

# ABSUM: a Knowledge-Based Abstractive Summarizer

Pierre-Etienne Genest, Guy Lapalme

RALI-DIRO

Université de Montréal

P.O. Box 6128, Succ. Centre-Ville

Montréal, Québec

Canada, H3C 3J7

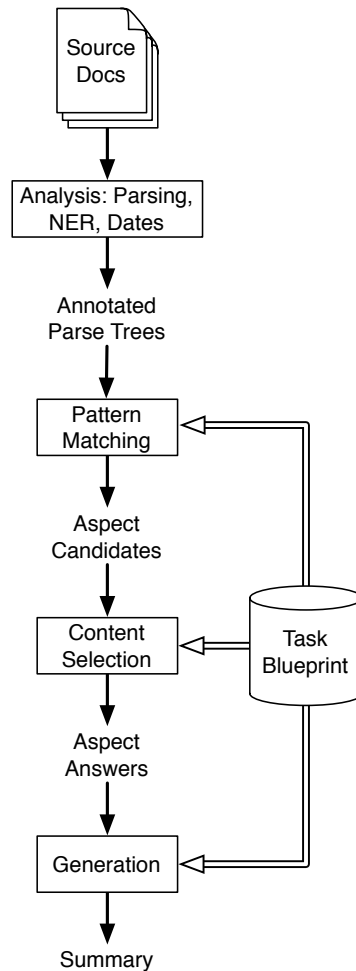
{genestpe, lapalme}@iro.umontreal.ca

*This paper introduces a flexible and scalable methodology for abstractive summarization called K-BABS. Following the analysis of the source documents a knowledge base called a task blueprint is used to identify patterns in the representation of the source documents and generate summary text from them. This knowledge-based approach allows for implicit understanding and transformation of the source documents' content, given that the task blueprint is carefully crafted for the summarization task and domain of interest. ABSUM is a system that implements this methodology for the guided summarization task of the Text Analysis Conferences. Knowledge for two broad news categories has been manually encoded. Evaluation shows that the abstractive summaries of ABSUM have better linguistic quality and almost twice the content density of state-of-the-art extractive summaries. When used in combination with an extractive summarizer, evaluation shows that ABSUM improves the summarizer's coverage of the source documents by a statistically significant amount, and exceeds the content score of the state of the art in text summarization. A discussion of extensions to this work including ways to automate the knowledge acquisition procedure is included.*

## 1. Introduction

Abstractive summarization is one of the main goals of text summarization research, but also one of its greatest challenges. The authors of a recent literature review (Loret and Palomar 2012) even conclude that “abstractive paradigms [...] will become one of the main challenges to solve” in text summarization. In building an abstractive summarization system, however, it is often hard to imagine where to begin and how to proceed in order to incorporate some kind of semantic understanding of the source documents to create a shorter text that contains only the relevant elements for the task at hand.

This paper introduces the **Knowledge-Based Abstractive Summarization (K-BABS)** approach, to address various summarization tasks and domains in a flexible and scalable way. Its architecture relies on an analysis of the source documents and on a task blueprint. This resource describes how to transform the representation of the text into natural language for the summary. It implicitly encodes knowledge about the summarization task into rules applied by the summarization system. The task blueprint, which can be constructed automatically, semi-automatically, or manually, guides every step of the summarization process.



**Figure 1**  
Workflow diagram for the architecture of ABSUM.

The **Abstractive Summarizer of Université de Montréal (ABSUM)** is a system which implements K-BABS for the guided summarization task of the Text Analysis Conference<sup>1</sup> (TAC). Figure 1 shows the architecture of ABSUM. First, an analysis that includes the syntactical parsing and semantic tagging of the source documents is performed. Then, abstraction schemes defined in the (manually constructed) task blueprint are applied to the annotated parse trees to detect candidates for each aspect that needs to be covered in the summary. The answer for each aspect is selected by parameterizable heuristics in the content selection step, which takes into account that the analysis module is not error-free. Finally, the summary is generated based on a generation plan provided by the task blueprint, using templates to generate sentences for each aspect.

ABSUM shows excellent performance on a test set made up of two categories from TAC 2011's guided summarization track. The summaries generated are very short, content-rich, and

<sup>1</sup> [www.nist.gov/tac](http://www.nist.gov/tac)

of good linguistic quality. Their content density is almost two times superior to that of any other automatic summarizers, though they often lack coverage of the source documents. Thus, they gain to be complemented with sentences extracted automatically. When used in combination with a state of the art extractive summarizer, the evaluation shows that ABSUM increases the content of the summaries by a statistically significant amount, without negatively affecting linguistic quality.

This paper has the following organization. In order to put K-BABS in perspective, Section 2 discusses different approaches to summarization that have been proposed over time. Section 3 outlines the general approach K-BABS. Section 4 describes the abstractive summarizer ABSUM. The methodology and results of the evaluation performed are presented and analyzed in section 5. Section 6 is a discussion of K-BABS given the experimental results of ABSUM. Finally, Section 7 concludes.

## 2. Automatic Summarization Strategies

Summarization systems are usually considered to be either *extractive* (which typically refers to sentence extraction, with or without some post-processing edits allowed), or *abstractive* (which typically refers to everything else). (Spärck Jones 2007) suggests the term *non-extractive* instead of abstractive, to include strategies that do not produce abstracts, like synopses and reviews. This terminology is adopted in this paper, with the addition of the category *semi-extractive*, to refer to approaches that compress or merge sentences at a syntactical level, but do not produce new syntactical structures or lexical units.

### 2.1 The ITG Model

(Spärck Jones 1999) considers that the basic model of an automatic text summarizer has the following three stages, forming the acronym ITG:

1. *Interpretation* of the source text into a source text representation
2. *Transformation* of the source text representation into a summary representation
3. *Generation* of the summary text from the summary representation

Her model provides a “common means” for comparing systems based on “the real logic underlying [them]”. The kind of transformation that is expected is a kind of compressive operation on the source text representation that can involve reasoning, inference, conceptualization, simplification, etc.

### 2.2 Extractive Summarization

In the last two decades, automatic text summarization has been dominated by extractive approaches that rely on shallow statistics. Graphs, centroids, latent semantics, machine learning, linear programming and other sophisticated statistics-based algorithms have become the state of the art. They tend to favor a bag-of-words or ngram representation of the text and make use of rudimentary knowledge resources, either general or domain-related, usually consisting of lists of words or cue phrases. Citing all relevant extractive approaches is outside the scope of this paper, and recent surveys (Nenkova and McKeown 2011) (Lloret and Palomar 2012) should be consulted for a more detailed discussion.

With regards to the ITG model, the interpretation of the source text in extractive summarization does not lead to a representation other than textual. The transformation step as defined above

is non-existent, while the generation step is reduced to selecting and ordering full sentences extracted from the source text. In this regard, the best-performing systems, which have been purely extractive prior to this work, have very limited text understanding capabilities.

It remains important to recognize the successes of extractive summarization. Though there is still a long way to go in text summarization, extractive systems already do a reasonable job at summarizing: in the latest evaluation campaign, TAC 2011, the average responsiveness of the top systems was considered “barely acceptable” by human assessment (Owczarzak and Dang 2011), which is nothing to scoff at. This level of performance is possible thanks to years of developing new extractive models and fine-tuning systems.

Nevertheless, there are good reasons to believe that extractive summarization is approaching a ceiling in performance. As has been stated by (Spärck Jones 2007), “the work on extractive summarizing has picked the low-hanging fruit, and the overall trend has been more technological than fundamental.” Systems that claim to be different from one another have all become statistically indistinguishable in evaluation results (such as the top 10 submissions at the TAC 2011 evaluation campaign). The results of the HexTac (Genest et al. 2009) experiment, which evaluated the performance of humans doing pure sentence extraction, show that this ceiling in performance is rather unsatisfactory. Human sentence extraction turns out to perform poorly when compared to that of regular (abstractive) human summaries, and not that much better than the best automatic systems. Even though these results show that there is still some room to improve on the current extractive methods, there are good reasons to conclude that most of what can be done with extraction has been done and that little more significant improvement is to be expected, at least for the summarization of news articles; and a similar upper limit will likely be reached in time for other domains if they become the new focus of evaluation campaigns.

Overall, the best performance to be expected from approaches based only on sentence extraction is unsatisfactory.

### 2.3 Semi-Extractive Summarization

Semi-extractive summarization techniques are similar to extractive summarization in that they aim to construct the summary directly from extracted text, but do not limit the selection to whole sentences, and allow for word or phrase deletion, sentence compression and merging of phrases from different sentences.

The ADAM system by (Pollock and Zamora 1975) was perhaps the earliest work on semi-extractive summarization. ADAM edits the sentences after they have been extracted from the source text. This idea is used by many extractive systems today, mostly in the form of simple deletions, like removing unwanted words and phrases, and substitutions, such as resolving anaphoric pronouns by the referred entity. Approaches that incorporate sentence edition within a limited scope are typically considered to remain within the extractive summarization paradigm.

When more substantial sentence edition is performed, the term sentence compression is used instead. Sentence compression is a field that is studied in its own right, though summarization is usually considered among the NLP tasks for which it can be useful (Cohn and Lapata 2009). Sentence compression has also been studied specifically in the context of summarization (Knight and Marcu 2000) (Zajic et al. 2008). Sentence selection can be performed in the same way as in any of the extractive summarization systems, either before or after the compression. Sentence compression can also be considered as part of the summarization task itself, exploiting discourse information to drive the decisions of the compression process (Daumé and Marcu 2002). Sentence splitting (Genest and Lapalme 2011a) can be considered as a special kind of sentence compression in which each sentence is split into several very short sentences that each contain only one information item.

Sentence fusion is another category of semi-extractive summarization approaches that has been explored in recent years. In (Barzilay and McKeown 2005), themes (clusters of similar sentences) are first identified from the source documents and a selection is made of which themes are important for the summary. This process is similar to the sentence selection of centroid-based extractive summarization methods for multi-document summarization (Radev et al. 2004). A representative sentence for each theme is then generated by sentence fusion, using heuristics that add and remove parse tree fragments from a base sentence and then generating a sentence from the enhanced parse tree. (Krahmer et al. 2008) suggest that query-driven sentence fusion may be a more tractable task that improves readability, and also consider the union (without repetition) of sentences that are merged, rather than intersection. (Filippova and Strube 2008) introduce the use of semantic information to avoid incorrect fusions. (Tanaka et al. 2009) use a similar approach to address a very similar task called sentence revision, which considers always the lead sentence as the base sentence, and adds additional information from the rest of the document. Sentence fusion has also been accomplished by relying on tokenization and tagging rather than full parse trees by (Filippova 2010).

Semi-extractive summarization approaches rely on a syntactical manipulation of the source text to form new sentences. They typically manage to fit more salient information in the summary by removing extraneous details and joining similar information from different sources, thus also avoiding redundancies that are typical of extractive summarization. From the standpoint of the ITG model, the operations performed can hardly be considered a conceptual transformation from a source text representation to a summary representation, because they are conducted at a purely syntactical level. In this way, semi-extractive summarization remains similar to extractive summarization in terms of its capacity for text understanding.

Sentence compression and sentence fusion seem to lead to an improved performance in some cases, but it does not seem to have been tested on shared data sets from evaluation campaigns, and compared against state-of-the-art extractive systems. An exception to this statement is sentence splitting, but it performed rather poorly overall.

## 2.4 Non-Extractive Summarization

Contrary to the aforementioned strategies, non-extractive summarization typically relies on some form of text understanding in its process to generate a summary. The sentences that appear in non-extractive summaries usually cannot be generated by extraction and syntactical manipulations, often including new (unseen in the source text) sentence structures and new lexical units. Having access to some form of conceptual representation of the source text and/or the summary, it is often possible, in non-extractive summarization, to organize the structure of the summary in a meaningful way, as opposed to facing the problem of sentence ordering, typical of extractive and semi-extractive approaches.

A lot of work in artificial intelligence in the 1970s has been conducted on semantic analysis and has led to some applications to text summarization. This paper focuses instead on approaches that can be implemented in an unrestricted setting, as opposed to a controlled language setting or similar artificial scenarios where full semantic analysis may become feasible.

The FRUMP system (DeJong 1982) is a non-extractive summarization system that was applied to unrestricted text from a newswire agency, using an Information Extraction (IE) system. The text's topic is identified as one of the known topics, and the text is mapped to that topic's *sketchy script*. A sketchy script describes domain-specific world knowledge that the system uses to predict probable events in a chain of events as typically reported. Topics and sketchy scripts are manually written. They are used by the system to disambiguate word senses where necessary, infer some events (that were not reported but implied, or simply not recognized by the system's analysis) and drive the generation process. The output summary is typically an information-

rich generated sentence. DeJong reports on the difficulties of good coverage from the various resources (dictionary, analysis tools, topics, sketchy scripts), with over half of the missing or wrong stories associated to the lack of coverage of language and world knowledge resources.

This idea of using IE to accomplish non-extractive summarization has been used in other works as well. The SUMMONS system (Radev and McKeown 1998) produces summaries about multiple sources on terrorist activity using a pre-existing information extraction system to provide filled templates about each document. They report the source of each information item, reporting contradictions and changes over time. Discourse planning is used as part of the generation process. The RIPTIDES system (White et al. 2001) is another IE-based non-extractive summarizer using filled scenario templates as the source of information for the summary, and then generating a summary from these templates.

In all three IE-based systems presented here, the IE system used was designed for a purpose other than summarization. The summarization task was tackled from the angle of natural language generation from data, not really as a text-to-text task. This shows that these systems have a fully developed interpretation step within the ITG model. FRUMP reaches some minor transformation in which events can be inferred, and some selection and comparison of information is performed in SUMMONS and RIPTIDES to account for the multi-document task, but this accounts only for a fraction of the work; in all cases, the text and summary representations are the same. In other words, what *can* be extracted by IE is considered as being what *should* appear in the summary. The generation step is also much more in line with the ITG model than with extractive techniques. FRUMP and SUMMONS have not been formally evaluated against other automatic summarization systems. RIPTIDES showed an improvement over a simple extractive baseline.

A different approach, coined as knowledge-based text summarization, is the TOPIC summarization system (Hahn and Reimer 1999). The source text is parsed to map its content to a terminological knowledge representation based on manually crafted domain knowledge. Saliency operators yield paragraph-level topic descriptions which can then be combined into a text graph of the whole document, which presents a hierarchy of its topics. A user may then select topics of interest by traversing this text graph, or a summary may be generated automatically with a degree of specificity that depends on how deeply the graph is explored. This work has a custom conceptual representation of the text in a form similar to templates, and also a different representation for the summary in the form of text graphs. The ITG model is thus fully deployed in TOPIC. This system is able to produce indicative summaries by exploring only the top levels of the text graph, and much more informative summaries as the text graph is explored. It does not appear that the system was completed and ready to be used widely with unrestricted text; no evaluation other than a qualitative assessment of selected outputs was provided.

Other abstractive summarization systems focus on adding indicative information about the source documents (Kan et al. 2001) (Saggion and Lapalme 2002). Indicative summaries provide information *about* the source text rather than only information found *within*. This cannot be done using only extraction or semi-extraction, given that it requires some understanding of the source text and of what kind of information should be reported to the user, usually including other sources of information than just the original text. A recent example comes from (Zhang et al. 2013), who propose to summarize Twitter discussions on a given topic by classifying related speech acts, and reporting separately the most salient statements, questions, suggestions and comments made by the tweet authors, in order to generate short summaries that are both informative and indicative.

### 3. K-BABS: A High-Level Approach for Abstractive Summarization

Intuitively, an ideal summarizer should be capable of reproducing or emulating all operations that humans perform when summarizing. According to observations by (Hasler 2007), the operations of deletion, insertion, replacement, reordering and merging, can allow humans to transform an extract into a good abstract. Many of these operations, while doable by a machine, are hard to apply at the right place in a predictable way, because they may be performed by humans for reasons that depend on a deep understanding of the text and world knowledge, and such understanding and knowledge are not typically available when machines perform operations such as reordering and merging. (Endres-Niggemeyer 1998) shows that all steps taken by human summarizers can be simulated by a computer program, but these steps go far beyond rewriting and are more about decision making, which is the real challenge, rather than the production itself. Using only rewriting rules without world knowledge, such as what is usually done in semi-extractive summarization, is probably insufficient if the aim is to reach a near human-level performance.

Thus, an ideal summarizer should have access to a way of understanding the source text based on world knowledge in order to generate a good summary. Given access to such resources, there is no reason to limit the summarizer only to rewriting rules, and abstractive (non-extractive) summarization is warranted. In other words, there is a need for intermediate representations of the text and the summary, as in the ITG model. A way to analyze the source text in a deep semantic way, to transform the resulting source text representation into a summary representation, and to generate natural language from that representation, is needed.

To make this ideal a reachable goal, and given the kind of tools available today for analysis of the source text, the transformation and generation steps of the ITG model may be conceptually wrapped up into a single operation, encoded in an *abstraction scheme*. The text is first analyzed into some intermediate representation, and then abstraction schemes are applied to transform this representation directly into summary text. This proposed approach is called **Knowledge-Based Abstractive Summarization (K-BABS)**.

#### 3.1 Abstraction Schemes

Our implementation of abstraction schemes will be described in section 4.3, but they can be implemented in a variety of ways along the following principles:

1. Abstraction schemes *identify patterns in the source text representation*, which may include variables that need to be instantiated. For example, given a predicative intermediate representation, we could be interested in identifying this simple pattern, related to walking:  $WALKER(X), LOCATION(Y)$ , where  $X$  and  $Y$  are variables to be instantiated and the comma represents the “and” operator.
2. Abstraction schemes *provide a generation template*, which may include variables from the pattern. The template will generate textual output when the abstraction scheme’s pattern is identified in the source text representation. For example, the following is a simple generation template related to the previous pattern:  $X$  walks  $Y$ , which could be instantiated for example as  $John$  walks in the park.

Applying an abstraction scheme is thus equivalent to applying a transformation from source text representation directly to a textual output.

Relying on this kind of transformation has several advantages, which make K-BABS both flexible and scalable. The process of applying abstraction schemes by the summarization system

remains always the same, making this approach flexible from a programming standpoint, since new tasks only require a new set of abstraction schemes, obtained either manually or automatically. Increasing the coverage of the types of information to be described in the summary depends only on adding more abstraction schemes, in a relatively additive manner. The abstraction schemes can be simple and few in a first implementation, and become much more complex and numerous as new domains and new summarization scenarios are covered. Re-using abstraction schemes or modifying existing ones during this process make this approach very scalable.

The following toy example illustrates some possible uses of abstraction schemes in 3 different summarization scenarios:

John was very nervous as he walked into a student-packed classroom on the morning of the last day of high school. Copies were passed and three long hours later everyone was done. John was the first out, feeling confident that he would pass. Indeed, a month later he was proud to receive a diploma.

(1) **Informative summary:** John walked into a classroom. Copies were passed. A month later, he received a diploma.

(2) **Informative summary with reasoning:** John attended an examination taken on the last day of class. He obtained a passing grade. He was awarded his high school diploma.

(3) **Indicative summary with reasoning:** This story is about John, an examination that he attended, a diploma that he received, and the emotions that he felt throughout.

In order to write summaries such as (1), an information extraction system can fill slots about events related to going to class. Then, abstraction schemes are applied on this intermediate representation to transcribe IE template slot fillers into natural language for the summary. Assuming a simple IE template for the event of walking, our above abstraction scheme example can be reused, using the pattern: WALKER(*X*), LOCATION(*Y*); and the generation template: *X* walked *Y*; generating the summary phrase John walked into a classroom. The rest of the summary can be generated with similar abstraction schemes.

While summary (1) is not extractive, no new information is added either. Information that lies outside the scope of the event of interest is ignored and no reasoning is performed. Existing IE-based non-extractive summarization methods (Radev and McKeown 1998) (White et al. 2001) work in such a way, and in this sense they are directly compatible with the more general K-BABS approach.

Summary (2) makes use of abstraction schemes to perform a certain amount of reasoning. One way to go about it would be to detect a script that these events fit into, and then inspect the appropriate script, say, about events related to high school education. The source text representation becomes very high level, and an abstraction scheme pattern could be represented simply as this: SCRIPT\_TRIGGERED(Examination), PROTAGONIST(*X*). The generation template would then be *X* attended an examination.

Reasoning is truly performed by a complex analysis tool that yields high-level information in the source text representation. In theory, abstraction schemes patterns can also be used to



accomplish the same kind of reasoning with access to the same kind of predicative intermediate representation as before. The pattern only needs to be more intricate and specific: `LOCATED_INTO(X, classroom)`, `PASS(_, copy)` at time `T1`, `BEING_DONE(X)` at time `T2`, `TEMPORALLY_BEFORE(T1, T2)`. The generation template would then be the same. Creating such an abstraction scheme requires some understanding, but applying the scheme is simple and straightforward.

Indicative summaries such as (3) may also be generated by applying abstraction schemes. In this case, the schemes would call for the recognition of entities and actions that involve them, assuming an interpretation step which provides some kind of semantic tagging and syntactical parsing. Again, the abstraction schemes perform transformation and generation in a single step. For example, a scheme could detect if there is an entity of type `person`, and it is observed as the subject of the verb `to feel` or the subject of the verb `to be` followed by a word that can be found in a list of emotions. If this occurs more than once in the source text, then the output should include a mention that the story is in part about emotions felt by that entity.

### 3.2 K-BABS Architecture

The basic premise of K-BABS is that well-defined abstraction schemes can be easy to read and execute by the machine, while implicitly performing transformations of arbitrary complexity on the source text representation, leading to natural language text that can be included in the summary. These abstraction schemes can be defined as transformations that match certain types of language manifestation patterns from an intermediate text representation and output generated natural language to satisfy a specific need that depends on the goals set for the summarizer. Because abstraction schemes encode not only world knowledge, but also knowledge about the task to perform, the knowledge base which contains all the schemes is called a *task blueprint*.

This implies a simple high-level architecture for K-BABS: first, analyze the source text into some intermediate representation; second, apply abstraction schemes from the task blueprint to generate the summary directly. Generally speaking, the task blueprint should not only provide abstraction schemes, but also a generation plan as well, to provide instructions for assembling the final summary while applying desired constraints on it, such as determine which abstraction schemes to favor in order to respect a size limit.

This approach thus depends on two key factors: the richness and accuracy of the intermediate representation provided in the analysis step, and the quality and coverage of the task blueprint. A set of analysis operations that should be sufficient in most cases includes: syntactical parsing, semantic tagging, word-sense disambiguation, and coreference resolution. The summary scenarios described in the toy example shown above can all be solved with abstraction schemes given access to such an analysis.

## 4. ABSUM: an Implementation of the K-BABS Approach

The **Abstractive Summarizer of Université de Montréal (ABSUM)** has been implemented following the K-BABS approach, which has the advantage of providing a direct control over (and some understanding of) the content of the summary. It is designed to address the summarization task known as guided summarization and makes use of no pre-existing event extraction system.

ABSUM is split into three modules: analysis, which provides syntactical and semantic information about the source documents, task blueprint, which provides a full description (made up of hand-written abstraction schemes and a generation plan) of what the summarization system should do for each category, and the summarization system itself, which mostly runs the abstraction schemes from the task blueprint on the representation of the source documents given by the analysis. The summarization system itself is split into 3 steps, as illustrated in

Figure 1: pattern matching to identify aspect candidates, content selection to determine which candidates are aspect answers, and generation of summary in natural language. The remainder of this section describes the task of guided summarization, each of the three modules, and the process of generating hybrid summaries. These descriptions tend to remain rather factual and section 6 will provide a broader discussion.

#### 4.1 Guided Summarization

The organizers of the evaluation campaigns Document Understanding Conference<sup>2</sup> (DUC), running from 2001 to 2007, and its successor the Text Analysis Conference (TAC), running a summarization task from 2008 to 2011, aimed to challenge system developers to move toward better text understanding, notably by suggesting new tasks like topic-oriented summarization, summarization at higher compression rates, multidocument summarization and update summarization. In hindsight, many observers are under the impression that it mostly had the effect of promoting a greater sophistication of shallow statistics-based approaches, which is laudable, but probably not what was anticipated.

The latest summarization task starting at TAC in 2010, guided summarization, is probably the one best designed yet to motivate a move towards non-extractive approaches – it is also its stated goal. Guided summarization is an oriented multidocument task in which a category is attributed to each cluster of 10 source documents to summarize in 100 words or less. Five categories were selected: Accidents and Natural Disasters, Attacks, Health and Safety, Endangered Resources, and Investigations/Trials. For each category, a list of aspects to cover in the summary is given.

Figure 2 shows the aspects for the categories `ATTACKS` and `ACCIDENTS AND NATURAL DISASTERS`. These are the two categories currently supported in ABSUM. They were chosen because they have similar aspects, and because they include the aspects `WHEN` and `WHERE`, which seemed easier to address at first. The three other categories in TAC 2011 have fewer aspects and none about location and date; they are: `HEALTH AND SAFETY`, `ENDANGERED RESOURCES`, and `INVESTIGATIONS AND TRIALS`.

- 2.1 `WHAT`: what happened
- 2.2 `WHEN`: date, time, other temporal placement markers
- 2.3 `WHERE`: physical location
- 2.4 `PERPETRATORS`: individuals or groups responsible for the attack
- 2.5 `WHY`: reasons for the attack
- 2.6 `WHO_AFFECTED`: casualties (death, injury), or individuals otherwise negatively affected
- 2.7 `DAMAGES`: damages caused by the attack
- 2.8 `COUNTERMEASURES`: countermeasures, rescue efforts, prevention efforts, other reactions

**Figure 2**

Aspects for the `ATTACKS` category of TAC’s guided summarization task. The category `ACCIDENTS AND NATURAL DISASTERS` has the same aspects, but excludes aspect 2.4.

#### 4.2 Analysis

The goal of the analysis module is to provide an intermediate representation of the source documents that will then be used as input for applying the abstraction schemes. This includes

---

<sup>2</sup> [duc.nist.gov](http://duc.nist.gov)

dependency parsing of all the sentences in the source documents, morphological analysis to identify lemmas, recognizing named entities, and resolving dates. Because the task blueprint is only applicable if the syntactical and semantic information is relatively accurate, several strategies have been elaborated with the only purpose of increasing the accuracy of the parsing. These strategies are described here in great detail, in a desire to be thorough and transparent, and for this work to be reproducible.

**4.2.1 Preprocessing.** Preprocessing produces text files in one-sentence-per-line format from the SGML input files. Regular expression substitutions are used to clean up the text and make it uniform and as easy to parse as possible. Sentence segmentation is performed by a Java program which uses the class `java.text.BreakIterator` and a list of common abbreviations. Some sentences are filtered, such as those ending with an exclamation or interrogation mark.

**4.2.2 Sentence Parsing and Morphological analysis.** The freely available Stanford Parser is used to perform dependency parsing (“collapsed dependencies”) on each sentence of each source document (de Marneffe et al. 2006). The also freely available GATE morphological analysis tool (Cunningham et al. 2002) is called to identify the lemma associated to each token, given the part-of-speech attributed to it by the Stanford Parser. Any improvements to the parsing might affect part-of-speech identification and in turn the accuracy of the morphological analysis.

**4.2.3 Semantic Annotations and Additional Cleaning.** An iterative process was designed to reduce parsing errors by using information from the semantic analysis to make the parsing task simpler at each step. In total, the Stanford Parser, as well as the morphological analyzer, are run three times, with two additional steps in between, called Preparation Steps 1 and 2.

*Preparation Step 1.* The cluster frequency and document frequency of all the lemmas are computed at this point, before any modification is made to the text. The cluster frequency is the frequency of a lemma in all the documents. The document frequency is the number of documents that contain at least one instance of a lemma. Lemmas are saved only as strings, not as a string and part-of-speech pair. These frequencies are used later during content selection.

To remove some badly parsed or content-poor sentences that the summarization system will have trouble dealing with, sentences containing less than 6 words, with no verb, are removed.

The named entity recognition system Stanford NER (Finkel et al. 2005) is run on all the source documents. The output provides tags on strings identified as either “person”, “location”, “organization” or “date” (other tags are ignored). In the case of an ambiguity, the most frequent tag for a string is selected. For each string tagged person, the given name, nobiliary particle, and surname are identified. The tag of a person’s given name or surname will be changed to “person” if it was something else.

A new text document is generated for the purpose of the second parsing iteration, cleaned of named entities as follows. Person, location and organization strings are replaced by a new unique string. Given names and surnames that appear alone are also substituted by the full name’s replacement string, in the unambiguous cases. The original strings are saved for use during the output stage of the summarization only. The motivation for these replacements is to simplify the work of the parser, where known noun phrases are replaced by a single entity that is unmistakably identified as a noun by the parser.

*Preparation Step 2.* Some common sentence structures in the corpus put a date in a position which sometimes leads to parsing errors, e.g. `The government on Sunday sent a rescue team to [ . . ]`, where `Sunday` will be incorrectly identified as a prepositional complement of `government`. Therefore, each date found in the text is removed, along with

accompanying words, in phrases such as *by last Tuesday* and *yesterday morning*. Regular expressions were built to identify a wide variety of such constructions. Each date is then resolved to a calendar date, month or year. The resolution of some relative dates requires the tense of the closest verb to be inspected. Dates that cannot be resolved are left untouched in the text. Parents of a removed date in the dependency tree will be identified as bearing that date. When a word could thus be attributed more than one date, neither will be kept.

After this final preparation step, the syntactical parser and morphological analyzer are run one last time, with observed improved accuracy. Figure 3 shows an example of the analysis provided by the Stanford Parser for a simple sentence. This is the kind of representation available to the abstraction schemes in the task blueprint, and on which the summarization system runs the pattern matching rules.

A second suicide bomber was reportedly killed by Israeli police officers before he blew himself up in the mall.

DET(bomber, A)	AMOD(bomber, second)	NN(bomber, suicide)
NSUBJPASS(killed, bomber)	AUXPASS(killed, was)	ADVMOD(killed, reportedly)
ROOT(ROOT, killed)	AMOD(officers, Israeli)	NN(officers, police)
AGENT(killed, officers)	MARK(blew, before)	NSUBJ(blew, he)
ADVCL(killed, blew)	DOBJ(blew, himself)	PRT(blew, up)
DET(mall, the)	PREP_IN(blew, mall)	

**Figure 3**

A sentence from cluster 9 of the test set, with the topic “Dimona Attacks”, and its representation by syntactical dependencies provided by the Stanford Parser in the analysis module.

#### 4.3 The Task Blueprint

<b>Scheme: killing</b>		
Pattern Matching	SUBJ_RELATIONS( <i>kill_verbs</i> , X)	→ WHO(X)
	OBJ_RELATIONS( <i>kill_verbs</i> , Y)	→ WHO_AFFECTED(Y)
	PREP_OF( <i>murder_nouns</i> , Y)	→ WHO_AFFECTED(Y)
	PREP_BY( <i>murder_nouns</i> , X)	→ WHO(X)
Generation Template	X <i>kill_verbs</i> Y	
<b>Scheme: event</b>		
Pattern Matching	PREP_IN( <i>event_lemmas</i> , X), LOCATION(X)	→ WHERE(X)
	PREP_IN( <i>event_lemmas</i> , X), ORGANIZATION(X)	→ WHERE(X)
	PREP_AT( <i>event_lemmas</i> , X), LOCATION(X)	→ WHERE(X)
	PREP_AT( <i>event_lemmas</i> , X), ORGANIZATION(X)	→ WHERE(X)
	DEP( <i>event_lemmas</i> , Y), DATE(Y)	→ WHEN(Y)
Generation Template	On Y, Z occur at/in X	

**Figure 4**

Abstraction schemes **killing** and **event**. The pattern matching rules define how to detect aspect candidates from the dependency parsing annotations and semantic information detected by the analysis module. The generation template defines how to realize a sentence for output. Notation: *word* or *lemma*, *variable*, *lemma group*, PREDICATE OR ASPECT. The special predicate DEP is the set of all the syntactical relations from the parser and the lemma group *event\_lemmas* is a set of many verbs and nouns strongly related to the category.

The task blueprint contains instructions about how to find aspect candidates from documents, and how to generate sentences and a full summary from that information. It explicitly defines

*abstraction schemes* and provides a generation plan for generating abstractive summaries on a predefined domain (or category). The way in which the blueprint is interpreted and used by the system is described in section 4.4. The methodology that was used to construct the task blueprint is given in section 4.3.3.

**4.3.1 Abstraction Schemes.** An abstraction scheme defines pattern matching rules and a generation template, with the goal of finding a specific type of category-relevant information in the source documents, and generating a single sentence for output in the summary. Two example schemes of the task blueprint for the ATTACKS category are given in figure 4.

*Pattern Matching Rules.* These rules contain one or more predicates that must be unified with dependency parsing relations or semantic knowledge. Such predicates usually have three parts: a predicate type which is the equivalent to a type of syntactical relation, a head lemma, and a child lemma. Sets of relations and lemmas are used instead of a single instance, in practice. For example, a rule that intends to find some entity that is killed can be represented this way:  $OBJ\_RELATIONS(kill\_verbs, Y) \rightarrow WHO\_AFFECTED(Y)$ , where *obj\_relations* is a set of syntactical relations from the Stanford Parser like *DOBJ* and *NSUBJPASS*, and *kill\_verbs* is a set of verb lemmas with related meaning and similar syntactical and semantic structures, such as *kill*, *murder*, *assassinate*, *shoot*, *stab*, *poison*, etc. Another type of predicate unifies a property to a variable, such as in the first rule of the scheme **Event**, where the predicate *LOCATION(X)* is found. Pattern matching rules determine strictly how candidates are found, be it candidates for the representative lemma of a lemma group or for category aspect candidates.

*Generation Template.* Each abstraction scheme must include one generation template, which describes how to generate a sentence containing the information found. It assumes that a selection within the candidates found by the unification of the pattern matching rules with the observed source documents is performed. Continuing with the example of the scheme **kill**, its generation template could simply be:  $X kill\_verbs Y$ , which specifies that a verb from within the lemma group *kill\_verbs* should have the subject *X* and the direct object *Y*. For each element of the generation template, a mention is made of whether or not that element is absolutely required to appear in the generated sentence. It is also mentioned whether or not the element can be several coordinated noun phrases or only one. In this example, the victim variable *Y* is required to be present and can contain several noun phrases, whereas it was decided that the perpetrator variable *X* may be omitted and that it must contain at most one noun phrase.

**4.3.2 Generation Plan.** The task blueprint must include a generation plan to describe how the summary will be constructed from the sentences generated by the abstraction schemes. The generation plan lists the schemes in the order in which their generated sentence will appear in the summary, and in this way provides the full structure of the summary. The processing of each abstraction scheme actually takes place sequentially in the summarization system, according to the order set out here in the generation plan. 5 gives the generation plan for the ATTACKS category.

Being responsible for the structure of the summary, the generation plan may also provide indications to the abstraction schemes about what content to include or exclude from the summary. In the case of the ATTACKS category, for instance, the generation plan will prohibit the same entity to be declared as having been killed by someone, and also as having died, or as having been injured by that same person. Similarly, if the summary declares that some Mr. X has shot someone, the generation plan will prohibit the abstraction scheme **suspecting** from generating that Mr. X is suspected of conducting the attack. Without the generation plan, these kinds of redundancies would appear in the summaries by applying each abstraction scheme blindly.

Abstraction Scheme	Example structure of the generated sentence
<b>event</b>	On a date, an attack/murder/shooting/etc. occurred at/in a location.
<b>beingHitGeneric</b>	X was attacked/hit/struck.
<b>killing</b>	X killed/murdered/shot/etc. Y.
<b>dying</b>	X died.
<b>injuring</b>	X wounded/injured Y.
<b>destroying</b>	X was damaged/destroyed.
<b>arresting</b>	X was arrested.
<b>suspecting</b>	X was suspected of conducting the attack.
<b>responsibilityClaim</b>	X claimed responsibility for the attack.
<b>helping</b>	X sent/provided/offered support/aid/help.
<b>beingRescued</b>	X was rescued.
<b>evacuating</b>	X was evacuated.

**Figure 5**

An ordered list of abstraction schemes that serves as the generation plan for the category ATTACKS.

**4.3.3 Methodology for Writing Task Blueprints.** Two task blueprints were written for AB-SUM, one on the ATTACKS category, and the other on the ACCIDENTS AND NATURAL DISASTERS category.

Several tools were used, starting with the development set made up of the TAC 2010 clusters, including example source documents, topics, and reference summaries for each category. An English thesaurus was consulted to help populate the various lists of the task blueprints, including some lists which are later described in the summarization system (section 4.4) like stoplists. This was also complemented by internet queries about English words and their usage. A list of the available syntactical dependencies of the Stanford Parser and an interface for running it on sentences are also necessary, for reference, but also for discovering typical sentence structures that may be analyzed in an incorrect or unexpected way by the parser.

The task blueprint writing process involved a lot of back and forth, between consulting the reference summaries to determine what should be said, observing the development set's documents and their language to determine what to look for and how it is analyzed by the parser, making modifications to the blueprint accordingly, and running the system on the development set to verify the effect of those modifications. Because the development set was so small (7 topics and a total of 70 documents per category), a lot of effort was spent trying to expand the rules and lists to as many unseen cases as could be foreseen.

#### 4.4 Summarization System

The summarization system's input consists of the analysis of the source documents, and the task blueprint for its category, which provides the knowledge necessary to write an abstractive summary. The top-level algorithm for writing a summary is to read the generation plan and sequentially generate a sentence for each of the abstraction schemes, when possible. The generated sentences are concatenated together, separated by a space character.

The remainder of this section describes how to generate a sentence from a scheme description within the task blueprint, and an analyzed cluster of 10 NewsWire documents. This is accomplished in three steps: pattern matching, content selection, and sentence generation. The process will be illustrated by an example based on a document cluster with the topic "Dimona Attacks", taken from the test set, and the abstraction scheme **killing**, which appears in Figure 4. The story on the Dimona attacks is about two terrorists who attempted to blow themselves up with explosives in crowded areas; the first bomber was successful and the second was killed by

police. Figure 3 shows a sentence relevant to this example, and its representation after parsing by the analysis module.

**4.4.1 Pattern Matching.** An abstraction scheme’s pattern matching rules contain one or more predicates, as discussed in section 4.3.1. Predicates may have one or two arguments, up to one of which may be a variable linked to an aspect of the category. Example rules and predicates are given in Figure 4.

Pattern matching begins by going through all the predicates and attempting to find matches for each one within the analyzed source documents. A rule is applied if all of its predicates have a match in the same sentence, given also that a variable or a lemma group that appears in more than one predicate of the same rule is matched to the same entity.

The application of a rule leads to the creation of a *candidate* for the generation process. This is the result of an instantiation of a variable or a choice within a lemma group, which provide a suggested lemma for the content of the generated sentence. Rules that contain a variable produce candidates for a specific aspect of the summary, and those candidates are called *aspect candidates*. Candidate identification is illustrated by the right-hand side of the rules in Figure 4.

On the example sentence from Figure 3, pattern matching attempts to apply the rules for the abstraction scheme **kill** (Figure 4). For example, the relation `NSUBJPASS(killed, bomber)` yields the aspect candidate `WHO_AFFECTED(bomber)`, because each element of the syntactical relation matches the pattern: `NSUBJPASS` is a member of the syntactical relations group *obj\_relations*, and `kill` is a member of the lemma group *kill\_verbs*, so `bomber` instantiates `X` in the pattern matching rule. Similarly, the syntactical relation `AGENT(killed, officers)` yields the aspect candidate `WHO(officer)`.

An aspect candidate is given a date if its source word bears a date or if one of his parents in the dependency tree does. Dates were attributed to words during Preparation Step 2 of the semantic annotation part of the analysis, described in section 4.2.3.

**4.4.2 Content Selection.** Content selection is the process of selecting the aspect candidate that will become the *aspect answer* for each aspect, and selecting its modifiers for generation, where appropriate. It also includes the selection of a lemma from each lemma group that must appear in the scheme’s generated sentence, which is done in the same way as selecting aspect answers.

The content selection heuristics presented here were selected to maximize the accuracy of the information selected, and the readability of the sentences generated, because false and unreadable sentences often leave a bad impression to the reader who might dismiss the whole summary as useless. In other words, precision was somewhat favored over recall. This comes at the cost of preventing some valuable information from making it into the summary, because the number of abstraction schemes that lead to a sentence being generated in the summary is lower than could otherwise be possible.

*Filtering Aspect Candidates.* Before selecting aspect answers, the aspect candidates resulting from pattern matching are filtered according to several criteria. For instance, stoplists are used to avoid words that are undesired answers for a category aspect, such as `war`, which could be identified as the perpetrator of an attack, because it “caused deaths”, or `other` because it refers to something else and that no coreference resolution is performed. Similarly, all words that can be considered a type of attack are not considered.

In the corpus used, some sentences can be repeated verbatim in more than one source document (often because they are two articles from the same press agency), leading to repeated aspect candidates. Repeated aspect candidates from identical sentences are filtered out.

The date is also a big factor in filtering out some candidates. In the categories covered, the date of the main event is considered to be the date identified in the special scheme called **Event**,

which is called first by each generation plan. In the other schemes, lemmas for which the most frequent date associated with its aspect candidates is not within a month of the main event's date are removed.

Aspect candidates for which the lemma has been selected in previous schemes for the same aspect are filtered if the current abstraction scheme disallows it.

Finally, all lemmas with a frequency of one within the remaining aspect candidates are removed, in case the information is not salient enough and to avoid relying too much on a single sentence parse, given that the parser makes mistakes. This is a heavy-handed choice which greatly emphasizes precision over recall, as per the intended design.

*Selecting Aspect Answers.* An aspect answer is a lemma selected to be the head of the noun phrase that will be generated to answer a category aspect. It is selected among the lemmas given by all the aspect candidates for that aspect (after filtering). If a coordinated noun phrase is allowed by the generation template, then all lemmas are used. Otherwise, the lemma that has the highest frequency among the remaining aspect candidates is selected. This choice of the lemma with the highest frequency as the noun phrase head may change later, if the resulting noun phrase with a different head would still contain that highest frequency lemma, though as a modifier.

To continue the Dimona attacks example, the application of pattern matching rules for the abstraction scheme **kill** on all the source documents yielded the following aspect candidates for the perpetrator (with frequencies in parentheses): `police(4)`, `officer(3)`, `blast(1)`; and the following for the victims: `bomber(4)`, `second(2)`, `attacker(1)`. Filtering removes `blast` and `attacker` because they have a frequency of 1. Thus, for now `police` and `bomber` are the selected aspect answers, with `officer` and `second` kept in reserve as alternative NP heads, and as favored modifiers. It is also needed to select a verb from within the *kill\_verbs* group, and it is the lemma `kill` which was selected given the observed verbs: `kill(17)`, and `shoot(10)`.

*Selecting Noun Modifiers for the Noun Phrase.* Once a noun lemma has been selected as a candidate answer (or as one of several candidate answers in the case of a tie for highest frequency), its modifiers must be selected to form a noun phrase. Four types of modifiers are considered: noun-noun modifiers, numerical modifiers, adjectives, and "of" prepositional modifiers (a noun that modifies the head noun through the preposition `of`). These are directly observable in the dependency parse tree.

The combination of noun modifiers selected must be one that was observed at least once as part of a noun phrase that was an aspect candidate. To select which combination of modifiers is selected, a score for each combination is computed, based on all the occurrences of its modifiers as a modifier to the head noun in the source documents. The basic score is an average of the frequency of the modifiers of a combination. The frequency of each modifier is multiplied by 100 first, when the modifier is a noun-noun modifier whose lemma was another candidate for that aspect. A combination of modifiers which includes all the modifiers of another combination is given the score of the shorter one instead, if this improves the score – this favors a combination of more modifiers where available. In the end, the combination of modifiers with the highest score is selected.

Special care is given to numerical modifiers. If written in English, they are all translated into numerical strings (except for `one` which is only used to identify the singular number). This ensures that either spelling is equivalent and that numerical modifiers occupy less space in the summary.

When a prepositional modifier is selected, the whole process of selecting noun modifiers, capitalization, number and specifier must be recursively called on its lemma, to define this additional noun phrase, part of the larger one. This is not done on noun-noun modifiers, because



the Stanford Dependency Parser does not usually have a noun phrase as the noun-noun modifier of the head noun, resorting instead to a list of nouns each being a modifier to the head.

When it is possible to select another head lemma for the noun phrase while still keeping the aspect answer in the noun phrase (because that lemma appears as a modifier of the new head lemma), the whole process is restarted, this time with this new head, and requiring that the aspect answer appears as a modifier.

This occurs in the Dimona attacks example discussed previously, where the aspect answer `police` appears as a modifier of the other less frequent aspect candidate `officer`, thus this latter word is used and the modifiers selected are `Israeli` and `police`. The modifiers for the other aspect answers are `second` and `suicide` for `bomber`, and `none` for `second`.

*Named Entities and Capitalization.* Since lemmas in this system are always considered to be lowercase, it is needed to determine if the head noun of the noun phrase and any other nouns occurring as noun-noun modifiers require capitalization or not. If the lemma is one of the unique strings that were used as replacement for locations, organizations and persons in Preparation Step 1 (see section 4.2.3), then it will be reverted to its original realization, which is a capitalized string. Otherwise, a lemma which occurs more often capitalized than not within the source documents will be capitalized.

*Selecting the Number and the Specifier.* Noun phrases with a numerical modifier will be set to plural automatically. There is an exception for numbers that are incorrectly labeled as numerical modifiers, e.g. a 7.0 magnitude earthquake. When a singular head noun is observed in the source documents with a number greater than one as a numerical modifier, that modifier does not affect the number of the head noun for generation either. Noun phrases without a numerical modifier are set to singular or plural according to how frequently they were seen as one or the other in the source documents.

By default, singular noun phrases are attributed an indefinite article (`a` or `an`) as specifier and plural noun phrases are attributed no specifier. A noun phrase that contains a named entity is instead attributed the definite article (`the`) as specifier, except for locations that have not been observed with a definite specifier in the source document. Lemmas that have never been observed without a definite article in the source documents are attributed one for generation as well. Noun phrases that include a prepositional modifier always use the definite article. The lemma that is selected as the main event noun by the **Event** scheme will be attributed a definite article as specifier if it appears again in the summary. Finally, some lemmas always get a definite article, because of their special role in a category or for general reasons of readability, such as words with a meaning similar to `suspect` and `assailant`.

In the Dimona attacks example, all three noun phrases are singular and given the indefinite article `a` or `an`.

*Removing Redundant Noun Phrases.* Noun phrases generated to fill up a slot in the generation template of an abstraction scheme may be allowed to be coordinated or not, as mentioned before. In the case where a coordination is possible, redundancy between the coordinated noun phrases must be avoided.

When the head lemma of a noun phrase appears as a noun-noun modifier in another noun phrases, that noun phrase is removed. Similarly, a noun phrase that already appears as a prepositional modifier will not be included in the coordination either. This is to avoid cases such as “The hurricane hit the east coast of the United States and the United States”.

In the ongoing example, the noun phrase with `second` as its head is removed because this lemma appears as a modifier in the other noun phrase with which it would be coordinated. This avoids generating: A second and a second suicide bomber were killed.

*Special Case: People.* The lemma `person` often occurs as the head noun of noun phrases in the corpus used, for the categories of interest. Depending on the context, words with an implicitly similar meaning will be used in the source documents, such as `others`, `victims`, `casualties`, `people/persons`. They are all considered to be the same lemma by the system, which will always use the word `person` (and the plural form `people`) to represent them for generation.

Another option to detect this kind of information would have been to write pattern matching rules to deal specifically with finding the number of dead and injured. This may have worked equally well, but additional work would be needed in avoiding to mention the same information again in schemes that are more generic.

Because of the special role of counting people for news stories about attacks, accidents and natural disasters, the choice of a numerical modifier for the lemma `person` is critical. News stories will differ in terms of the number of deaths and so on, and favoring (quite gruesomely) the largest observed number among several options of numerical modifiers for a given aspect is the strategy that has been adopted. This tends to lead to a better summary also because of the nature of the source documents, which are often published on different dates and so the number of casualties is revised as more information becomes available. The alternative of selecting the most recent number has also been considered, but it has the downside of possibly choosing a number that was used in a context with a smaller scope; e.g. the latest article might report the number of casualties in one town, whereas the event of interest affected a large area.

A more involved approach would take into account the specificity and date of each number, and perhaps try to establish a confidence level for each one, but this was not implemented here.

*Subject-Verb-Object Constraint.* Selecting the candidate answers and the lemma from a lemma group that fill up the slots of an abstraction scheme's generation template is done independently. This may lead in some cases to falsehoods and nonsense. To avoid this problem, only subject-verb-objects triples that have been observed together at least once in the source documents are allowed.

The process of selecting subjects and objects is done greedily, starting with the subject. After a rejection, the process starts over iteratively at the level of selecting aspect answers. When no agreeing subjects and objects can be found, only one element is kept, whichever has been identified as required in the generation template, if any.

In the Dimona attacks example, some source document sentences (including the one shown at the beginning of this section) indeed contain a verb that is part of the *kill\_verbs* group and that has `officer` as its subject, and `bomber` as its object. In some cases, such as this one, it may occur that the number of one of these lemmas is not the same as in the source sentence, because that decision was based on the frequency with which that lemma was used with either singular or plural.

**4.4.3 Sentence Generation and Postprocessing.** During Content Selection, all the elements of the abstraction scheme's generation template have been selected. This will include a verb and possibly noun phrases for either the role of subject or object. Cases in which no subject and no object are selected, or if a required element is missing, do not lead to a sentence being generated.

If more than one subject or more than one object has been selected, the noun phrases are coordinated and the resulting noun phrase is set to plural.

The verb is always set to the past tense because the summaries generated describe past events. If there is no subject, the verb is set to the passive voice.

The sentence is realized using the SimpleNLG (Gatt and Reiter 2009) library, which takes care of subject-verb agreement, verb tense and voice, noun phrase coordination, start-of-sentence

capitalization, and end-of-sentence punctuation. Some generation templates include a fixed end-of-sentence string to be appended, and SimpleNLG supports this possibility in a clean way.

After sentence generation, a tiny step of postprocessing reverts numbers back to the usual format that uses commas as thousand separators (the commas had been removed during preprocessing).

In the Dimona attacks example, the sentence generated for the **killing** abstraction scheme can be seen in Figure 6, where it appears as the second sentence of the summary generated by ABSUM. The other summaries generated on the test set can be found in Figure 10.

On February 4, 2008, a suicide attack occurred in Dimona. An Israeli police officer killed a second suicide bomber. An Israeli woman and three Israelis died. The military wing of the Palestinian Fatah movement claimed responsibility for the attack.

**Figure 6**

Summary generated by ABSUM on cluster 9 of the test set, under the topic “Dimona Attacks”.

#### 4.5 Hybrid Summaries

ABSUM produces very short summaries, sometimes even less than 10 words long. This is the expected behavior of a system which greatly favors precision over recall in the content selection process. Thus, the summaries generated are always far shorter than 100 words long and lack coverage of the source document. For now, ABSUM does not have a size limit in the output, and, even then, it never comes close to outputting 100 words.

In order to give ABSUM a fair chance against 100-word long summaries during the evaluation process, a hybrid alternative that uses sentence extraction to lengthen the summary and reach the desired size is considered. This hybrid summarizer is forced to start the summary with the output of ABSUM, and then completes the summary using state-of-the-art extractive techniques.

This idea to have an abstractive/extractive summarization hybrid should probably be applied to any non-extractive system that is unable to cover all of the information desired, and it has been implemented before in the RIPTIDES summarizer (White et al. 2001). In the case of ABSUM, aspects such as “WHY” and “COUNTER-MEASURES” of the ATTACKS category will not be fully covered by the pattern matching rules, even in the best of cases, and in some cases, other aspects will not be covered correctly either. Bringing in some extracted sentences fills these gaps and provides a safeguard for when a topic is outside the range that is meant to be covered by ABSUM.

The automatic summarizer with the best overall responsiveness amongst all the participants of TAC 2011 was CLASSY (Conroy et al. 2011). Researchers John Conroy and Judith Schlesinger generously agreed to produce hybrid summaries that begin with the output of ABSUM and end with the output of their CLASSY 2011 extractive summarization system. Slight modifications to their system were necessary in order to adjust to the additional abstractive input and to try to apply their redundancy avoidance to that content.

## 5. Evaluation

ABSUM has been evaluated according to the recent standards and methodology developed in the DUC and TAC evaluation campaigns. This includes the manual scoring of overall responsiveness, linguistic quality and content. Automatic scoring such as with ROUGE (Lin 2004) has not been performed because it is meant as an approximation of manual scoring, which is more reliable, and would therefore be meaningless in the face of manual scoring results.

The evaluation set contains 18 clusters of 10 NewsWire articles from TAC 2011's guided summarization track. The evaluation campaign provided 44 clusters, but only the ones with the categories `ATTACKS` and `ACCIDENTS AND NATURAL DISASTERS` were included, because other categories were not covered yet in ABSUM.

Five automatic summarizers were evaluated alongside with a human reference:

**ABSUM** as described in section 3.

**CLASSY 2011** System with the best overall responsiveness in the standard guided summarization task of 2011. Output taken directly from TAC's official results.

**ABSUM/CLASSY Hybrid** Summaries that begin with ABSUM's output, and are completed by CLASSY 2011's system.

**Extractive baseline** This baseline extracts as many of the first sentences of the most recent document as possible without exceeding 100 words. Output taken directly from TAC's official results.

**Abstractive baseline** This is an abstractive approach that was submitted at TAC in 2011 (Genest and Lapalme 2011b). It splits sentences based on subject-verb-object triples in the analysis tree to produce short sentences, and selects the most salient one based on statistics-based heuristics.

**Human-written models** The human-written model was selected randomly for each cluster from the four available models, and used as a reference and upper limit for the evaluation.

The results of the manual evaluation are shown in Table 7, and each score is described below.

	Overall. R.	Ling. Qual.	Mod. Pyr.	Size	Content Density
ABSUM	2.07	3.67	0.277	22.6	0.0119
CLASSY 2011	3.20	3.39	0.520	98.0	0.0053
ABSUM/CLASSY Hybrid	3.31	3.28	0.600	97.6	0.0061
Extraction baseline	2.70	3.76	0.395	84.5	0.0046
Abstraction baseline	1.70	1.44	0.451	97.9	0.0046
Human-written models	4.54	4.69	-	96.6	-

**Figure 7**

Scores of overall responsiveness, linguistic quality, modified Pyramid, and content density, for ABSUM, the CLASSY 2011 summarizer, a hybrid of ABSUM and CLASSY 2011, an extractive summarizer baseline, an abstractive summarizer baseline, and human-written model summaries. The size is the average length of the summaries measured by their average number of words.

### 5.1 Linguistic Quality and Overall Responsiveness

Human assessors were asked to score the summaries according to the same guidelines as those used at TAC 2011 for overall responsiveness and linguistic quality. They can be found in Table 8. The assessors had access to the category aspect definitions, the cluster's topic name, the 10 articles, and the summaries themselves. They were not told which summaries came from what

source, and specifically asked to avoid any bias related to recognizing a summary as probably coming from the same source as another one from a different cluster.

Linguistic Quality	Assign an overall linguistic quality score based on consideration of the following factors: grammaticality, non-redundancy, referential clarity, focus, structure and coherence
Overall Responsiveness	Assign an overall responsiveness score based on both the linguistic quality of the summary and the amount of information in the summary that helps to satisfy the information need defined for the topic's category.

**Figure 8**

Manual evaluation guidelines given to the human assessors.

A total of 10 human assessors evaluated the summaries from the 18 clusters so that each summary was evaluated 3 times. Each assessor was thus given between 5 and 6 clusters to evaluate. The evaluation scores are whole integers between 1 and 5, with the following scale: very poor (1), poor, barely acceptable, good, very good (5).

ABSUM obtains a good linguistic quality score of 3.67 on average, which is much higher than the other automatic systems with the exception of the extractive baseline. This baseline usually gets good linguistic quality scores because the summary is composed of the leading sentences of an article written by a human. It is important to point out that because the size of the sample is so small, the difference of 0.28 point between the averages of ABSUM and CLASSY is not enough to conclude without a doubt that ABSUM has a better linguistic quality than CLASSY. As suggested in (Rankel et al. 2011), a Wilcoxon Signed Rank test, a paired statistical test for distributions that may not be normal, was performed, and a p-value of 0.12 was obtained, which allows for concluding that linguistic quality is probably improved in the short summaries over CLASSY, but that it is not certainly so from a statistical standpoint.

For both overall responsiveness and linguistic quality, CLASSY and the hybrid summaries are statistically indistinguishable, with the hybrid system having a slight edge in overall responsiveness, and CLASSY having a slight edge in linguistic quality. From talking with assessors, it appears that the non-redundancy criterion played the biggest role in lowering the linguistic quality scores for both of them. The hybrid also suffered from grammaticality problems sometimes caused by ABSUM, and CLASSY had a bit more problems with structure and coherence overall. Both systems greatly outperform the other automatic systems in overall responsiveness, including the baselines.

Unsurprisingly, the human-written summaries are far ahead in all respects.

To evaluate inter-annotator agreement, the number and degree of the differences in how each assessor ranks the six summaries on a given cluster was computed. For all pairs of summaries 1 and 2 and all pairs of assessors A and B, A and B are considered to agree if they both rank summaries 1 and 2 the same way (either 1 has a better score than 2, 2 has a better score than 1, or 1 has the same score as 2); A and B are considered to disagree if one of them gave the same score to 1 and 2 while the other gave different scores to one summary and the other; and A and B are considered to contradict each other otherwise, which is in cases where one assessor scores 1 better than 2 and the other scores 2 better than 1.

The result of this analysis is given in Figure 9. Overall, assessor agreement is satisfactory, with assessors agreeing in almost two thirds of cases. The number of contradictions is somewhat high at 10 percent, but it is to be expected in natural language processing evaluation tasks of this kind.

Overall Responsiveness		
Agreements	525	65%
Disagreements	210	26%
Contradictions	75	9%
Linguistic quality		
Agreements	511	63%
Disagreements	214	26%
Contradictions	85	10%

**Figure 9**

Agreements, disagreements and contradictions between assessor scores in the manual evaluation of overall responsiveness and linguistic quality.

## 5.2 Content: the modified Pyramid Score

As is usually done in the latest summarization evaluation campaigns, a Pyramid content score is also computed, according to the methodology outlined by (Nenkova et al. 2007). Because this evaluation is re-using the data set from the TAC 2011 evaluation campaign, the pyramids are readily available. Semantic Content Units (SCUs) were manually assigned to the ABSUM and hybrid summaries from those identified in the four human-written summaries from each cluster and normalized the scores according to the “modified Pyramid” methodology (without jack-knifing). The resulting score can be interpreted as a percentage, where 0 means that the summary is information-less and 1 means that the summary contains the expected amount of information that would be found in a human-written summary. In Figure 7, human-written summaries have no pyramid score because they are themselves the reference.

The most striking result of the modified Pyramid evaluation is that the hybrid summaries have a much higher average than CLASSY’s purely extractive ones. This difference is also statistically significant according to a Wilcoxon Signed Rank test ( $p$ -value  $< 0.05$ ). This means that **on average, ABSUM increases the amount of informative content of a state of the art system**. CLASSY was the chosen system to compare against, because it has the best overall responsiveness in the evaluation campaign, but it did not have the highest pyramid score at TAC 2011. The system (described in (Mason and Charniak 2011)) with the highest average Pyramid score on the 18 clusters of the test set had an average of 0.591, which is still inferior to the hybrid system’s 0.600, though not significantly so.

## 5.3 Content Density

To account for variable summary lengths, the Pyramid metric was developed with the intent that the weight of the SCUs in a summary, which provides the Pyramid score, should be normalized by the total number of SCUs in the summary. This was very hard to implement in practice and this is why the modified Pyramid method is more widely used instead. It is only appropriate for summaries of roughly the same size, however, and so can be misleading in this case, where ABSUM outputs much shorter summaries than the other systems it is compared against.

To account for the varying summary sizes, content density is measured, where it is defined as the modified Pyramid of a summary divided by the number of words of the summary. The averages shown in Figure 7 represent an average of this content density score over 18 summaries.

The content density of ABSUM is higher than that of any other automatic system in the evaluation, by a factor of almost 2 to 1. This is also the case when compared to any other

Cluster Topic	Summary generated by ABSUM	C	L	O
1. Amish Shooting	On October 2, 2006, a shooting occurred in Lancaster County. Amish girls were assaulted. A suicidal gunman shot the classroom of girls and himself.	28	5.0	2.3
5. Plane Crash Indonesia	On January 1, 2007, a plane crash occurred in Indonesia. An Adam Air plane crashed. 90 people died.	27	4.0	2.0
8. Cyclone Sidr	On November 15, 2007, a cyclone occurred in Bangladesh. The cyclone caused a storm. The powerful cyclone hit the southwestern coast of Bangladesh. Hundreds died. Homes were destroyed. Thousands were evacuated.	43	3.7	2.7
9. Dimona Attack	On February 4, 2008, a suicide attack occurred in Dimona. An Israeli police officer killed a second suicide bomber. An Israeli woman and three Israelis died. The military wing of the Palestinian Fatah movement claimed responsibility for the attack.	34	4.0	2.3
10. Earthquake Sichuan	On May 12, 2008, an earthquake occurred in Sichuan. The powerful earthquake struck Sichuan province. 8,533 people died.	38	3.3	2.3
11. Finland Shooting	On November 7, 2007, a school shooting occurred in Finland. A teenaged gunman shot eight people and himself. Seven classmates, headmistresses, victims of a high school, five teenage boys, a wounded, 18-year-old pekka-eric Auvinen, two girls and three women died.	53	2.3	2.0
15. Oil Spill South Korea	On December 7, 2007, an accident occurred in South Korea.	6	3.7	1.0
16. VTech Shooting	On April 16, 2007, a campus shooting occurred at Virginia Tech. 32 people were killed.	24	5.0	2.0
22. Minnesota Bridge Collapse	On August 1, 2007, a ground collapse occurred in Minnesota. A bridge crashed. Seven people died. 79 people were injured.	26	3.0	1.7
23. USEmbassy Greece Attack	On January 12, 2007, a rocket attack occurred in Athens. The U.S. Embassy was struck. A third-floor bathroom was damaged. A domestic militant group was suspected of conducting the attack.	34	4.0	3.3
24. Indonesian Mud Volcano	In January 2007, an eruption occurred in East Java. 13 people died. The marine ecosystem was damaged.	15	4.3	2.0
26. Reporter Shoe Bush	In December 2008, a shoe attack occurred in Iraq.	7	3.0	1.0
30. Borneo Ferry Sinking	In January 2007, a ferry accident occurred. 13 survivors of the ferry disaster were rescued.	0	3.3	1.7
33. Glasgow Airport Attack	In June 2007, an airport attack occurred in Glasgow Airport. Two people were arrested. The airport was evacuated.	41	3.7	1.7
37. Crane Collapse	On March 15, 2008, a construction accident occurred. A towering crane crashed. Four people died. 24 people were injured. A five-story building and parts of five buildings were demolished. A building was evacuated.	65	3.3	2.7
39. Pirate Hijack Tanker	On November 15, 2008, Pirate attacks occurred in Bahrain. Merchant vessels, 92 ships and the large crude tanker Sirius Star were attacked.	5	2.7	1.7
40. Bangladesh Flood	On August 1, 2004, a monsoon flood occurred in Bangladesh. 126 people died. Standing crops and 250,000 houses were damaged.	29	3.3	1.7
42. Jaipur Bombs	On May 13, 2008, a terror attack occurred in India. A market was hit. 80 people were killed. Pakistan-based Islamic militants were suspected of conducting the attack.	26	4.3	3.3

**Figure 10**

The summaries generated by ABSUM on the test set, with their scores for Content (Modified Pyramid times 100), Linguistic quality, and Overall responsiveness. The latter two are an average of 3 assessments on a scale from 1 to 5. 18 of the 44 clusters from TAC 2011's official data set were used.

Cluster Topic	Summary generated by the ABSUM/CLASSY hybrid
8. Cyclone Sidr	<i>On November 15, 2007, a cyclone occurred in Bangladesh. The cyclone caused a storm. The powerful cyclone hit the southwestern coast of Bangladesh. Hundreds died. Homes were destroyed. Thousands were evacuated.</i> A fierce cyclone packing extreme winds and torrential rain smashed into Bangladesh's southwestern coast Thursday, wiping out homes and trees in what officials described as the worst storm in years. A powerful cyclone with strong winds started pounding on Bangladesh's south and southwestern coast from Thursday evening. The money will go to German relief organisations working in cooperation with local partners to alleviate suffering caused by Cyclone Sidr.
16. VTech Shooting	<i>On April 16, 2007, a campus shooting occurred at Virginia Tech. 32 people were killed.</i> Thirty-two people were killed and at least 15 injured in two shooting attacks at Virginia Tech on Monday during three hours of horror and chaos on this sprawling southwestern Virginia campus. U.S. President George W. Bush said at the memorial service held at Virginia Tech Tuesday that the shooting rampage marked "a day of sadness" for the entire nation. Virginia Tech President Charles Steger said Tuesday all classes for the rest of the week at the university have been canceled.
37. Crane Collapse	<i>On March 15, 2008, a construction accident occurred. A towering crane crashed. Four people died. 24 people were injured. A five-story building and parts of five buildings were demolished. A building was evacuated.</i> A piece of steel fell and sheared off one of the ties holding it to the building, causing the structure to detach and topple, said Stephen Kaplan, an owner of the Reliance Construction Group. Neighborhood residents said they had complained to the city several times about the construction at the site, saying crews worked illegal hours and the building was going up too fast.

**Figure 11**

Summaries generated by the ABSUM/CLASSY hybrid for 3 clusters of the test set. The first part of each summary, in *italics*, is the output from ABSUM, while the rest is generated by sentence extraction with the CLASSY system.

automatic system in TAC 2011's evaluation campaign. Short summaries have an advantage when it comes to content density, because any system that shortens its output will typically improve its content density, by selecting sentences in which they have higher confidence. However, in ABSUM's case, not only the summaries are shorter, but the sentences are as well, and they tend to contain much less uninformative and useless bits of information than in extracted sentences. This also explains why adding ABSUM's sentences at the beginning of a state of the art system like CLASSY significantly improves its informative content.



## 5.4 Qualitative Observations

Figure 10 shows all the summaries that were automatically generated by ABSUM on the test set. This covers only the clusters with topics from within 2 of the categories, which explains why the topic numbers are not consecutive in the figure.

While the summaries are short, they are often very informative, such as in the case of the summary for “37. Crane Collapse”. This 33 word long summary has a modified Pyramid score of 0.647 whereas CLASSY’s 98 word long summary only has a score of 0.235. At the other end of the spectrum are the summaries for “26. Reporter shoe Bush” and “15. Korea Oil Spill” which provide very little relevant information. These cases highlight the typical problem of favoring precision over recall: spills and the kind of damage that they induce, as well as attacks such as a shoe attack which has no casualty or damage at all, were not observed in the development set nor foreseen, and so there were no rules to deal with them directly in the category’s task blueprint.

The linguistic style of these summaries is by design telegraphic, generating intentionally short, to-the-point sentences to express content efficiently. However, even given that style, a few summaries remain awkward-sounding or even ungrammatical. The worst one in that regard is probably the summary for the topic “11. Finland Shooting”, in which the third sentence includes false, confusing and redundant information, as well as a person’s name which should be capitalized. Another interesting bad case is the summary for the topic “39. Pirate Hijack Tanker”, which is informative but also awkward and misleading. The Sirius Star was attacked on the day mentioned, but the other ships were reportedly attacked earlier in the year and were not the focus of the source documents.

Generally speaking, there are however very few false facts in the summaries. Claiming false information as fact in abstractive summaries is a very serious problem that puts the credibility of the user in the system in jeopardy. This problem is much less present in extractive summaries, where the problem usually occurs only in cases where an anaphoric word gets resolved to the wrong antecedent. As mentioned before, ABSUM addresses this problem by being very strict in what will be allowed to be generated by the content selection process.

As can be seen in Figure 11, the hybrid summaries suffer a lot from redundancy issues, whether the abstractive part was short or long. This is especially apparent in the hybrid summary for the topic “8. Cyclone Sidr” where poor Bangladesh appears to be hit 3 to 4 times by the same cyclone, in a way that sounds extremely redundant.

The linguistic errors discussed for ABSUM apply to the hybrid as well, but not the problem of low coverage, since the extractive part fills in the information gaps, such as in the hybrid summary for the topic “16. VTech Shooting”, where the extracted sentences add extra information, though the summary contains again an obvious repetition at the start of the third sentence.

The third hybrid summary is a good example of both parts doing their jobs well, where the abstractive content covers the most salient information in a very efficient way, and the extracted sentences add details and peripheral information.

## 6. Discussion

### 6.1 Going Back to K-BABS

K-BABS suggests that abstractive summarization can be conducted in two steps: analysis of the text into a source representation, and generation of the summary on the basis of abstraction schemes. In practice, applying this principle requires a methodology for content selection as well, and this is what ABSUM did. Ideally, this extra step should be more conceptual than it currently is, and many tricks and stratagems were required to ensure the quality of the final output. The abstraction schemes were written with no assumption of how to select content for generation,

and in that sense they remain “pure”, unconcerned with how to apply the knowledge it contains in a practical setting. Bringing together the information extracted from the pattern matching rules therefore required a lot of care in the content selection part of the summarization system.

The most limiting factor, however, does not lie in the way the abstraction schemes are written, but rather in the very first step: analysis. Tools for the analysis of text, and especially syntactical parsing, have become sufficiently accurate to make a system such as ABSUM possible, but other tools were not performing as well as would be necessary, especially noun phrase coreference resolution, for which no freely available system seemed to perform even remotely close to the level that would be required. Handling time better could go a long way, as well as dealing with documents or clusters that contain several events of the same type – this has led to several errors by ABSUM in the test set.

As the tools for analyzing syntax, more or less complex semantic information, text structure, coreferences, and so on, perform better, so will systems that implement the K-BABS approach. It will also have the effect of greatly simplifying the programming, since a lot of development time was spent on preprocessing and the analysis module (aiming to reduce parsing errors) for ABSUM, and even much more on content selection heuristics that would balance its lack of noun phrase coreference resolution and the parsing inaccuracies that remained.

## 6.2 Differences between ABSUM and Other Non-Extractive Summarizers

K-BABS is conceptually a more general approach to non-extractive summarization than what has been previously proposed. There is no assumption made about the type of analysis that is performed. The abstraction schemes look for patterns in the source text representation that can be much richer than just taking the output of the analysis module as is. The transformations performed by the abstraction schemes are aimed directly at generating a summary, taking into account the specific summarization task at hand.

Previous systems, notably SUMMONS (Radev and McKeown 1998) and RIPTIDES (White et al. 2001), took the output of a pre-existing information extraction engine, which contained relevant information, and generated sentences to express that information in the form of a summary. Although on the surface, it may seem that the pattern matching rules and generation templates in ABSUM’s abstraction schemes perform similarly, there are some important differences that are worth highlighting.

One of the key differences with previous work is that the slot fills in ABSUM are open-ended. Many aspect candidates are considered for inclusion as an aspect answer, allowing for the aggregation of information over varying sources, in combinations that appear in no unique document. This is made possible because no preexisting information extraction system is used, the task blueprints being written specifically for a pre-existing summarization task.

ABSUM makes direct use of redundancy and other criteria to exclude certain aspect candidates, thus partly solving the problem posed by bad parses and incorrect applications of pattern matching rules. In this regard, the heuristics used for the appropriate selection of content for the summary represent a departure from previous work on non-extractive summarization. In previous work such as SUMMONS (Radev and McKeown 1998) and RIPTIDES (White et al. 2001), there is a strong sense that information extraction slots are assumed to be filled correctly (and if they are not, that is not the summarization system’s fault). SUMMONS even goes as far as to use generation operators to compare and contrast content from different documents, which necessarily assumes accurate template slot fills. Assuming relatively accurate slot fills is easier to do when the summarization task tackled is defined specifically around what is available in the output of an information extraction system, as was done for SUMMONS and RIPTIDES. ABSUM was designed to address a pre-defined summarization task, namely TAC 2011’s Guided Summarization.

ABSUM's focus on precision over recall has led to taking fewer risks overall, making for perhaps a less ambitious system, but we believe that this is why the performance of ABSUM turns out to be so good, even when compared with the state of the art in extractive summarization. No systematic evaluation of SUMMONS was conducted, and RIPTIDES was only compared against a very simple baseline. In both cases, only well-behaved examples were presented, whereas we have shown all of our results from the test set, including a lot of mistakes and an analysis of some of the errors produced by the system.

### 6.3 On the Automation of Abstraction Scheme Acquisition

An obvious development bottleneck in K-BABS has to do with building domain- and task-specific blueprints for the system. It requires considerable manual labor and will never fully cover everything that needs to be said about various aspects of a category or domain. This can be addressed by finding a way to acquire pattern matching rules (and perhaps generation templates as well) automatically, and then possibly reviewing and correcting them manually, where needed. Here are some ways that have been used to extract similar kinds of patterns automatically and that might be adapted to the needs of ABSUM and other systems that might implement K-BABS.

Automatic schemata acquisition was first proposed by (Mooney and DeJong 1985), where consecutive actions performed by an entity are (to some extent) assumed to have a causal relationship or to lead to a common goal. When a set of actions lies outside the known schemata, a new one is created to account for this observation. Somewhat similarly, (Chambers and Jurafsky 2008) and (Chambers and Jurafsky 2009) perform the unsupervised learning of narrative schemas (also referred to as narrative event chains), and this has been applied to creating information extraction templates that extract the same kind of information found by ABSUM's pattern matching rules, in (Chambers and Jurafsky 2011). (Saggion 2013) also produces information extraction templates using unsupervised learning, but without using full parsing or semantic resources. The typical event participants and the event chains or narrative schemas are learned jointly, by relying on observed syntactical relations and coreference chains. This kind of knowledge can be used to generate pattern matching rules and generation templates to populate a task blueprint.

(Filatova et al. 2006) propose a methodology to automatically extract domain patterns from large corpora. In this work, a domain is a type of event, such as earthquakes and presidential elections. The most representative verbs of a domain are identified, and the most frequently observed syntactical patterns in which these verbs occur are considered as information extraction patterns. To apply such a technique in the case of guided summarization, for instance, would require a human to then manually select which template patterns fit with which aspect, and to add a generation template by hand. This would greatly speed up the task blueprint writing process. (Li et al. 2010) have a similar approach, but based on entities rather than verbs, in order to produce the kind of information extraction patterns that would, for example, fill slots for a Wikipedia infobox about a person or organization. This could also be relevant depending on the kind of summaries desired.

Overall automatically acquiring pattern matching rules for abstraction schemes such as the ones used in ABSUM appears feasible, but considerations of keeping a high precision would probably require human intervention as part of a semi-automatic strategy to fill in the task blueprint, and some amount of testing on a development set is required to prevent rules that tend to lead to false positives. There are plans to explore this in future work.

### 6.4 The Hybrid Summarization Setting and a New Task for Evaluation Campaigns

Another way to address the lack of coverage of manually (or automatically) constructed task blueprints is to develop summarization systems that are specifically designed for the purpose of

generating the latter part of a hybrid summary, for which some aspects are known to be poorly covered.

An extractive summarization system designed for hybrid summarization will have to avoid redundancy with content over which it has no control. This is very different from the normal extractive summarization framework and has important consequences on system design. Some extractive summarizers attempt to select all the sentences at the same time, such as approaches that use linear programming (Gillick et al. 2009). In such a context, special constraints need to be added for avoiding redundancy with sentences that are outside the scope of the extractive summary. Also, many approaches use lists of words to favor sentences that answer specific aspects for guided summarization, such as (Conroy et al. 2011), and if one of those keywords is not in the summary but some sentence in the source text contains it, then that sentence is likely to be selected by the algorithm. This does not usually account for lexical variability and the abstract summary may choose one way to express the same thing that the extractive summary will absolutely want to include. To resolve this issue, it would be possible to have the abstract summarizer communicate to the extractive one which aspects of the domain category it believes that it has answered, and which words and phrases are believed to answer those aspects.

The hybrid system could also be built from two different abstractive systems, one implementing K-BABS such as ABSUM, followed by some kind of sentence compression-based approach. The compression could specifically be driven to remove phrases or words that are redundant with the fully abstractive part of the summary. This would be very useful in cases similar to the topic “VTech Shooting” in Figure 11, where removing the phrase `thirty-two people were killed` from the third sentence would greatly improve the hybrid summary, since that phrase is repeated verbatim. Obviously, the second sentence could be removed instead, which is something hybrid summaries with better interaction between the two systems should be able to accomplish.

Building extractive or semi-extractive summarization systems in the hybrid summarization setting requires its own experimentation, and while the type of abstractive summarizer and its quality will affect the task, the basic difficulties should remain the same. A new task for summarization evaluation campaigns could therefore be something along the lines of *summary completion*, in which the first half of the summary is provided, and the second half must be filled in to add more depth and details while avoiding repetitions. This is at the same time analogous to and different from update summarization, where an update summary should avoid repeating information assumed to be already known to the reader, from previously published documents. When elaborating the setting for evaluating summary completion systems, the summaries’ first half could be written by humans, but the goal would be to apply the techniques developed in hybrid summarization.

Another challenge that the ABSUM/CLASSY hybrid does not address is whether using extracted information is always relevant or not. Assuming that the non-extractive summarizer is able to provide more generated sentences than it currently does, perhaps it still remains relevant to include some extracted sentences in the summary, in order to increase the coverage. After all, non-extractive summarizers – or at least those using IE or abstraction schemes – look for specific kinds of content in a top-down way, whereas extractive and semi-extractive summarizers perform a bottom-up discovery of relevant but sometimes unexpected material. To benefit from both requires a delicate balancing that calls for experimentation not yet performed in this work.

## 6.5 Adaptation of ABSUM to Other Tasks and Domains

Any language expert (used in a broad sense) should be able to write task blueprints for any new task or domain of interest, given enough time and the right tools. No programming proficiency is absolutely necessary, since the abstraction schemes are purely declarative (ABSUM’s task

blueprints are written in XML files). Elementary knowledge in syntactical analysis and world knowledge about the new task and domain (and some common sense!) should suffice. At the core of K-BABS is the abstraction scheme, and given the kinds of syntactical and semantic information that are made available by the analysis module, anyone can write abstraction scheme pattern matching rules and generation templates that will be used in a predictable manner by the summarization system. Writing a task blueprint for new categories is thus fairly straightforward, even though some care and a lot of trial and error are required for getting satisfactory results.

For ABSUM, a blueprint was written for one guided summarization category, alongside the development of the summarization system. The many design decisions for the summarization system, including the choice of several heuristics and parameters, are specific to the guided summarization task, the corpus, or more generally to the summarization of news, but not to a given topic category. Only when the system was completed did work begin on the second category. This provides insight into the amount of effort necessary to write a blueprint for a new category, assuming that the context remains the same, namely the guided summarization of news articles.

The time spent to add an additional category can be considerable, and it was estimated to be around 100 man hours, purely for writing the task blueprint. Actually, a lot more time was spent to add functionalities to the summarization system in parallel to writing rules, because it did not support all the features necessary for tackling the new category. Some of those changes to the summarization system also affected processing of the first category as well. As more categories are added, the system will reasonably need fewer and fewer improvements and modifications, and only new task blueprints will need to be written, as is intended. The categories used by TAC are very broad (*ATTACKS* rather than school shootings, terrorist attacks, political assassinations, etc., and *ACCIDENTS AND NATURAL DISASTERS* rather than transportation accidents, earthquakes, tsunamis, etc.). The numerous kinds of events that must be covered in a single category consequently require numerous rules and more effort in constructing flexible word lists. More restricted categories would make task blueprint writing less time consuming and more intuitive.

For other types of input documents and other summarization tasks, some changes are likely necessary, especially with respect to the choices made regarding some of the content selection heuristics. As a simple example, the fact that aspect candidates that appear less than twice are filtered out was a design decision made to compensate for occasional bad parses, and because important information tends to be repeated in the context of multidocument news summarization with 10 source documents. The minimum frequency of 2 used here is a parameter (and there are a few more like it in the content selection module) that can be adjusted according to the needs of the task at hand. Most of the stoplists have to be revised depending on the tasks as well. More complex design decisions can be put into question depending on the needs of the task of interest, such as the choice of accepted noun modifiers, the constraint put on observed subject-verb-object triples, the special processing of words such as *people*, the constraint of giving at most two arguments (subject and object) to verbs during generation, etc.

Dealing with other tasks is already possible within the current implementation, because the generation template does not have to resemble the pattern matching rules, as is currently the case in ABSUM. Indicative summaries can be generated with only slight adjustments to the system. The kind of deep understanding of the documents enabled by the K-BABS approach also enables to compare and contrast different documents with very novel sentences generated for that purpose. This could prove very interesting in a variety of settings, such as multidocument summarization where inconsistencies between sources can be reported concisely, and update summarization where changes in some key elements can be reported explicitly. Multilingual summarization is another interesting angle that should be explored, since the generation

templates of the abstraction schemes can be written in a different language, and only translating short phrases at a time would be necessary.

## 7. Conclusion

This work has demonstrated both the desirability and the feasibility of abstractive summarization. The K-BABS approach shows great promise, given the excellent evaluation results of ABSUM. Performance in the context of hybrid summarization is superior to the previous state of the art, in terms of the amount of content in the summary. It is also the first instance of an abstractive system performing better than state-of-the-art extractive systems on a shared task and according to a state of the art evaluation protocol.

This can probably be attributed to ABSUM's focus on achieving good precision, i.e. the attention that was given to avoid bad summaries arising from incorrect parses and bad applications of pattern matching rules. This empirical consideration which is a departure from previous literature in abstractive summarization leads to very interesting theoretical questions given that it will always be the case that analysis tools will not be perfect, and so there will be a need to perform good abstractive summarization in spite of this challenge of dealing with noisy data. Is there a way to establish a confidence in certain aspect candidates? And what about their realization? What are other possible strategies to select a noun or verb phrase realization for an aspect answer, and which is best?

Even though ABSUM performed well during evaluation, it was not first developed with the goal of outperforming other approaches at all cost, but rather to test hypotheses about natural language understanding and processing, as exposed by the K-BABS approach, which also considers the flexibility and scalability of the approach as a necessary criterion. After all, the goal of summarization research should be, on one hand, to achieve good performance at the task of summarizing text today; and on the other hand, to develop and improve techniques and theories on language understanding and semantic analysis which will help address a variety of natural language processing tasks tomorrow. Whereas lately, it seems that researchers in text summarization have focused on the first goal at the detriment of the second, ABSUM reached the first goal by aiming at the second.

## Acknowledgement

This work greatly benefited from the help of Dr. John Conroy and Dr. Judith Schlesinger who adapted the system CLASSY for hybrid summarization, ran it on the test set, and made their results available so that the hybrid summaries could be evaluated.

Thanks are also extended to the volunteers who performed the evaluation. Without their contribution, manual evaluation of ABSUM would not have been possible.

This work was funded in part by the Fonds de recherche du Québec – Nature et technologies (FRQNT).

## References

- Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.
- Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June. Association for Computational Linguistics.
- Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 602–610, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *J. Artif. Int. Res.*, 34(1):637–674.
- John M. Conroy, Judith D. Schlesinger, Jeff Kubina, Peter A. Rankel, and Dianne P. O'Leary. 2011. CLASSY 2011 at tac: Guided and multi-lingual summaries and evaluation metrics. In *Proceedings of the Fourth Text Analysis Conference*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology.
- 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.
- Hal Daumé, III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 449–456, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*. The Stanford Natural Language Processing Group.
- Gerald DeJong, 1982. *An Overview of the FRUMP System*, pages 149–176. Lawrence Erlbaum.
- Brigitte Endres-Niggemeyer. 1998. *Summarizing Information: Cognitive Strategies*. Springer-Verlag Beling Heidelberg.
- Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 207–214, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 177–185, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Albert Gatt and Ehud Reiter. 2009. Simplenlg: a realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Pierre-Etienne Genest and Guy Lapalme. 2011a. 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, USA, June. Association for Computational Linguistics.
- Pierre-Etienne Genest and Guy Lapalme. 2011b. Generated abstracts for TAC 2011. In *Proceedings of the Fourth Text Analysis Conference*, Gaithersburg, Maryland, USA, November. National Institute of Standards and Technology.
- 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.
- David Gillick, Benoit Favre, Dilek-Hakkani Tür, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD Summarization System at TAC 2009. In *Proceedings of the Second Text Analysis*

- Conference, Gaithersburg, Maryland, USA. National Institute of Standards and Technology.
- Udo Hahn and Ulrich Reimer. 1999. Knowledge-based text summarization: Saliency and generalization operators for knowledge base abstraction. *Advances in Automatic Text Summarization*, pages 215–232.
- Laura Hasler. 2007. From extracts to abstracts: Human summary production operations for computer-aided summarization. In *Proceedings of the RANLP 2007 workshop on computer-aided language processing*, pages 11–18.
- Min-Yen Kan, Kathleen R. McKeown, and Judith L. Klavans. 2001. Applying natural language generation to indicative summarization. In *Proceedings of the 8th European workshop on Natural Language Generation - Volume 8, EWNLG '01*, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.
- 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.
- Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, HLT-Short '08*, pages 193–196, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Peng Li, Jing Jiang, and Yinglin Wang. 2010. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10*, pages 640–649, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop: Text Summarization Branches Out*, pages 74–81.
- Elena Lloret and Manuel Palomar. 2012. Text summarisation in progress: a literature review. *Artif. Intell. Rev.*, 37(1):1–41, January.
- Rebecca Mason and Eugene Charniak. 2011. Extractive multi-document summaries should explicitly not contain document-specific content. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages, WASDGLM '11*, pages 49–54, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1, IJCAI'85*, pages 681–687, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.
- Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4, May.
- Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the TAC 2011 summarization track: Guided task and aesop task. In *Proceedings of the Fourth Text Analysis Conference*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology. <http://www.nist.gov/tac/publications/>.
- J. J. Pollock and A. Zamora. 1975. Automatic abstracting research at chemical abstracts service. *Journal of Chemical Information and Computer Sciences*, 15(4):226–232.
- Dragomir R. Radev and Kathleen R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500.
- 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.
- Peter Rankel, John M. Conroy, Eric V. Slud, and Dianne P. O'Leary. 2011. Ranking human and machine summarization systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 467–473, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Horacio Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with sumum. *Comput. Linguist.*, 28(4):497–526, December.
- Horacio Saggion. 2013. Unsupervised learning summarization templates from concise summaries. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, USA, June. Association for Computational Linguistics.
- Karen Spärck Jones. 1999. Automatic summarising: Factors and directions. *Advances in Automatic Text Summarization*, pages 1–12.



- Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Inf. Process. Manage.*, 43(6):1449–1481, November.
- Hideki Tanaka, Akinori Kinoshita, Takeshi Kobayakawa, Tadashi Kumano, and Naoto Kato. 2009. Syntax-driven sentence revision for broadcast news summarization. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, UCNLG+Sum '09, pages 39–47, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multidocument summarization via information extraction. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.
- David M. Zajic, Bonnie J. Dorr, and Jimmy Lin. 2008. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4):1600 – 1610.
- Renxian Zhang, Wenjie Li, Dehong Gao, and You Ouyang. 2013. Automatic Twitter Topic Summarization With Speech Acts. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(3):649–658.

