Generated Abstracts for TAC 2011

Pierre-Etienne Genest, Guy Lapalme
RALI-DIRO
Université de Montréal
P.O. Box 6128, Succ. Centre-Ville
Montréal, Québec
Canada, H3C 3J7
{genestpe,lapalme}@iro.umontreal.ca

Abstract

We have begun work on a framework for abstractive summarization and decided to focus first on a text generation module. In our submitted system, sentences are generated from the syntactic parse of the original sentences. The summaries are written entirely from automatically generated sentences, as well as generated date and location modifiers when appropriate. The evaluation shows above-average performance in the content metric Pyramid, and low scores in linguistic quality.

1 Introduction

TAC's Guided Summarization task was designed in the hope that teams would start working on methods that move away from sentence extraction. The HexTac experiment (Genest et al., 2009) had already shown that there is an empirical limit to the performance of purely extractive methods and we believe that summarization research will be moving towards abstractive approaches. This is why we started working on a fully abstractive framework for summarization (Genest and Lapalme, 2011) that requires both text understanding and text generation. Last year, we presented at TAC a fully abstractive system that focused on the text generation part of our framework (Genest and Lapalme, 2010). Work is in progress to improve on this, but time was running out before this year's evaluation and we submitted the same system as last year's, with a few improvements.

Our proposed framework for fully abstractive summarization is illustrated in figure 1, along with some of the alternatives. Extractive summarization consists of selecting sentences directly from the source documents and generating a summary from them. Sentence compression (Knight and Marcu, 2000) (Cohn and Lapata, 2009) first compresses the sentences and chooses from those and the source documents' sentences to form a summary; it may also be completed in the reverse order, which is to select sentences from the source documents and then compress them for the summary. Sentence fusion (Barzilay and McKeown, 2005) first identifies themes (clusters of similar sentences) from the source documents and selects which themes are important for the summary (a process similar to the sentence selection of centroid-based extractive summarization methods (Radev et al., 2004)) and then generates a representative sentence for each theme by sentence fusion.

On the other hand, our proposed framework for abstractive summarization relies on an abstract representation of the information within the source documents. We propose the concept of *Information Items* (INIT) to help define this abstract representation. An INIT is the smallest element of coherent information in a text or a sentence. It can be something as simple as some entity's property or as complex as a whole description of an event or action. We believe that such a representation could eventually allow for directly answering queries or guided topic aspects, by generating sentences targeted to address specific information needs. As shown in the diagram, the selection of content occurs within the abstract rep-

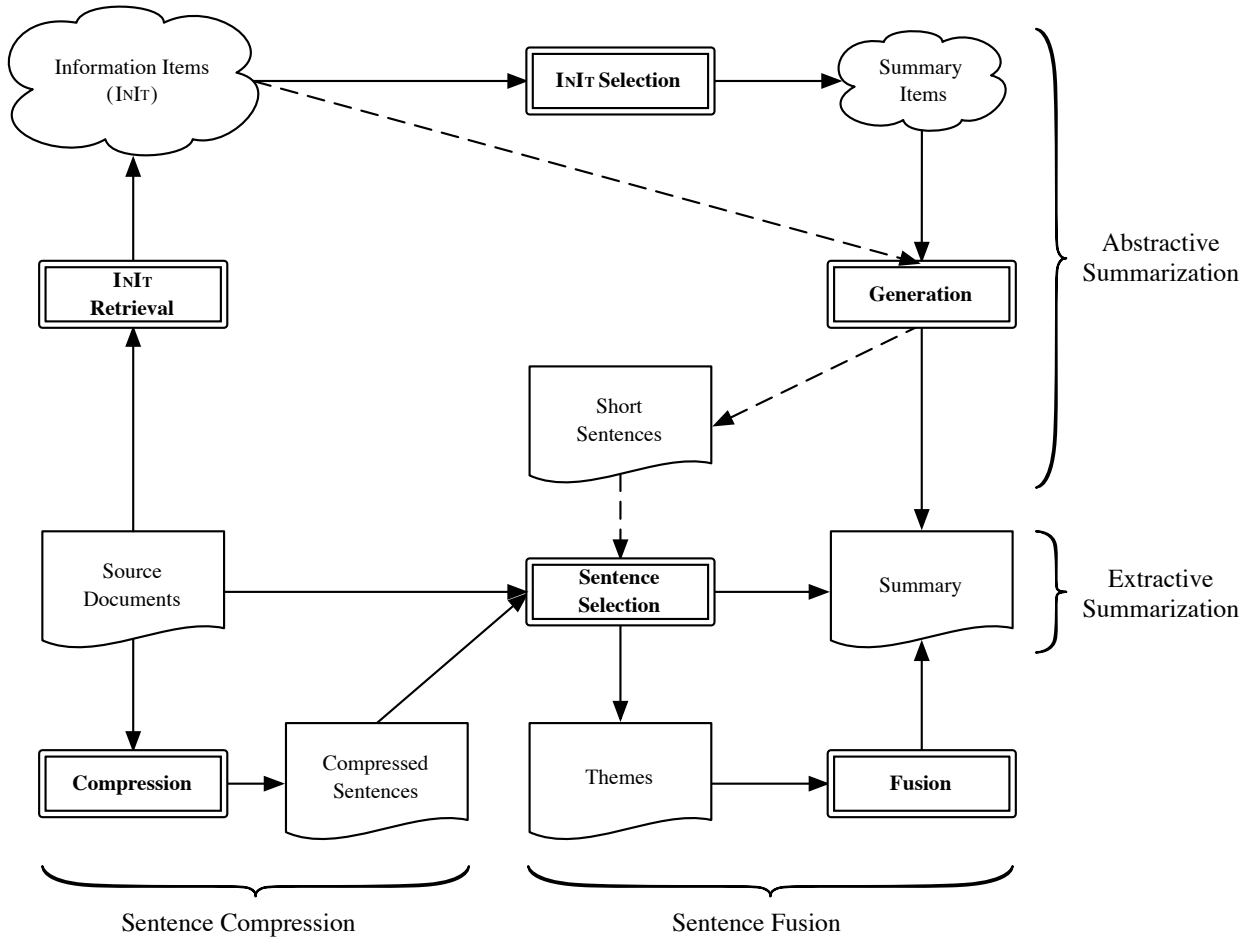

Figure 1: Workflow diagram of our suggested approach for abstractive summarization, compared to pure extractive summarization, sentence compression, and sentence fusion for summarization. The dashed line represents the simplified framework used by our submitted system.

resentation, and it is followed by the generation of sentences and the summary.

Currently, our system focuses on the generation aspect of this framework, and follows the dashed line in figure 1. Sentences are directly generated from the INITs, which are themselves limited to an implementation as subject–verb–object triples, with an associated date and/or location. Examples of the INIT extraction and sentence generation processes are provided in figure 2. Our system generates the summary sentences, rather than extracting sentences directly from the original text. From each INIT and the sentence from which it originates, we have developed rules to generate new sentences, that include only one item of information. We then compute a score based on the frequencies of the terms in the generated sentence, to select the most relevant to the summary. Finally, generating the summary requires some care, because each INIT potentially has a date and a location associated to it. The generated sentences, as they appear in the summary, also include automatically generated date and location modifiers.

We have submitted two runs. The first is exactly the same as last year's, while the second uses the cleaned sentences provided, and attempts to better fill the word limit for each summary. Neither make any attempt to treat update summaries differently. For completeness, this paper includes a full description of our system, but note that the majority of it is identical to last year's.

Original Sentence The Cypriot airliner that crashed in Greece may have suffered a sudden loss of cabin pressure at high altitude, causing temperatures and oxygen levels to plummet and leaving everyone aboard suffocating and freezing to death, experts said Monday.

Information Items

1. airliner – crash – *null* (Greece, August 15, 2005)
2. airliner – suffer – loss (Greece, August 15, 2005)
3. loss – cause – *null* (Greece, August 15, 2005)
4. loss – leave – *null* (Greece, August 15, 2005)

Generated Sentences

1. A Cypriot airliner crashed.
2. A Cypriot airliner may have suffered a sudden loss of cabin pressure at high altitude.
3. A sudden loss of cabin pressure at high altitude caused temperatures and oxygen levels to plummet.
4. A sudden loss of cabin pressure at high altitude left everyone aboard suffocating and freezing to death.

Generated Sentence in the Summary On August 15, 2005, a Cypriot airliner may have suffered a sudden loss of cabin pressure at high altitude in Greece.

Original Sentence At least 25 bears died in the greater Yellowstone area last year, including eight breeding-age females killed by people.

Information Items

1. bear – die – *null* (greater Yellowstone area, last year)
2. person – kill – female (greater Yellowstone area, last year)

Generated Sentences

1. 25 bears died.
2. Some people killed eight breeding-age females.

Generated Sentence in the Summary Some people killed eight breeding-age females.

Figure 2: Two examples of INIT extraction and text generation from a single sentence, and the generated sentence selected for the summary. Examples taken from clusters D1012C-B and D1025E-A.

2 Summarization System

Our approach consists of 6 steps described in the following sections.

2.1 Preprocessing

The preprocessing stage formats the input for easier use. First, it patches the SGML-like files into well-formed XML. From the XML, we extract the text for each document. A few minor modifications are made in the text to make the parsing easier, like removing quotation mark characters, parentheses and their content, replacing some contractions with their full form, and pre-segmenting the text into sentences. At the end of preprocessing, the result is a text file with one sentence per line. A similar but

simpler process was used in our second run, starting from the already cleaned documents provided this year.

2.2 Annotation and Parsing

We run an information extraction engine on the pre-processed document cluster. This produces annotations on the cluster of the words' part-of-speech tags, and words or groups of words that are locations and dates.

We also parse each sentence of the cluster of documents, resulting in a full syntactical dependence tree.

2.3 Information Item Retrieval

We then extract the information items from each sentence, an information item being defined as a subject–verb–object triple. Sample INITs are given in figure 2.

Every verb encountered forms the basis of a candidate INIT. We also identify the verb's subject and object, if they exist, from the parse tree. Many candidates are rejected for various reasons: the difficulty of generating a grammatical and meaningful sentence from them, the observed unreliability of parses that include them, or because it would lead to incorrect INITs. We implemented the rejections using the following rules:

- Verb is a present participle
- Verb is in infinitive form
- Verb has a conjunction with another verb
- Verb is 'to be'
- Verb is not identified as a verb by the ANNIE POS Tagger
- Subject or object is identified as a verb by the ANNIE POS Tagger
- Subject or object is a relative pronoun
- Triple is part of a conditional clause
- Triple has no subject and no object

Experimentally, those criteria were selected because the INITs containing them often lead to generated sentences with incorrect grammar or content. Roughly half the candidates are rejected.

2.4 Sentence Generation

From each INIT retrieved, we generate a new sentence. Our main tools to do this are the original parse tree of the sentence from which the INIT is taken, and an NLG realiser to generate the new sentence. Sample generated sentences are illustrated in figure 2.

Our process can be described as translating the parts that we want to keep from the dependency tree provided by our parser, into a format that the realiser understands. This way we keep track of what words play what role in the generated sentence and we select directly which parts of a sentence appear in a generated sentence for the summary.

Sentence generation follows the following steps:

- Generate a Noun Phrase (NP) to represent the subject if present
- Generate a NP to represent the object if present
- Generate a NP to represent the indirect object if present
- Generate a complement for the verb if one is present and there was no object
- Generate the Verb Phrase (VP) and link all the components together, ignoring anything else found in the original sentence

NP Generation

Noun phrase generation is based on the subtree of its head word in the dependency parse tree, and it is always done in the same way, whether the noun is a subject, object, indirect object, or noun complement (calls to build NPs are recursive). The head in the subtree becomes the head of the NP. Additional parts of the NP are added according to the children of the head in its parse subtree. The following children of the head in the parse tree are 'translated' for the NLG realiser:

- A determiner is kept as is, or changed from "the" to "a" if the NP's modifiers have been removed
- A preposition leads to building a Prepositional Phrase (PP); see below
- A number modifier becomes a 'pre-modifier'
- A noun modifier leads to building an NP and placing it as pre- or post-modifier according to its original position in the original sentence
- A noun that is in a relation of conjunction with the head noun leads to building an NP for that noun and a conjunction between them

- A genitive noun leads to building a genitive NP and setting it to pre-modifier
- An adjective modifier is set as pre- or post-modifier according to its position in the original sentence
- All other children in the subtree are ignored and effectively removed during generation

PP Generation

Prepositional phrases are generated when they are the complement of a noun phrase or when they replace the object as complement of a verb. The head of the PP is the preposition. If the preposition has a noun complement, then we generate an NP for it, otherwise we do not generate the PP.

Verb Complement Generation

When an INIT has no object, then we attempt to find another complement instead, should the verb have no interesting meaning without a complement. The first modifier of the verb that follows the verb in the sentence will be used, except if the complement is the subject of the verb or if it is a punctuation. Prepositional phrase modifiers are generated as described above. All other modifiers are included as verb complements in full, with all of their subtree from the parse. This step includes complements in the form of a verb in the infinitive form, such as in the sentence "George decided to leave", which would be generated in its entirety.

VP Generation

Finally, the verb phrases are also generated from the verb and some of its children. Guests, auxiliaries, negation and perfect form are all kept. Then the NPs generated for the subject, object and indirect object are added with their appropriate function. The verb complement is added if it was generated. If there is an object but no subject, the VP is set to passive. The tense (past or present) of the VP is set to the tense of the verb in the original sentence.

2.5 Dates and Locations

Guided Summarization categories 1 and 2 both include aspects related to time and space, whereas the other categories do not. For these two categories, we use our date and location annotations.

We do not include any words identified as a date or a location in the sentence generation process. In particular, words considered a date or location are ignored while building a NP, PP or complement. Instead, when exactly one date and/or exactly one location appear in the subtree of the verb of an INIT triple, that INIT is assigned a date and/or location, as illustrated in the examples of figure 2. Only dates that can be resolved are used, other dates are ignored. These dates and locations are generated at the time of summary generation instead of sentence generation, as will be described in section 2.7.

Dates are resolved when it is easy to do so, and ignored otherwise. If the date is parsable (such as "January 23" or "3 March, 2006"), then that date is used directly. The words "yesterday" and "today" are interpreted according to the date of publication. The days of the week ("Monday", "Tuesday", etc.) are interpreted as referring to the latest such day before the date of publication. This is not always right but works properly most of the time.

Date and location are used in the summary of the first example of Figure 2. They are not shown in the second example because it is in category 4, different from 1 or 2.

2.6 Sentence Ranking

We use the document frequency (DF) – the number of documents that include an entity in its original text – of the lemmas included in the generated sentence as the main scoring criterion. The generated sentences are ranked based on their average DF (the sum of the DF of all the unique lemmas in the sentence, divided by the total number of words in the sentence). Lemmas in our stop-list and lemmas that are included in a sentence already selected in the summary have their DF reduced to 0, to avoid favoring frequent empty words, and to avoid (perhaps too severely) redundancy in the summary.

2.7 Summary Generation

Abstractive summarization requires text and sentence planning. Text generation patterns can be used, based on some knowledge about the topic, and in the case of Guided Summarization, based on answering specific aspects of the category.

For now, we restrict text planning to a temporal ordering of the sentences, and adding dates and locations to the generated sentences when appropriate, that is for categories 1 and 2. We select sentences

from the ranking performed before, until we go over the word limit of 100 words. Those sentences are ordered by the date of their INIT when it can be determined. Otherwise, the day before the date of publication of the article that included an INIT is used instead. We plan to improve our temporal analysis in the future. All generated sentences with the same date are grouped in a single coordinated sentence. The date is included directly as a pre-modifier "On *date*," in the first INIT of the coordination, and the other INIT's of that date are added to form a coordinate sentence.

Each INIT with a known location has its generated sentence appended with a post-modifier "in *location*", except if that location has already been mentioned in a previous generated sentence of the summary.

At the end of this process, the size of the summary is always above the word limit of 100. We remove the least relevant INIT (see section 2.6) and restart the summary generation process. We keep taking away the least relevant INIT in a greedy way, until the length of the summary is under the 100-word limit. This naive solution to never exceed the limit was selected because INIT's were initially thought to always lead to short generated sentences. However, it turns out that some of the generated summaries are too short because some INITs that were removed can be quite long. Our second run adds an additional step here to find recursively sentences that can still fit within the word limit, from amongst the 20 best-scored sentences.

2.8 Resources Used

This year's system is programmed almost entirely in Java, and makes use of the Minipar parser, the GATE framework, and the SimpleNLG natural language realiser. The preprocessing step also makes use of bash, sed and XSLT.

2.8.1 Minipar

Syntactical parsing is an essential part of our approach. For now, we use the MINIPAR parser (Lin, 1998) from the command-line to build a syntax dependency tree for each sentence of the source documents. MINIPAR provides syntactical dependency relations between each word and its parent in the parse tree, which may be a virtual node that in turn may have a syntactical role and/or an antecedent in the parse tree. A lemma and a part-of-speech is also provided for each word. MINIPAR always produces a complete parse tree, but it is sometimes incorrect. Incorrect parse trees may lead to false or misleading INITs and ungrammatical generated sentences.

2.8.2 GATE and GeoNames lists

We use the GATE framework (Cunningham et al., 2002) to generate annotations on the documents of each cluster. We specifically use certain IE features of GATE, the so-called ANNIE plug-ins, especially the Tokenizer, POS_Tagger, and Gazetteer. We use grammatical rules and data already included in the GATE standard build, with the notable addition of two geographical location lists found on GeoNames (GeoNames, 2010) – those named 'US' and 'cities1000', that include a comprehensive list of US locations and city names.

2.8.3 SimpleNLG

SimpleNLG (Gatt and Reiter, 2009) is a natural language generation framework implemented as a Java library. It realises sentences based on the words we choose and the syntactical relations we identify between them. The syntactical roles known to SimpleNLG are not the same as those of Minipar, and some roles have different names in one and the other. For example, Minipar has *determiners* ('the' or 'a' for example) whereas SimpleNLG calls them *specifiers* to noun phrases.

3 Results and Discussion

Tables 1 and 2 provide the results of our two runs, with Rali1 being the identical system as last year, and Rali2 having an improved sentence selection.

As was the case last year, the linguistic quality of our summaries is very low. This is to be expected from an early attempt at abstractive summarization, whereas we believe most of the other systems used sentence extraction. These low linguistic quality scores are also accompanied by relatively low overall scores, because of the strong correlation between the two metrics. On the other hand, the Pyramid scores are above average for Rali2. We believe that this comes from taking advantage of our sentence generation process, which produces short

factual sentences, and the severe avoidance of redundancy in our sentence selection. Pyramid is also the only metric for which we observe a significant difference in score between our two runs, because Rali2, unlike Rali1, attempts to fill the word limit, as explained in section 2.7.

Like last year, and as can be seen in figure 3, our two runs stand out from the other automatic systems by having a high Pyramid score relative to their linguistic quality score. We observe that Rali2 has the largest ratio of Pyramid to linguistic quality scores in the TAC 2011 evaluation.

Part A	Pyr.	Ling. Q.	Overall R.
Rali1	0.356	2.136	2.386
Rali2	0.393	2.114	2.455
Avg	0.377	2.756	2.686
Best	0.477	3.750	3.159
Models	0.785	4.908	4.761
Part B			
Rali1	0.227	1.864	1.909
Rali2	0.246	1.841	1.932
Avg	0.277	2.752	2.259
Best	0.353	3.341	2.591
Models	0.673	4.821	4.712

Table 1: Scores of Pyramid, linguistic quality and overall responsiveness for our two runs, the average of automatic systems, the best score of any automatic system, and the average of the human models.

Part A	Pyr.	Ling. Q.	Overall R.
Rali1	33	42	39
Rali2	29	43	37
Part B			
Rali1	37	45	41
Rali2	33	47	40

Table 2: Ranks for each manual evaluation metric of our two runs in the TAC 2011 competition, out of the 48 automatic runs.

4 Conclusion and Future Work

We intend to improve on the system submitted this year in many ways, following more closely the framework we have discussed in the introduction. In particular, we are working on a semantic analysis of the text, that would lead to more interesting and useful INITs. Content selection would then happen amongst INITs rather then amongst sentences. Sentence generation will also be improved, relying less on the original structure and more on a deeper analysis, although the work done so far will be helpful in developping more advanced techniques.

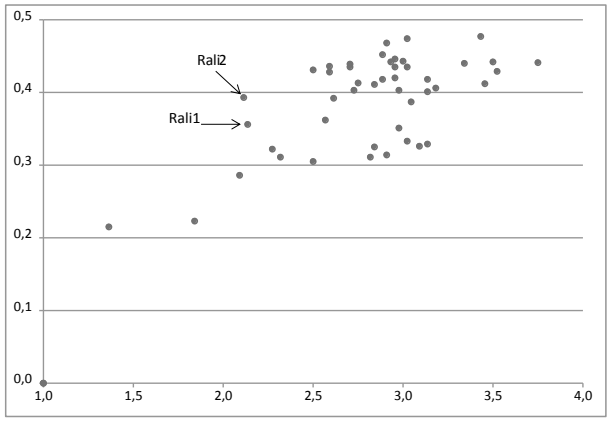

Figure 3: Scatter plot of Pyramid score with respect to linguistic quality in part A, for all the automatic systems competing in TAC 2011.

References

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *J. Artif. Int. Res.*, 34(1):637–674.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA*.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: a realisation engine for practical applications. In *ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93, Morristown, NJ, USA. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2010. Text generation for abstractive summarization. In *Proceedings of the Third Text Analysis Conference*, Gaithers-

burg, Maryland, USA. National Institute of Standards and Technology.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73, Portland, Oregon, June. Association for Computational Linguistics.

Pierre-Etienne Genest, Guy Lapalme, and Mehdi Yousfi-Monod. 2009. Hextac: the creation of a manual extractive run. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology.

GeoNames. 2010. http://geonames.org/.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proc. Workshop on the Evaluation of Parsing Systems*, Granada.

Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.