.

(Virvou *et al.* 00) M. Virvou, D. Maras, and V. Tsiriga. Student modelling in an intelligent tutoring system for the passive voice of english language. *Educational Technology and Society*, 3(4), 2000.

(Witte *et al.* 04) R. Witte, A. Bergler, Z. Li, and M. Khalifé. Multi-erss and erss 2004. In *Proceedings of the Document Understanding Proceedings Workshop, DUC-2004*, Boston, MA, 2004.