Using a Text Engineering Framework to Build an Extendable and Portable IE-based Summarisation System

Diana Maynard, Kalina Bontcheva, Horacio Saggion, Hamish Cunningham, Oana Hamza
Dept of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK

Paper ID:

Keywords: summarisation, Information Extraction, GATE, evaluation

Contact Author: Diana Maynard (diana@dcs.shef.ac.uk)

Under consideration for other conferences (specify)?

Abstract

In this paper we show how tools provided by a text engineering framework (GATE) have been used to build an IE-based summarisation system in the domain of occupational health and safety. The core of the application is based on pattern-action grammar rules, which can easily be extended or ported to new domains. The GATE framework was also used to evaluate automatically the system's performance.

Using a Text Engineering Framework to Build an Extendable and Portable IE-based Summarisation System

Paper ID:

Abstract

In this paper we show how tools provided by a text engineering framework (GATE) have been used to build an IE-based summarisation system in the domain of occupational health and safety. The core of the application is based on pattern-action grammar rules, which can easily be extended or ported to new domains. The GATE framework was also used to evaluate automatically the system's performance.

1 Introduction

According to (Mani, 2000), the goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application needs. In this paper we present an IE-based summarisation system (HaSIE), which aims at producing a summary from annual company reports about the companies' performance on Health and Safety issues. The extracted summaries allow the automated production of statistical metrics describing the level of compliance with Health and Safety recommendations and any relevant legislation that may be implemented.

The system detects whether or not the text contains information about health and safety and if so, it extracts relevant passages and conceptual information such as awards and number of employees. In this particular application it is vital that we detect correctly whether the report contains health and safety information, because this is one of the main user requirements. The reason why a given report might not contain such information is because the annual company reports contain primarily financial, managerial and performance information.

This summarisation work has been carried out using the GATE architecture. It offers support for the majority of tasks typically performed as part of building a new NLP application, e.g. module development and testing, corpus annotation, and performance evaluation. The framework also has a thorough Unicode support, which enables the development of applications and resources in multiple languages (see (Tablan et al., 2002)).

The rest of the paper is structured as follows. Section 2 briefly outlines the GATE architecture and associated tools and resources used to develop the HaSIE system described here. and Section 3 describes the information extraction modules used. Section 4 describes in more detail the process of extracting information and creating the summaries. Next we discuss evaluation in Section 5 and give some preliminary results for our system. Finally, Section 6 discusses how the HaSIE system is related to other approaches to summarisation.

2 GATE in Brief

GATE (Cunningham, 2002) is an architecture, a framework and a development environment for human language technology modules and applications. During the design phase, the **architectural elements** guide and structure the overall shape of the system. The **framework** helps during the development phase by providing ready made implementations for parts of the architecture. It is open to new types of data, processing resources or visual components, which can be easily added and integrated into new or existing systems. Finally the **development environment** facilitates exploitation of the framework, by aiding overall development

and providing a debugging mechanism for new modules. Because GATE is a component-based model, this allows for easy coupling and decoupling of the processors, thereby facilitating comparison of alternative configurations of the system or different versions of the same module. The availability of tools for easy visualisation of data at each point during the development process aids immediate interpretation of the results.

A set of reusable modules is provided with the backplane, which are able to perform basic language processing tasks such as POS tagging and semantic tagging. These eliminate the need for users to keep reinventing the same resources, and provide a good starting point for new applications. We used these components as a basis for building the HaSIE summarisation system described below (see Section 3).

GATE distinguishes between data, algorithms, and ways of visualising them. In other words, GATE components are one of three types:

- **LanguageResources** (LRs) represent entities such as lexicons, corpora or ontologies;
- **ProcessingResources** (PRs) represent entities that are primarily algorithmic, such as parsers, generators or n-gram modellers;
- **VisualResources** (VRs) represent visualisation and editing components that participate in GUIs.

These resources can be local to the user's machine or remote (available via HTTP), and all can be extended by users without modification to GATE itself. In order to abstract the language processing from the actual format of the data, GATE supports a variety of input and output formats including XML, RTF, HTML, SGML, email and plain text.

2.1 Finite-state transduction support in GATE

In order to make it easier to build new processing resources, GATE comes with built-in finite-state transduction capabilities, which we have used as the core of the summarisation process (see Section 4). The transducer runs on grammars written in the JAPE (Java Annotations Pattern Engine) language (Cunningham et al., 2002), which describes patterns to match and annotations to be created as a result. JAPE is a version of CPSL (Common Pattern Specification Language) (Appelt, 1996), which provides finite state transduction over annotations based on regular expressions. A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules, and which run sequentially. Patterns can be specified by describing a specific text string, or existing annotations (e.g. annotations created by the tokeniser, gazetteer, part-of-speech tagger, or document format analysis). Rule prioritisation (if activated) prevents multiple assignment of annotations to the same text string.

Creating new modules and applications on the basis of JAPE, such as a summarisation component for another domain, is a low-overhead task, because the user only needs to be concerned with writing new grammar rules (though other modules such as the gazetteer may also need to be updated or modified). The amount of tuning necessary for a new domain can vary depending on the type of information that needs to be extracted, and how similar the domain and text structure is to that which existing components are designed for. For example, we reuse much of the same information about companies, dates, numbers, etc from the default Information Extraction components, so very little tuning was needed in this case, especially since the patterns we aim to identify for health and safety are quite easy to define, but this cannot be guaranteed for all other domains or applications.

So far, we have successfully used JAPE for named entity recognition, sentence splitting, and summarisation, and we intend to experiment with it in other fields such as shallow syntactic parsing. Although at the moment we are using hand-crafted rules, it would be possible for an application to learn rules automatically for the Jape transducers in a manner similar to

(Day et al., 1997). These rules could then be verified or amended by a human if necessary, as they are human readable.

2.2 Evaluation tools

A vital part of any language processing application is the evaluation of its performance, and a development environment for this purpose would not be complete without some mechanisms for its measurement in a large number of test cases. GATE contains two such mechanisms: an evaluation tool (AnnotationDiff) which enables automated performance measurement and visualisation of the results, and a benchmarking tool, which enables the tracking of a system's progress and regression testing.

Gate's AnnotationDiff tool enables two sets of annotations on a document to be compared, in order to either compare a system-annotated text with a reference (hand-annotated) text, or to compare the output of two different versions of the system (or two different systems). For each annotation type, figures are generated for precision, recall, F-measure and false positives.

The AnnotationDiff viewer displays the two sets of annotations, marked with different colours (similar to 'visual diff' implementations such as in the MKS Toolkit or TkDiff). Annotations in the key set have two possible colours depending on their state: white for annotations which have a compatible (or partially compatible) annotation in the response set, and orange for annotations which are missing in the response set. Annotations in the response set have three possible colours: green if they are compatible with the key annotation, blue if they are partially compatible, and red if they are spurious. In the viewer, two annotations will be positioned on the same row if they are coextensive, and on different rows if not.

GATE's benchmarking tool differs from the AnnotationDiff in that it enables evaluation to be carried out over a whole corpus rather than a single document. It also enables tracking of the system's performance over time. Furthermore, the system can be run in verbose mode, where for each performance figure below a certain threshold (set by the user), the non-coextensive annotations (and their corresponding text) will be displayed. This information is useful e.g., when developing new JAPE grammar rules to cover cases currently missed by the system.

3 GATE and the HaSIE system

HaSIE uses a set of Named Entity Recognition modules adapted from the ANNIE information extraction system, which comprises a set of IE components included in GATE. HaSIE uses the following processing resources:

- Tokeniser - which splits the text into individual word, number and punctuation tokens.
- Sentence Splitter - which splits the text into individual sentences.
- Part-of-Speech Tagger (Hepple, 2000) - which produces a part-of-speech tag on each token.
- Gazetteer - which contains lists of proper names and keywords used by the grammar.
- Semantic Tagger - a JAPE Transducer which annotates text with information such as entity types.

The first three components are taken directly from ANNIE; the gazetteer and JAPE transducer are modified versions of ANNIE components, which reflect the specific information needs of the project. The HaSIE gazetteer contains additional information about accidents, and words related specifically to the field of health and safety. The HaSIE semantic tagger contains hand-coded rules to identify textual patterns which are to be annotated, for example, to find noun phrases which contain information about accidents.

4 Extracting the relevant information

The grammar (semantic tagger) rules and gazetteer lists used in HaSIE are determined by corpus analysis, and are hand coded. Although

this seems like an onerous task, the whole system was adapted from ANNIE, the core IE system, in a very short space of time with minimum effort. This is partly because the GATE infrastructure and modular design makes it easy to adapt the existing resources, and partly because a very few rules suffice to cover a wide range of cases.

4.1 Extracting key words and phrases

The first step in the procedure is to extract specific words and phrases related to health and safety (e.g. "SHE", "Occupational Health") accidents (e.g. "accident", "injury", "death"), dates, company names, etc. Some of these, such as the health and safety and accident annotations, are used in later phases of the grammar (see Section 4.2); others, such as the name of the Company, are output directly. Figure 1 shows part of a sample text (the Debenhams company report downloaded from the Internet) annotated with such entities.

The annotation types extracted in this phase are: Company, Organization, Jobtitle[1], Accident, HSE, Date, Percent, Money and Number. With the exception of Company, Accident and HSE, these types (and the rules used to extract them) are the same as those used in the default ANNIE system. Although we do not use all of these types for the final output, they are used as temporary annotations to help us with the summarisation process, as described in the next section.

The following rule shows how we extract an HSE annotation, using the result of gazetteer lookup and part-of-speech tagging. We look for a Jobtitle annotation (found in a previous phase of the grammar), followed by one or more numbers, common nouns or adjectives (in any combination) followed by something which has been found in a gazetteer list of HSE keywords. If this pattern is matched, then the whole pattern is annotated as an HSE.

```
Rule: HSE2
(
 ({Jobtitle}|
  {Token.category == CD}|
  {Token.category == NN}|
  {Token.category == JJ})+
 ({Lookup.majorType == hse})+
):key
-->
 :key.TempHSE = {rule= HSE2}
```

4.2 Combining information

The second step is to use a JAPE grammar to define rules which extract sentences or paragraphs which describe health and safety issues (depending on the issue). The grammar depends on previous annotations such as the result of sentence splitting, tokenisation, and the initial annotations produced. We use paragraphs to describe more general issues, such as parts of the document relating generally to health and safety, and sentences to describe more specific information, such as accident and incident rates.

4.2.1 Paragraph Annotations

In order to generate paragraphs about HSE, we use a grammar phase consisting of hand-coded rules. Each rule is assigned a priority, such that the highest priority rule is matched first (if possible). If this rule is not fired, the rule with the next highest priority is matched, and so on. The priorities are determined according to how specific the rule is. The rule which is most specific is allocated the highest priority, because it is more likely to generate a correct result than a general rule, which is less precise. We give examples below of 3 rules used in the grammar phase which generates the paragraphs about HSE.

The first rule tries to match paragraphs containing an HSE annotation (produced in Step 1), followed by one or more Accident annotations, Environment annotations, or HSE Keys (possible separated by other words). An HSE Key is a word which in itself does not represent something related to health and safety, but when combined with an HSE annotation suggests more strongly the presence of something related to health and safety. For example "re-

[1]We identify this because we want to know whether there is somebody in the company whose role is to look after health and safety matters

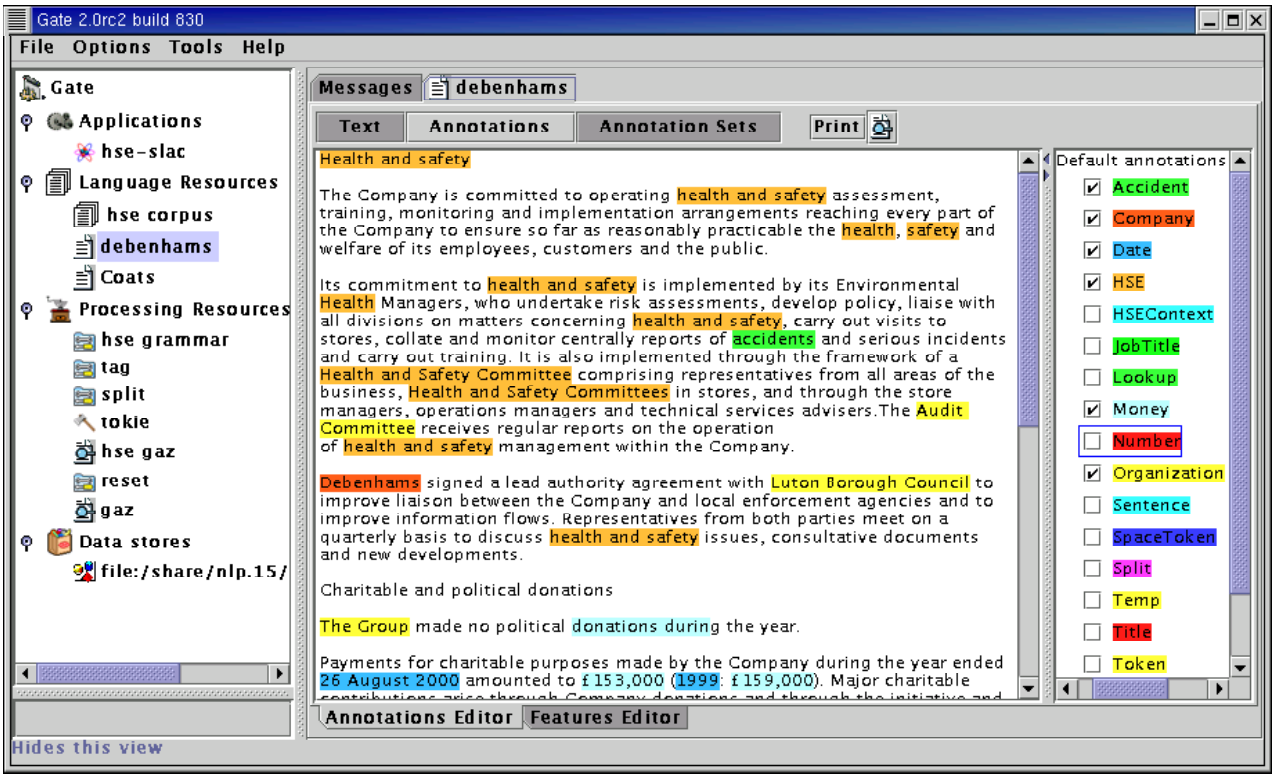

Figure 1: Debenhams report annotated with entities

port" on its own does not provide conclusive evidence, but when found with "health" provides a much stronger clue. "Environment" annotations on their own again do not provide much evidence, but when combined with an HSE annotation are again much more strongly linked with health and safety. If the rule finds a successful match, the whole paragraph is annotated.

The second rule tries to match paragraphs containing an HSE annotation, followed by one or more other HSE annotations or Organization annotations (possibly separated by other words). An HSE annotation refers to a word of phrase directly related to health and safety, an Organization annotation refers to a company name or name of another organization such as "Health and Safety Executive" or "EC". If the rule finds a successful match, the whole paragraph is annotated.

The third rule looks for an HSE Key followed by an HSE annotation (possibly separated by other words). If the rule finds a successful match, the whole paragraph is annotated. Since this is quite a general rule, it only gets fired if both the other rules fail to match.

Some paragraphs produced are quite general, giving non-specific information about company policies on health and safety, e.g.

```
''It is the policy of the Group
that each business maintains the high
standards necessary to safeguard the
health, safety and welfare of their
employees, customers and the general
public.''
```

Other paragraphs may give more explicit details, e.g.

```
''BAA has received a RoSPA gold
award for occupational safety for
the fourth year running.  The award
is given only if a consistently good
or continuously improving performance
can be demonstrated over a four-year
period.''
```

Figure 2 shows an example of the Debenhams report shows earlier, this time with the health and safety paragraphs annotated.

4.2.2 Sentence Annotations

Sentences about more specific facts are also identified, in particular relating to accidents and illnesses, e.g.

```
‘‘The accident frequency ratio for
construction projects was 0.4 (0.49)
per 100,000 hours worked, less than
a third of the national accident
frequency rate in the construction
sector.’’
```

These are identified using rules based on entities such as percentages, accident entities, and numbers.

We currently use two rules to identify accident statistics. The first rule tries to match a Number annotation (written in figures) followed immediately by an Accident annotation. The second rule is more complicated, and tries to match an Accident annotation, followed immediately by one or more Number annotations (in figures) or Percentage annotations, followed optionally by further Accident, number or percentage annotations. Number and Percentage annotations are produced during earlier phases of the grammar, and are based largely on gazetteer lookup and string identification (e.g. looking for the “%” sign). If this pattern is matched, the whole sentence is again identified.

4.3 Populating a database

Finally, the relevant annotated parts of the document are output by the system in a comma separated file format, which is then imported in an Access database. Finally, the relevant annotated parts of the document are output by the system in a structured file format, which is then imported in an Access database. The users inspect the summarisation results using a form interface (see Figure 3) and also SQL queries (e.g., to find out how many companies do not discuss health and safety issues in their annual reports). Prior to using the IE-based summarisation system, such analysis was performed manually, which involved locating the necessary information in several hundred reports, each between 50 and 100 pages long.

5 Evaluation

Evaluations of text summarization systems can be intrinsic or extrinsic (Sparck Jones and Galliers, 1995): intrinsic evaluation measures the content of the summary by a comparison with an “ideal” or “target” summary. Extrinsic evaluation measures how helpful summaries are in the completion of a given task, for example in question answering or text categorization. When the evaluation is done by comparing extracted textual units (sentences or paragraphs) to a set of ideal textual units, then coselection is measured by precision, recall and F-score (Firmin and Chrzanowski, 1999). While many researchers have resisted these measures because in generic summarization it is recognized that there is no “ideal summary” (Jing et al., 1998), in our domain-dependent IE-based summarization, there is a clear idea of what information should be included in the summary and which articles deal with health and safety information.

First, we carried out formative evaluation, involving input from our users, as part of the system development cycle. We processed 36 company reports (8.9 MB of HTML text) with HaSIE and asked one of our users to mark up in the resulting summaries (97.6 KB) the sentences which they found highly relevant, completely irrelevant, and acceptable. The results showed that the system had identified correctly that 8 of the reports did not contain relevant information. From the remaining 28 summaries, only 10 had some irrelevant sentences, but these sentences never constituted more than 50% of the summary. These results led to changing HaSIE to account better for ambiguous keywords, such as “health”, because often they occur in irrelevant terms like “health care” and “health insurance”. After these changes were implemented, 7 of the 10 problematic summaries no longer contained the irrelevant senteces.

After improving the system to deal with key-

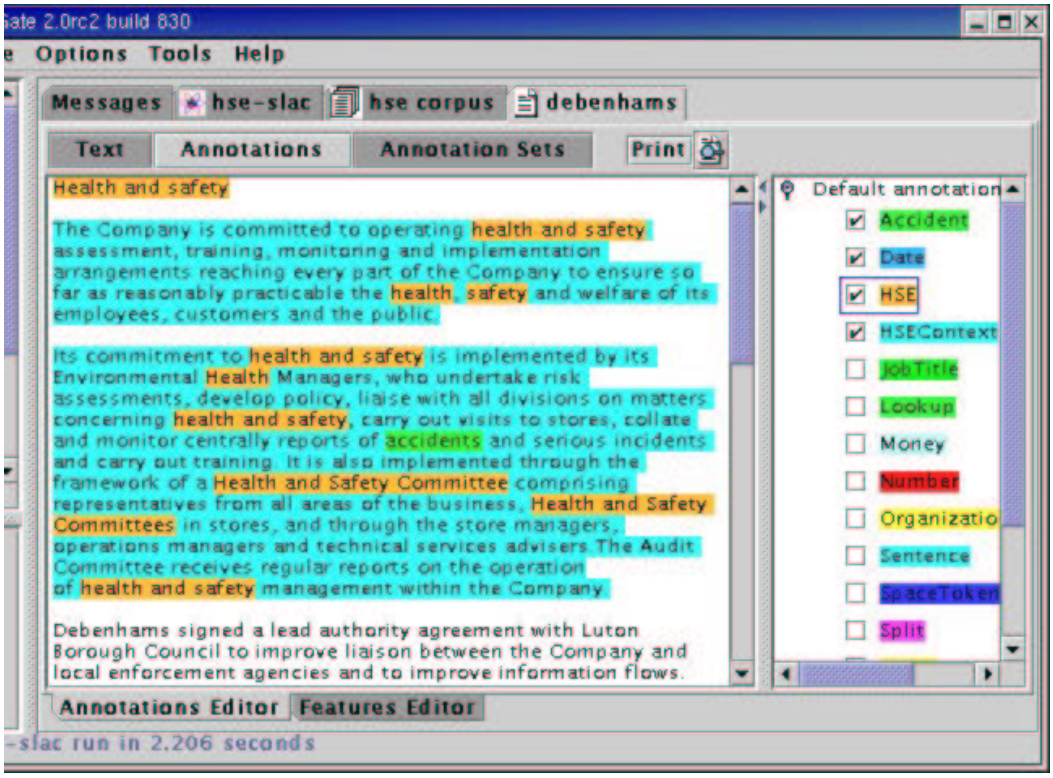

Figure 2: Debenhams report annotated with health and safety paragraphs

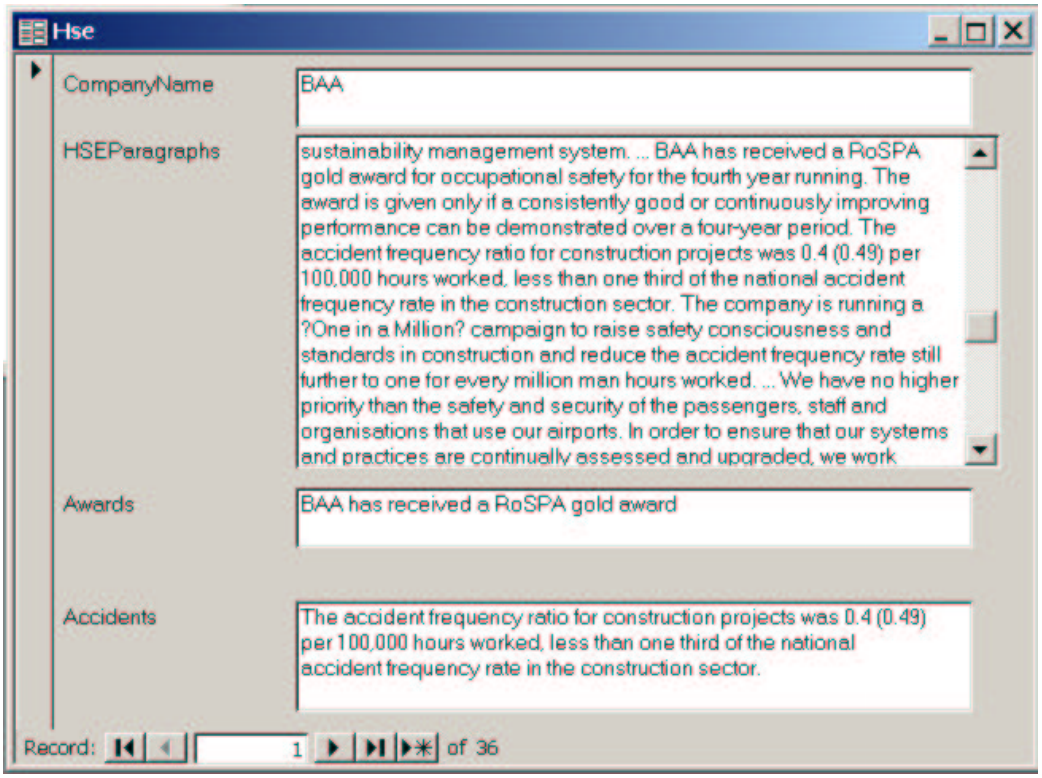

Figure 3: The Access form used for report summary analysis

word ambiguities, we are now in the process of creating a manually annotated HSE summary corpus for 80 of the company reports. To this end, we use HaSIE to bootstrap the annotation process and then correct its results manually, using the visual document annotation editor from GATE. This corpus will be used to measure the system's performance, using GATE's evaluation tools discussed above. Initial results, based on ten of these documents, show that the system achieves 80% precision and 83% recall, which is consistent with the performance estimates we obtained from the formative evaluation, after we discarded the irrelevant sentences which were no longer included after the ambiguity improvements.

6 Related Work

Unlike other knowledge-based approaches to summarization that depend on rich conceptual structures (Hahn, 1990; Rau et al., 1989), we rely on domain-specific terminology and robust Information Extraction techniques. Closely related to our approach is Concept Based Abstracting (CBA) (Paice and Oakes, 1999) where shallow processing is used to instantiate a semantic frame containing the most important concepts of the source document. Unlike CBA, which produces a short textual abstract, our system produces a set of text passages and instantiates a number of slots (e.g. company, number of employees, etc.) which are used by an information analyst to produce health and safety reports. Our rules for passage extraction rely on concept co-occurrence, and so are similar to the contextual templates employed in other IE approaches to summarization (Paice and Jones, 1993; Saggion and Lapalme, 2000).

7 Conclusion and Future Work

In this paper we have presented how GATE was used as a development environment for a summarization system. Our focus was on the adaptation of general natural language engineering tools to an specific summarization problem. While we have relied completely on IE for the purpose of this application, GATE also allows the programmer to easily deploy methods within the framework for computing surface level indicators of salience. In fact, several such GATE-based modules (e.g., sentence position, $tf * idf$) are currently being developed by one of the authors.

References

D.E. Appelt. 1996. The Common Pattern Specification Language. Technical report, SRI International, Artificial Intelligence Center.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, and C. Ursu. 2002. *The GATE User Guide*. `http://gate.ac.uk/`.

H. Cunningham. 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254.

D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain. 1997. Mixed-Initiative Development of Language Processing Systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*.

T. Firmin and M.J. Chrzanowski. 1999. An Evaluation of Automatic Text Summarization Systems. In I. Mani and M.T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 325–336.

Udo Hahn. 1990. Topic Parsing: Accounting for Text Macro Structures in Full-Text Analysis. *Information Processing & Management*, 26(1):135–170.

Mark Hepple. 2000. Independence and commitment: Assumptions for rapid training and execution of rule-based POS taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October.

Hongyan Jing, Kathleen McKeown, Regina Barzilay, and Michael Elhadad. 1998. Summarization Evaluation Methods: Experiments and Analysis. In *Intelligent Text Summarization. Papers from the 1998 AAAI Spring Symposium. Technical Report SS-98-06*, pages 60–68, Standford (CA), USA, March 23-25. The AAAI Press.

Inderjeet Mani. 2000. *Automatic Text Summarization*. John Benjamins Publishing Company.

Chris D. Paice and Paul A. Jones. 1993. The Identification of Important Concepts in Highly Structured Technical Papers. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proc. of the 16th ACM-SIGIR Conference*, pages 69–78.

Chris D. Paice and Michael P. Oakes. 1999. A Concept-Based Method for Automatic Abstracting. Technical Report Research Report 27, Library and Information Commission.

Lisa F. Rau, Paul S. Jacobs, and Uri Zernik. 1989. Information Extraction and Text Summarization using Linguistic Knowledge Acquisition. *Information Processing & Management*, 25(4):419–428.

H. Saggion and G. Lapalme. 2000. Concept Identification and Presentation in the Context of Technical Text Summarization. In *Proceedings of the Workshop on Automatic Summarization. ANLP-NAACL2000*, Seattle, WA, USA, 30 April. Association for Computational Linguistics.

K. Sparck Jones and J.R. Galliers. 1995. *Evaluating Natural Language Processing Systems: An Analysis and Review*. Number 1083 in Lecture Notes in Artificial Intelligence. Springer.

V. Tablan, C. Ursu, K. Bontcheva, H. Cunningham, D. Maynard, O. Hamza, Tony McEnery, Paul Baker, and Mark Leisher. 2002. A unicode-based environment for creation and use of language resources. In *Proceedings of 3rd Language Resources and Evaluation Conference*. forthcoming.