Building annotated resources for automatic text summarisation

Constantin Orăsan

School of Humanities, Languages and Social Sciences
University of Wolvehampton
Stafford Street, Wolverhampton, WV1 1SB
United Kingdon
C.Orasan@wlv.ac.uk

Abstract

Annotated corpora are necessary for automatic summarisation, but given how difficult is to produce them there are only few available. This paper presents an annotation tool which helps the human annotator to select the important units from a text. In addition to the tool, a new annotation scheme is proposed so that phenomena which such as presence of anaphoric expressions and redundancy can be marked. We argue that by annotating these phenomena the results of evaluation can be made more reliable.

1. Introduction

Like in many other fields of computational linguistics corpora are very important for automatic summarisation. Unannotated they are used to extract linguistic characteristics of the texts to be summarised (e.g. presence of cue words and indicating phrases) (Paice, 1981; Salanger-Meyer, 1990; Orăsan, 2001), or when annotated to train summarisation methods and to evaluate their results (Edmunson, 1969; Kupiec et al., 1995; Teufel and Moens, 1997). Being one of the newest fields in computational linguistics, automatic text summarisation still lacks resources (e.g. corpora or tools) and well established evaluation methodologies. The absence of annotated corpora for summarisation can be explained by the difficulty of the annotation task. Especially when the texts to be annotated are long, it is difficult to annotate without the help of an annotation tool. The tool presented in this paper offers a user-friendly annotation environment which tries to help the human annotator.

Annotation tools are very useful when the annotation scheme used is complicated. For example, the annotation of coreferential links cannot be done with high accuracy tool without a specially designed tool (Day et al., 1998; Orăsan, 2000). Given that most of the annotation schemes used for summarisation mark only the most important units[1] in a text, and eventually their importance, a special tool for the annotation is not fully justified. However, in this paper a more complex annotation scheme is proposed which can be applied easier with the help of a special annotation tool. In addition, the tool could prove particularly useful when long texts are annotated.

The structure of the paper is as follows: in section 2. we explain how corpora are used in summarisation. The existing methods for building corpora for summarisation are presented in section 3. A new annotation scheme for summarisation is proposed in section 4., and the tool which allows to apply this scheme is discussed in section 5.. Finally the tool is discussed and conclusions are drawn in section 6.

2. How annotated corpora are used in summarisation?

In summarisation, annotated corpora are mainly used to train machine learning algorithms (Kupiec et al., 1995; Teufel and Moens, 1997) and to evaluate summarisation methods (Edmunson, 1969). In addition to this, they can be used by linguists to investigate linguistic characteristics of sentences to be extracted for a summary and by students to learn how to produce summaries.

Whenever a annotated corpus is used to train machine learning algorithms a set of features is calculated for each unit in the text using algorithms specific to each summarisation method (e.g. presence of terms, length, position) and the information about the importance of the unit is taken from the corpus. In this case, the summarisation of a text means to classify each sentence as relevant for summary or not.

When an annotated corpus is used for evaluation, the list of units extracted from the text is compared with the list of units marked as important in the corpus. Given this comparison, the evaluation method cannot be directly applied to summarisation methods which produce abstracts and not extracts. The measures used for evaluation are taken from information retrieval:

$$Precision = \frac{True\ positives}{True\ positives\ +\ False\ positives} \quad (1)$$

$$Recall = \frac{True\ positives}{True\ positives\ +\ False\ negatives} \quad (2)$$

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

where *true positives* are units which were marked as important in the corpus and are correctly selected by the summarisation program, *false positives* are units wrongly selected by the summarisation program as being important and *false negatives* are units which were marked as important, but not selected by the program. Depending on which units are selected by the summarisation method, the values for precision, recall and f-measure differ. The comparison between two methods means comparison between these values.

[1] In this paper, the term *unit* is used in a broad sense including clauses, sentences and even whole paragraphs.

3. Building corpora for summarisation

Building corpora for summarisation is not an easy task. In this section we present different ways in which these corpora were built. In addition, we present the results of some experiments which try to find how much the human annotators agree.

3.1. Manually built corpora

One obvious way to produce annotated corpora is by asking human annotators to mark the most important units in a text. These units can be sentences (Edmunson, 1969; Kupiec et al., 1995), clauses (Marcu, 1999) or paragraphs (Salton et al., 1996). As in many other tasks it is important to make clear to the human annotator what to annotate and what not to annotate, by providing a set of guidelines.

Corpora were used in summarisation for the first time by Edmunson (1969) who asked human judges to identify the most important sentences from a heterogenous corpus consisting of 200 documents in the fields of physics, life science, information science and humanities. In order to ensure consistency of the annotation, judges were asked to follow a set of guidelines and to select those sentences which indicate *what* is the subject area of the paper, *why* is the research necessary, *how* is the problem solved and *which* are the findings. As can be noticed, these rules broadly correspond to the sections in a scientific paper. In addition, the annotators had to choose those sentences which minimise the redundancy and maximise the coherence. The presented annotation tool helps the annotator by identifying sentences which are likely to be important to a section. The problems of coherence and redundancy are addressed by the tool through a new annotation scheme which captures these phenomena.

The main problem with an annotation method like the one used in (Edmunson, 1969) is that human perceive the importance of unit in different ways, and as a result their agreement could be quite low (as shown in 3.2.). In order to make the annotation task easier (Kupiec et al., 1995) and (Teufel and Moens, 1997) asked human judges to align sentences from the summary with sentences from the full text. In a corpus of 188 scientific and technical documents, (Kupiec et al., 1995) found that only 79% of the sentences from the summary can be matched with sentences from the full text. Several rules were provided to make the alignment easier. Teufel and Moens (1997) found that only 31.7% of the sentences in the summary could be matched with sentences from the full text in their corpus of 202 articles from computational linguistics. In a small scale experiment, (Marcu, 1999) found that only 15% of the clauses in 10 abstracts taken from Ziff-Davis corpus could be aligned with clauses from the full text. These big differences between the results obtained by different researchers suggest that the success of the alignment depends very much on the type of text used and it also depends on how the matching is defined.

When humans are required to decide which are the most important units in a text, one of the most common problems is that they get tired easily, especially if the text is long and they are not familiar with the topics discussed in the text, and as a result the quality of the annotation decreases. In order to solve this problem, the proposed annotation tool reduces the length of the text necessary to read by employing highly reliable methods from summarisation.

3.2. Experiments for selection of important units by humans

The quality of an annotated resource is usually assessed using cross checking. In this case, the same file is annotated by more than one annotator and their annotations are compared. A high agreement between the annotators indicates not only that the annotation was done accurately, but it also indicates that it makes sense to perform such an annotation task.

Marcu (1997) and Tsou et al. (1997), argue that it is possible to obtain annotation with high degree of agreement between annotators. Marcu (1997) describes an experiment in which 15 independent judges were asked to rate each unit from 5 texts as very important, important and unimportant. Comparison between the annotation showed that the judges were consistent when they were asked to mark the very important and unimportant units, but less consistent with what they consider important. Simple majority voting (i.e. more than 7 judges chose the same category for a unit) was possible to apply in 87% of the cases to decide the importance of a unit. Statistical significance tests showed that the agreement between annotators is significant.

In a similar experiment Tsou et al. (1997) asked 6 groups of evaluators, 3 from North China and 3 from Taipei to mark with red 10% of the most important sentences or clauses, and with blue 15% of the next most important ones from 15 Chinese editorials. If in the previously mentioned research the judges received the texts split into units, in this one there is no indication that the judges received any instructions regarding the way they should identify a unit. The importance of each proposition was computed for each by using a weighted average measure called *perceived importance*. Comparison between propositions' perceived importance showed average overall inter-group consistency between North China and Taiwan, and high intra-group consistency. Therefore, it can be concluded that the background of the annotators plays an important role in selecting the important units.

3.3. Automatic building of corpora for summarisation

Even though it is difficult for humans to align units from a summary with units from the whole document, several methods which produce automatically this alignment were proposed. These methods rely on the fact that in the most cases a summary is produced through *copy-paste* from the whole document. The big advantage of such methods is that they can be used to produce large-scale corpora for summarisation with minimum effort. In this section two methods used to produce such annotated corpora are presented.

In (Marcu, 1999) a greedy method is used to eliminate those clauses from the full document which do not reduce the similarity between the summary and the reduced document. When it is not possible to reduce the text further on the basis of the similarity measure, the rhetorical structure of the reduced document is used to eliminate more unimportant clauses. The method was evaluated on 10 randomly selected articles from Ziff-Davis corpus with an average length of 1066 words and it performance was close to human performance. The method was subsequently used to create a corpus of 6942 texts with the important clauses annotated.

Another method which uses (full document, abstract) pairs to align sentences form the summary with sentences from the whole document was proposed in (Jing and McKeown, 1999). In this case, the abstract is seen as a sequence of words some of which appear in the full document. Therefore the problem of alignment is reformulated as a problem of finding the most likely position of the words from the abstract in the full document using an Hidden Markov Model. The method was evaluated on the same corpus used for evaluation in (Marcu, 1999) and similar results were obtained.

Given the success of these two methods, we decided to include them in the annotation tool. However, as all methods used in computational linguistics, these methods still make mistakes. In the light of this we decided to allow the human annotator to check and validate their results.

4. The annotation scheme

Usually when the most important units in a text are identified they are marked by a simple tag such as <**IMPORTANT**> **...** <**/IMPORTANT**>. In addition to this, an attribute **SCORE** can be provided to indicate how important is the unit. This annotation scheme is very simple, and as a result it can be easily applied to a text using a simple XML editor. However, this annotation scheme ignores certain phenomena which can influence the quality of an abstract, such as the redundancy and dangling anaphors. For this reason we present an extension of the existing annotation scheme which takes into account these phenomena.

Quite often in a document there are several important units which cover similar, if not the same idea, and, therefore, any of them could be included in a summary. The presence of these units can pose serious problems when an automatically produced summary is evaluated. As mentioned before, the automatic evaluation of an extract involves comparison between the list of units extracted by a summarisation program and the list of units marked by humans as important. If not all the important units covering a same idea are marked by the human annotator because he/she tries to reduce redundancy, it is possible that the automatic summarisation program will extract one which is not marked and this will be considered a mistake by the evaluation program. On the other hand, with the current evaluation methodology, it is not correct to mark all the important units which cover an idea because a summarisation program will either have to extract all the units about that idea, which will lead to high redundancy in the abstract, or its result will be considered not very good because only some of the marked units have been extracted.

In the light of these problems, we propose a first extension to the annotation scheme so that important units covering the same idea are linked together. This means that all these units will have to be marked by the human annotator, but the automatic summarisation program will have to extract only one of them. In addition to the solving the mentioned problems, this information allows us to compute the redundancy in an automatic summary. The fact that a unit is similar to another one is marked by the attribute **GROUP** in the <**IMPORTANT**> tag which indicates the id of the group of units to which the unit belongs.

When an extract is produced, it happens that a unit which contains a referential expression is extracted, but not the unit which contains the antecedent of the referential expression. This phenomenon is called *dangling anaphor* and reduces the legibility of extracts. In order to address this problem, we mark those units containing referential expressions and indicate the units which contain the entity referred. In most cases, the same entity appears in several units, and therefore, a decision has to be taken if all the units have to be marked as containing that referred entity. One solution is to mark only the first apparition on the entity in the text, but this is a decision which has to be specified in the annotation guidelines.

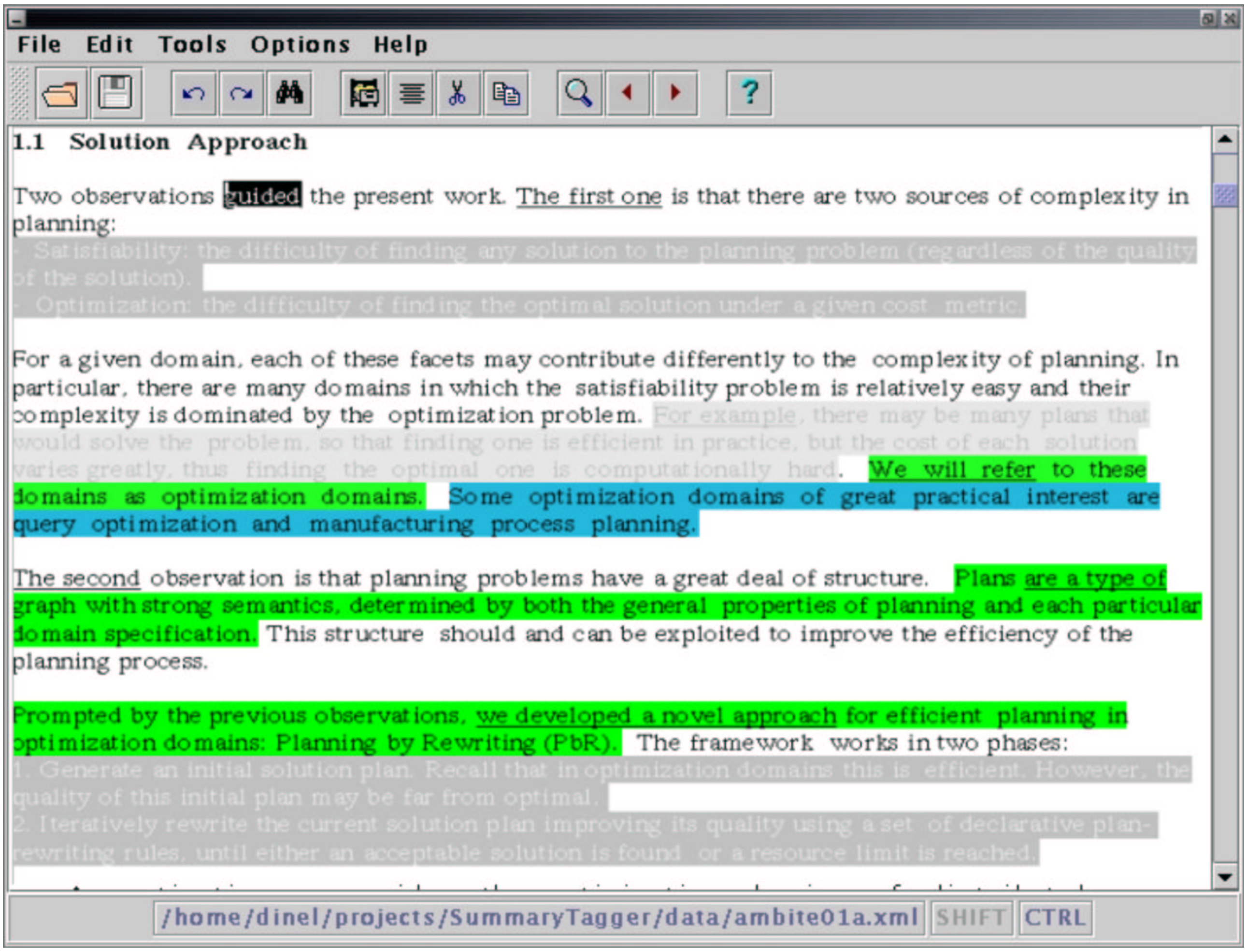
By including this information in a gold standard, we can require that an automatic summary which extracts a unit with a referential expression, extracts also a unit that contains the referred expression. In this way it will be possible to evaluate automatically an extract also in terms of cohesion. One could argue that such an extension is not necessary because if the corpus is coreferentially annotated the same information can be extracted from this annotation. This is true, but given how difficult is to annotate coreferential chains and how few coreferentially annotated corpora are available, it is easier to mark the relations between units and not entities. In addition, it is not necessary to annotate whole coreferential chains, which makes the annotation process easier.

All these extension to the standard annotation scheme are built into the tool presented in section 5.

5. The annotation tool

The results presented in (Marcu, 1997; Tsou et al., 1997) suggest that it is possible to obtain high agreement between annotators, and, as a result, reliable annotated corpus. Unfortunately, this high agreement was obtained on small texts like newspaper articles and not long texts like scientific articles. In light of this, we thought that it would be good if we could filter those sentences which do not to contain important information and highlight sentences which are very likely to be included in a summary. In addition to this, the annotation has to be applied in a very easy way, so that the human annotator can concentrate on the annotation process.

A good graphical interface offers the human annotator trouble-free and efficient interaction with the annotated text. The tool has to be easy to be used; with a minimum time to learn how it works. It also has to hide the unnecessary details. For example, in some cases the annotation is made on a text which already has some markup (e.g. sentence boundaries, formatting, etc.). If this markup does not help and should be hidden from the human



1.1 Solution Approach

Two observations guided the present work. The first one is that there are two sources of complexity in planning:

- Satisfiability: the difficulty of finding any solution to the planning problem (regardless of the quality of the solution).
- Optimization: the difficulty of finding the optimal solution under a given cost metric.

For a given domain, each of these facets may contribute differently to the complexity of planning. In particular, there are many domains in which the satisfiability problem is relatively easy and their complexity is dominated by the optimization problem. For example, there may be many plans that would solve the problem, so that finding one is efficient in practice, but the cost of each solution varies greatly, thus finding the optimal one is computationally hard. We will refer to these domains as optimization domains. Some optimization domains of great practical interest are query optimization and manufacturing process planning.

The second observation is that planning problems have a great deal of structure. Plans are a type of graph with strong semantics, determined by both the general properties of planning and each particular domain specification. This structure should and can be exploited to improve the efficiency of the planning process.

Prompted by the previous observations, we developed a novel approach for efficient planning in optimization domains: Planning by Rewriting (PbR). The framework works in two phases:

1. Generate an initial solution plan. Recall that in optimization domains this is efficient. However, the quality of this initial plan may be far from optimal.
2. Iteratively rewrite the current solution plan improving its quality using a set of declarative plan-rewriting rules, until either an acceptable solution is found or a resource limit is reached.

/home/dinel/projects/SummaryTagger/data/ambite01a.xml

Figure 1: The annotation tool

annotator.[2] In most of the cases the human annotators are field experts with little knowledge about annotation schemes and computers, or none. Therefore, the editor has to be designed in such a way that the humans provide the necessary information in a very simple way, using the mouse and the keyboard, and the task of saving the file in the appropriate format remains to be done by the tool. In some cases the annotation schemes have to follow different recommendations (e.g. TEI). The tool can be designed in such a way that it follows the recommendations and does not allow illegal constructions.

The annotation tool assists the human annotator in three ways: it identifies sentences which can be important and sentences which apparently do not contain any relevant information; it allows the human annotator to mark those sentences which cover the same information and the sentences which have referential expressions outside the sentence.

Figure 1 presents a screenshot of the tool. As can be seen some of the sentences are highlighted, whereas some of them are quite difficult to read. The highlighted sentences are sentences which were identified as being important for the document. The sentences which are hardly readable are sentences which contain special formatting (e.g. lists) or indicating phrases. The indicating phrases are underlined in the text.

5.1. Deciding the importance of a unit

The annotation tool decides if a units could be important or not by using well known methods in automatic summarisation. All these methods rely on different parameters which can be adjusted at the running time by the human annotator. Depending on the type of text, the human annotator can decide to run one or more of these methods on the text, or he/she can decide to manually annotate the important units. At any point the annotator can see the list of units highlighted or hidden by the tool and override its decisions. In this section, the methods included in the annotation tool are presented. At present, all these methods included in the tool process sentences and not clauses. However, if the annotator wants to select a clause he/she can do it manually.

5.1.1. Important terms

One of the most common used methods to find important units in a text is to select those units which have

[2] Actually, our annotation tool requires that the input text has the sentence boundaries marked and contains information about the part of speech for each word.

a high score. Usually the score of a unit is computed as the sum of the score of words from that that unit. These scores can be 1 if the frequency of a term is between certain values (Luhn, 1958), or TF*IDF scores (Zechner, 1996). Even though it was shown that such an extraction method is too weak on its own, it is still widely used in combination with other extraction methods.

We decided to compute the scores of a word from a text using TF*IDF because it can be implemented easily and it is a generally accepted scoring method. At present we intent to include more scoring methods for words.

It happens quite frequently that a word receives a high score even if it is no really important for the document. In order to address this problem we decided to give the human annotator the option to ignore certain words. This can be done using the form presented in Figure 2. In addition to this, the users can decide to highlight sentences which have a score above a threshold and to hide those sentences which have a score lower than a threshold. They also can see all the sentences which were highlighted or hidden and override the tool’s decision.

5.1.2. Cue words

Another method which proved useful in automatic summarisation is the cue word method. First proposed by Edmunson (1969), the method uses presence of certain words in a sentence to select or reject it. This method was extended by Paice (1981) to consider certain phrases called *indicator phrases*. Very often the list of cue words and indicating phrases which mark an important or unimportant sentence depends very much on a genre. Therefore, the users of the annotation tool can decide to run the module which identifies the cue words and indication phrases in a text or not, and they also can customise the list according to the text’s genre.

5.1.3. Surface clues

It was noticed that often the important sentences are situated at the beginning of paragraphs or at their end. The users can choose to mark first and the last sentence of each paragraph as being important. After the module is run, they can check manually each sentence selected by this method and confirm or reject its importance.

Another surface method which proved useful is to reject all sentences with a length under a threshold. The idea used in this case is that if the sentence is too short it does not contains enough information to make it worth including in any summary. The user of the system can decide about the threshold.

In addition to the cue words it is possible to select or reject those sentences containing certain XML formatting. For example it is common that the first occurrence of an important term in text is marked using special formatting like italics. The presence of such a tag can make the tool to highlight the sentence containing the tag. The same method can be used to reject sentences. In a scientific document most of the sentences containing an equation can be excluded for the summary. The users can decide to include or remove all the sentences which contain tags specified by the user.

5.1.4. Similarity with a summary

Two methods which automatically identify the important units from a text using its summary were mentioned in section 3.3. At present, these two methods are being integrated in the system so that the users can use an existing summary to identify the important sentences in a text. Given that these methods depend on several parameters, the users can adjust their value in order to obtain the best performance. As in the case of the other inclusion/rejection methods, the users can override the systems decision. This option can become important for the method proposed in (Marcu, 1999) where without information about the rhetorical discourse structure, as it is in our case, it is possible that more sentences than necessary will be selected.

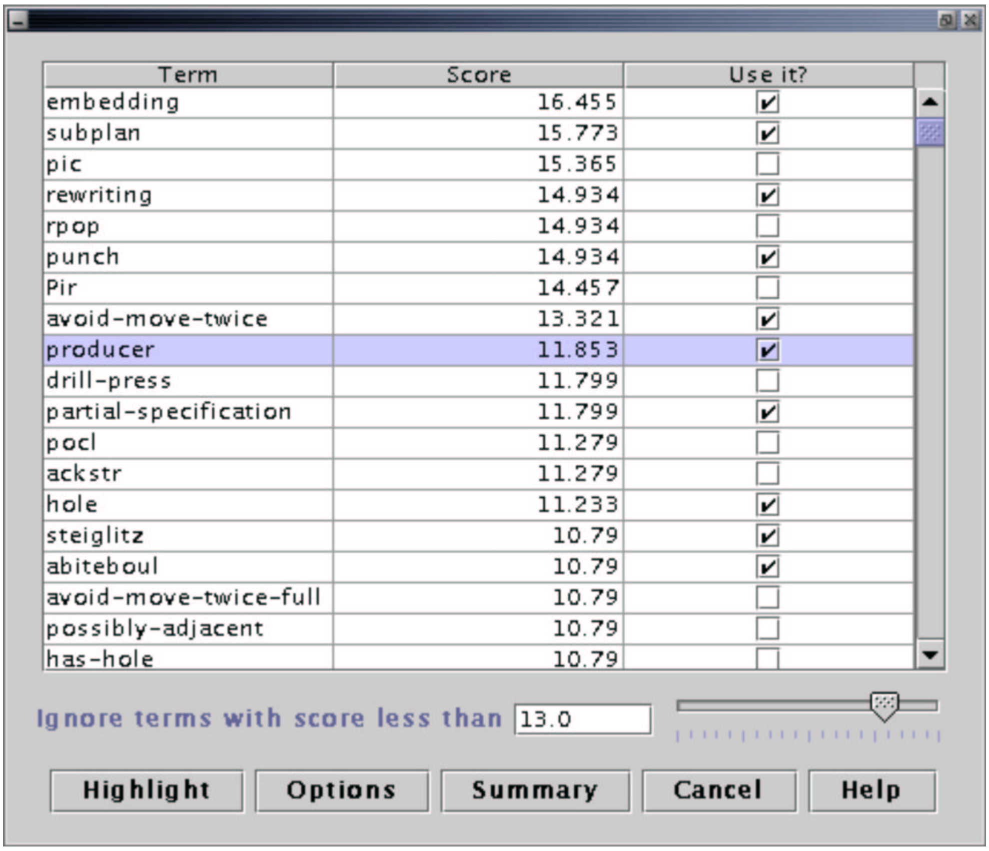
5.2. Marking redundant units

As mentioned in section 4. we consider that it is important to indicate which important sentences cover almost the same information. In order to achieve this, the annotation tool allows the human annotator to group sentences with similar meaning. The grouping is done in a very easy way, by clicking on these sentences. The normal operations to create a group, add elements and remove elements from a group have been implemented. The group is indicated in the resulting file by the **GROUP** attribute in the **IMPORTANT** tag which indicate the ID of the group to with it belongs. At present, we investigate how this operation can be speeded up by including a reliable method which identifies the similar sentences.

5.3. Eliminating dangling anaphors

Dangling anaphors constitute a frequent phenomenon in the extracts making it not very legible. Several summarisation methods tried to address this problem by including the sentence containing the referred expression in the summary (Paice, 1981), but to the best of our knowledge there was no automatic evaluation to see how well the task is performed due to lack of coreferentially annotated corpora. Our tool offers an alternative to this problem. Whenever a sentence containing a dangling anaphor is selected, the human annotators can select one or more sentences which contain the antecedent for the sentence, and which should be included in an extract if the sentence containing the anaphoric expression is extracted. The link between the sentences is marked by the attribute **LINK** which indicates the ids of sentences containing the antecedent for the anaphoric expression.

At present the only automatic procedure which was implemented in the program to speed up the annotation process is the identification of anaphoric pronouns. High precision rules from (Paice and Husk, 1987) have been implemented to identify non referential uses of the pronoun *it*. We also intend to extend the system so that we can identify non anaphoric definite descriptions and to automatically propose sentences which could contain the antecedent.



Term	Score	Use it?
embedding	16.455	☑
subplan	15.773	☑
pic	15.365	□
rewriting	14.934	☑
rpop	14.934	□
punch	14.934	☑
Pir	14.457	□
avoid-move-twice	13.321	☑
producer	11.853	☑
drill-press	11.799	□
partial-specification	11.799	☑
pocl	11.279	□
ackstr	11.279	□
hole	11.233	☑
steiglitz	10.79	☑
abiteboul	10.79	☑
avoid-move-twice-full	10.79	□
possibly-adjacent	10.79	□
has-hole	10.79	□

Ignore terms with score less than 13.0

Figure 2: The form which allows to select the terms

6. Discussion and conclusions

The proposed annotation tool tries to fill a gap in the summarisation field. Its main purpose is to speed up the annotation process through a user-friendly interface and by implementing several automatic ways to identify possible important/unimportant sentences. Given that none of the existing methods in automatic summarisation work with 100% accuracy, the human annotator can override the tool's decisions. In addition to identifying the important/unimportant sentences in a text, we argued that for training and testing automatic extraction methods it is necessary to identify sentences which cover the same information and to indicate which sentences should be included in an extract to avoid dangling references.

One possible problem which can appear during the annotation process is that the human annotator trusts too much the tool and accepts all its decisions. This problem cannot be address through automatic means and the only solution is to check the quality of the annotation using cross validation techniques. The tool makes possible this cross-validation by allowing to compare different annotations of the same text and computing agreement scores.

The tool will not be used only for annotation. Given that all the actions of the human annotator are logged, it will be possible to investigate how humans decide if a sentence is important or not. Another important information which will be extracted from the logs will be the types of modules which are frequently used by the annotators. This will allow us to learn which methods are considered the best for finding important/unimportant sentences. This information will be very useful for our project which plans to develop a tool for computer-aided summarisation because will show us which methods are considered reliable by humans and, therefore, should be included. Another information which will be stored in the logs will be the adjustment of the parameters and thresholds used by different methods. The best values (i.e. the once which make the human override the least of the tool's decisions) will be determined for future use. It is very likely that these values are not common for all type of texts, therefore we hope to be able to use the tool as a way to assess how the text genre determines the kind of methods used and the values for its parameters.

In light of this, the tool will allow us to learn more about the way different automatic summarisation methods work. In addition to proposing sentences which are important/unimportant, each summarisation method can be run on its own as a summarisation method. This means that the tool can be also used to compare summaries produced by different methods. Therefore, it could be used by users, such as undergraduate students, lectures teaching automatic summarisation and researchers in this field, trying to understand how the adjustment of different parameters can influence the quality of the summary produced. At present, we investigate the possibility to allow the user to combine several methods to produce a summary.

7. Acknowledgments

This research is part of the AHRB funded project for developing a computer-aided summarisation tool.

8. References

David Day, John Aberdeen, Sasha Caskey, Lynette Hirschman, Patricia Robinson, and Marc Vilain. 1998. Alembic workbench corpus development tool. In *Proceedings of the First International Conference on Language Resource&Evaluation*, pages 1021 – 1028, May.

H. P. Edmunson. 1969. New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264 – 285, April.

Hongyan Jing and Kathleen R. McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 129 – 136, University of Berkeley, CA, August.

Julian Kupiec, Jan Pederson, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th ACM/SIGIR Annual Conference on Research and Development in Information Retrieval*, pages 68 – 73, Seattle.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159 – 165.

Daniel Marcu. 1997. From discourse structures to text summaries. In I. Mani and M. Maybury, editors, *Proceedings of the ACL/EACL '97 Workshop on Intelligent Scalable Text Summarization*, pages 82 – 88, Madrid, Spain. ACL.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *The 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 137–144, Berkeley, CA, August.

Constantin Orăsan. 2000. CLinkA a coreferential links annotator. In *Proceedings of LREC'2000*, pages 491 – 496, Athens, Greece.

Constantin Orăsan. 2001. Patterns in scientific abstracts. In *Proceedings of Corpus Linguistics 2001 Conference*, pages 433 – 443, Lancaster University, Lancaster, UK, March.

Chris D. Paice and G.D. Husk. 1987. Towards the automatic recognition of anaphoric features in english text: the impersonal pronoun *it*. *Computer Speech and Language*, 2:109 – 132.

Chris D. Paice. 1981. The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases. In R. N. Oddy, C. J. Rijsbergen, and P. W. Williams, editors, *Information Retrieval Research*, pages 172 – 191. London: Butterworths.

F. Salanger-Meyer. 1990. Discoursal flaws in medical english abstracts: A genre analysis per research- and text-type. *Text*, 10(4):365 – 384.

Gerard Salton, James Allan, and Amit Singhal. 1996. Automatic text decomposition and structuring. *Information Procssing & Management*, 32(2):127 – 138.

Simone Teufel and Marc Moens. 1997. Sentence extraction as a classification task. In *Proceedings of the ACL'97/EACL'97 workshop on Intelligent Scallable Text Summarization*, pages 58 – 59, Madrid, Spain, July 11.

Benjamin K. Tsou, Hing-Lung Lin, and Tom B. Y. Lan. 1997. A comparative study of human efforts in textual summarization. In *Proceedings of PACLING '97*, Meisei University, Tokyo, Japan, September 2 - 5.

Klaus Zechner. 1996. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *COLING - 96, The International Conference on Computational Linguistics*, volume 1, pages 986–989, Center for Sprogteknologi, Copenhagen, Denmark, August.