Experimenting with Clause Segmentation for Text Summarisation

Stephen Wan and Cécile Paris
ICT Centre*
CSIRO
Locked Bag 17, North Ryde, Sydney
NSW 1670, Australia
Firstname.Lastname@csiro.au

Abstract

In this paper, we describe our experiments with clause segmentation in producing summaries for the TAC 2008 Update Summarization Track. The submitted runs were designed to determine if a heuristic clause segmentation applied before sentence selection would improve summarization results by reducing the need for sentence compression approaches. A baseline summariser was used to test this hypothesis. The TAC results achieved suggest that a slight trend in improvement was detected.

1 Introduction

In this paper, we describe the automatic text summarisation system implemented by CSIRO for participation TAC 2008 Update Summarization Track. It represents the results of an intensive three week development. This was the first time we participated in either DUC or TAC. We thus had to implement a system from scratch. Our aim was in participating was to investigate specific system-design issues for text summarisation. In particular, given our limited resources, we examined the issue of *Clause Segmentation* and its impact on text summarisation via text extraction methods.

Often, summaries are constrained in length. A sentence is usually extracted in response to some stimulus, such as a generic summarisation request or a representation of a user's interests or a specific user query. It is then usually shortened so that irrelevant information is not presented to the user.

To shorten a sentence, one might apply a series of heuristics commonly used to search for information of a secondary nature to the key proposition expressed in the sentence (for example, an embedded subordinate clause). Having identified this secondary information, one can delete them from the sentence, with the hypothesis that they do not provide enough core information. Such techniques have been used in previous DUC systems (for example, see Dunlavy et al. (2003)). Unfortunately, it may be the case that applying such simple sentence compression methods on extracted sentences might remove the fragments of the sentence that matched well with the stimulus.

For example, often, embedded clauses might indicate information of lesser importance, particularly when it modifies a noun phrase as in the case of the relative clause, "Ban Ki-moon, *who began his term in 2007*, addressed the General Assembly this morning." Unfortunately, if a user query was interested in specifically the beginning time of the term, the compression, "Ban Ki-moon addressed the General Assembly this morning." would not be a good answer to the user's information need.

State-of-the-art approach to sentence compression have been described in Knight and Marcu (2002) and Clarke and Lapata (2007). Applying these techniques, however, might be particularly problematic when sentence compression methods shorten the sentence without considering any additional context such as a user bias.

Clarke and Lapata (2007) models discourse parameters which are provided to the sentence compression algorithm so that content in focus within

*Information and Communication Technologies Centre

- Title: Airbus A380
- Narrative: Describe developments in the production and launch of the Airbus A380.

Figure 1: A Topic Statement

the overall document is not removed. Such a method could be applied to take the query into account in a similar way. However, these approaches do take some amount of effort to deploy and are usually dependent on suitable training data for machine learning methods.

Nevertheless, to meet the space constraint of 100 words imposed by the TAC 2008 competition, some method of shortening sentences is clearly necessary. For our participation this year, state of the art compression methods were not feasible in terms of resources, and heuristic approaches seemed to run the risk of omitting the very information that would provide a good answer.

In response to this, we designed a system that would apply clause segmentation to document sentences *before* they were selected. Once this segmentation has occurred, clauses that match queries would not require further compression since they are already short. An unsupervised heuristic clause segmentation tool was implemented based on the same heuristics for sentence compression, with some modifications to correct fragments such that they resembled grammatical sentences.

In the remainder of this paper, we describe the system developed for the TAC 2008 Update Summarization Track and the evaluation results that addressed this design questions.

2 The Update Summarisation Task

In the task specification, each summarisation training and test case is provided with a topic statement that represents the user's interests. It includes a *title* categorising that interest and a series of "essay question"-like directives outlining the specific aspects of the topic in which the user is interested. These directives are referred to as the *narrative*. An example is provided in Figure 1.

In addition to the topic statement, each training and test case also includes an ordered pair of document sets, referred to as Document Set A and B, respectively. The two sets are constructed such that documents in Set B have a time-stamp that occurs chronologically after documents in the first set. The update summarisation task requires that Document Set A is summarised with respect to the topic statement. This is essentially a standard query-biased multi-document summarisation scenario.

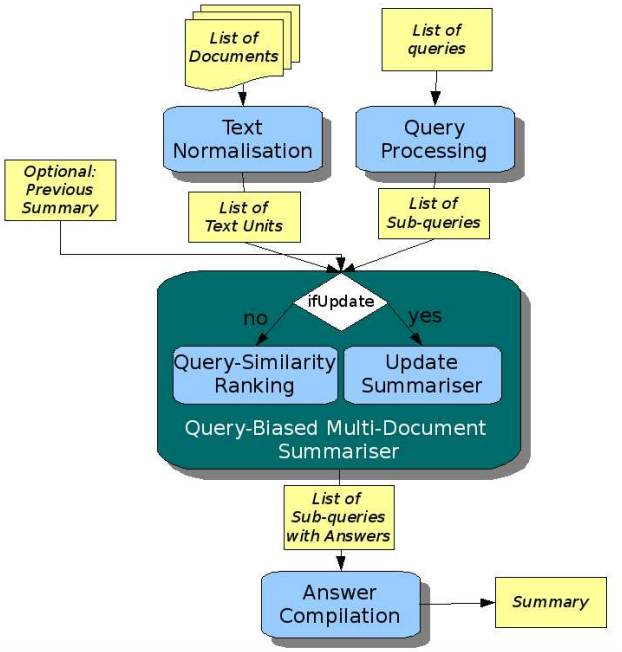

Figure 2: System Architecture Diagram

The update summarisation task also requires that Document Set B be summarised with respect to the topic statement *and* the first set to produce an *Update* summary. This update summary should select information that relates to the topic statement that does not appear in Document Set A.

3 System Architecture

3.1 An Overview

Our summarisation system was designed as a pipeline of four sub-tasks: *Query Processing*, *Text Normalisation*, *Query-biased Multi-document Summarisation* (QBMDS) and a *Answer Compilation*. The QBMDS stage decomposes into two variants, depending on which of the two sets is being summarised. Figure 2 presents a pictorial representation of this pipeline.

The topic statement is first analysed so that the narrative, usually a lengthy sentence involving multiple conjoined main clauses, is decomposed into component sub-queries. In the Text Normalisation stage, the document sets, representing the results of a document search process, are first pre-processed to delimit sentences. If the clause segmentation system parameter is set, then sentences are separated into clauses. We refer here to the smallest unit of text segmentation as a *Text Unit* which can either be, in this case, a sentence or a clause.

Each sub-query is then matched in the QBMDS stage against the text units in the document set. The matching process returns a ranked list of text units, the top-ranking of which will be considered as answers to the sub-query.

Finally, to generate the summary, we iterate through each sub-query and its matched text units, choosing the best text units until the available space is exhausted. We now describe each of these stages in more detail.

3.2 Query Processing

Following Mollá and Wan (2006) the narrative of the topic statement is split into its component questions. In this work, a slightly different set of rules is used to perform sub-query splits. In particular, this system does not rely on creating grammatically correct questions from sub-queries, since these will be treated as bag-of-words queries by the summarisation module.

To separate the narrative into its component sub-queries, we first segment it into sentences, if multiple sentences exist, using the *sptoolkit* from Manchester University.[1] The sub-query segmentation algorithm then applies the following heuristics:

1. Segment narrative text into sub-queries where a WH-word is found.

2. Segment narrative text into sub-queries if a conjunction (in this case, "and" or ";") is found.

The heuristics are applied in the order specified so that heuristic (1) is applied to each sentence to produce a list of sub-queries. Each of these sub-queries is then analysed with heuristic (2) to produce fine-grained sub-queries.

The sample topic statement in Section 1 would be transformed into the following sub-queries:

Sub-Queries:

- describe developments in the production
- launch of the Airbus A380

Finally, when searching for text units that match the sub-query, words from the title of the topic statement are appended to introduce the topic for the search.

Unique Search Terms for Sub-Queries:

- describe developments in the production Airbus A380
- launch of the Airbus A380

3.3 Text Normalisation

In this work, we decided to use our own clause segmentation approach since it needs to be of a length that is suitable as an answer to a sub-query. That is, we did not want the clauses to be too fine-grained as they must still contain sufficient information to be of use to the reader of the summary. As such, we opted to write a heuristic clause segmentation algorithm that would attempt to identify independent clauses (those that can stand by themselves as sentences).

To begin with, the text is segmented into sentences using the *sptoolkit*. Our clause segmentation algorithm uses a heuristic approach that breaks a string into components when certain strings are encountered. A post-process then repairs the segmented strings by concatenating it with its neighbour if a clause boundary is deemed to be unlikely.

Sentences are segmented whenever a conjunction or a set regular expression known to indicate clause splits is observed. In the latter case, this may include patterns containing certain punctuation marks. The string that triggered the clause split is referred to here as the *trigger string*. The string in between the trigger strings is referred to here as a *text segment*. The result of this first stage is a list of text segments and trigger strings.

[1] `http://text0.mib.man.ac.uk:8080/scottpiao/sent_detector`

Each text segment in the resulting list is then examined to see if it is a good independent clause. If the following is true:

- it is part of a list;
- it is a relative clause;
- it is simply too short;
- it begins with punctuation;
- it begins with attributive verbs like “said”;
- or, it begins with the copula (for example “is”, “was”, “were”)

then, it is deemed not to be an independent clause and is concatenated with the previous text segment with the appropriate trigger string re-inserted. The last four conditions in this list are there to handle any text segments that will display poorly when included in a summary.

3.4 Query-biased Multi-document Summarisation

3.4.1 A Vector Space Approach

To match text units to sub-queries, we use a standard information retrieval vector space approach (Salton and McGill, 1983) as applied to the text units segmented in the Text Normalisation stage. Such an approach has been used in a number of DUC entries (for example, see Radev et al. (2003)). Each text unit or sub-query was represented as a vector using unweighted term frequencies. The vector contains one dimension for every word in the vocabulary with the exception of those that occur in a stop-word list. We used cosine as the similarity metric between text units and sub-queries. The result is a ranked list of text units for a sub-query.

3.4.2 Update Summarisation

To perform the update summarisation task, we modified the vector space approach above by changing the way words were ranked. We used the Maximal Marginal Relevance (MMR) method (Carbonell and Goldstein, 1998) to choose text units that were similar to the sub-query but different from sentences chosen when summarising Document Set A.

3.5 Answer Compilation

Once text units have been ranked for each sub-query, all that remains is to choose the top ranking units until the word limit is reached. The generated summary is constructed such that each sub-query and the answers found for it are turned into an attribute and value pair, in lieu of a more advanced text generator.

At first, the sub-query is modified so that an imperative is changed into a declarative fragment. For example, imperative verbs like “describe”, “identify”, “trace” or “include” are removed leaving some noun phrase. In the sub-query from Section 1, “describe developments in the production” becomes “developments in the production”. This is referred to as the *query realisation*.

Until the word limit is reached, each sub-query is considered one by one and the top-ranked text unit for each sub-query is picked as the answer, if it has not already been included in the summary as an answer to some other sub-query. This ensures that a text unit is only seen once in the summary. The query realisation for the sub-query and the chosen text unit are concatenated using only a line break to denote that the former is the attribute and the latter is the value.

This approach runs the risk of omitting an entire sub-query if the word limit is reached prematurely. Examples of the output are presented in Figure 3 and 4 for the test case D0801A-A.

4 Evaluation

This year, the evaluation at TAC’08 was performed manually by NIST and measure both readability and content. The latter was measured using the pyramid method (Nenkova and Passonneau, 2004).

The results for the two variant of our system, Sentence Segmented (SS) and Clause Segmented (CS) are presented in Tables 1 and 2 in Figure 5, which show performance on Document Set A and B respectively. Numbers in the brackets refer to the rank in the evaluation. The metrics reported arace the Pyramid method, the number of Summarization Content Units (SCU), the linguistic quality and the responsiveness. There were 58 competing systems (peers). The run identifier in the evaluation (run id) for each was 8 and 38, respectively.

Both systems rank relatively low comparing to

- *First Document Set:*
  Developments in the production:
  Airbus also stresses the plane's fuel efficiency, claiming that a customer driving a compact car to the airport will burn more fuel per mile than the A380 requires to move one passenger 100 miles. A team of Airbus specialists has visited major airports over the past few years to determine how much work will be necessary to allow the A380 to use runways and terminals. The new Airbus "superjumbo" which will be officially unveiled Tuesday, is the product of a decade of designing.

  Launch of the airbus a380:
  A launch decision is expected in mid-2005.

- *Second Document Set:*
  Developments in the production:
  Airbus said Wednesday the delays were due to production problems linked to the cabin fittings demanded by the different clients. Airbus said Wednesday it was up to six months behind schedule in delivering its new superjumbo A380 aircraft to airlines due to production problems, a delay that could entail financial penalties.

  Launch of the airbus a380:
  Before February to the ICAO, well ahead of the launch of the European-made superjumbo. The European aircraft maker said that A380 deliveries to customers would be pushed back by two to six months.

Figure 3: Sample summaries generated without clause segmentation.

- *First Document Set:*
  Developments in the production:
  Airbus also stresses the plane's fuel efficiency, claiming that a customer driving a compact car to the airport will burn more fuel per mile than the A380 requires to move one passenger 100 miles. A team of Airbus specialists has visited major airports over the past few years to determine how much work will be necessary to allow the A380 to use runways and terminals. Here are some key dates in its development:.

  Launch of the airbus a380:
  A launch decision is expected in mid-2005.

- *Second Document Set:*
  Developments in the production:
  Airbus said Wednesday the delays were due to production problems linked to the cabin fittings demanded by the different clients. Airbus said Wednesday it was up to six months behind schedule in delivering its new superjumbo A380 aircraft to airlines due to production problems, a delay that could entail financial penalties. "We are in the process of reviewing the timetable.".

  Launch of the airbus a380:
  She added that experts would make recommendations before February to the ICAO, well ahead of the launch of the European-made superjumbo. "two to six months depending on the case".

Figure 4: Sample summaries generated with clause segmentation.

Table 1:

Sys.	Pyr.	SCU's	Ling.	Resp.
SS	0.17 (54)	2.50 (55)	1.37 (58)	1.62 (50)
CS	0.18 (53)	2.52 (54)	1.54 (56)	1.64 (54)

Table 2:

Sys.	Pyr.	SCU's	Ling.	Resp.
SS	0.13 (52)	1.62 (53)	1.25 (56)	1.45 (52)
CS	0.10 (50)	1.29 (51)	1.35 (57)	1.37 (52)

Figure 5: Results from the TAC 2008 Evaluation.

the other systems. This is not surprising since a very simple sentence extraction module was used to choose sentences. In particular, without suitable expansion of terms in the query to include related words and synonyms, the sentence extraction stage may, at times, miss good candidate answers. However, for our investigation, it is not the absolute ranking that is important, but rather, the relative ranking between the two systems.

As can be seen by the results in Table 1, clause segmentation seems to improve the ranking of the system for Document Set A. For Document Set B, the reverse is true for the pyramid scores; however, linguistic quality still improves. We attribute the poor responsiveness for Document Set B to a bug in our implementation of MMR for update summarization. We accounted for redundancy between text units chosen in Document Set A with Document Set B. However, we did not account for redundancy within sentences chosen for Document Set B.

We thus can only learn from the results in Table 1 which suggests that clause segmentation before sentence extraction improves summarisation results based on the TAC2008 rankings.

5 Conclusion and Future Work

We conclude that summarisation responsiveness and quality for an extraction based summariser can improve by first performing clause segmentation before summarisation. The relative improvement in rankings are encouraging. This approach is particularly useful if more advanced sentence compression approaches are not possible. In future work, we intend to further develop the clause segmentation approaches to use machine learning methods.

References

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11.

D.M. Dunlavy, D.P. O'Leary, J.M. Conroy, J.D. Schlesinger, S.A. Goodman, and M.E. Okurowski. 2003. Performance of a three-stage system for multi-document summarization. In *Document Understanding Conference 2003: Workshop on Text Summarization*, Edmonton, Canada, May.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Diego Mollá and Stephen Wan. 2006. Macquarie university at duc 2006: Question answering for summarisation. In *Proceedings DUC*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 145–152, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

D.R. Radev, J. Otterbacher, and D. Tam H. Qi. 2003. Mead reducs: Michigan at duc 2003. In *Document Understanding Conference 2003: Workshop on Text Summarization*, Edmonton, Canada, May.

G. Salton and M. J. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.