Deliverable 6.1 Infrastructure for Extractive Summarization

7 Enero 2014

1 Introduction

Due to the overabundance of textual information on-line, automatic text summarization [Saggion and Poibeau, 2013], the reduction of a text to its essential content, is fundamental for information systems dealing with textual content. In recent years text summarization research has intensified with well known evaluation programmes promoting the interest in the area. Comparison of different summarization approaches, which is usually done by relying on well stablished or widely used or accesible systems such as MEAD [Radev et al., 2004], is of paramount importance in text summarization. However, MEAD only provides few summarization functionalities or features such as a position-based feature, a controid-based feature, and a first-sentence similarity feature which could be limited for comparison purposes. Availability of customizable natural language processing systems, and not only ready-made applications, are very important. In this deliverable we describe the infrastructure for text summarization we are relying on in SKATER. It will allow us to create different summarization applications to be used in SKATER's demonstrators.

1.1 The SUMMA System

The SUMMA toolkit [Saggion, 2008] is a library which can be used to implement different summarization solutions, making it ideal for the creation of baseline and advanced summarization systems. It is based on the GATE system [Maynard et al., 2002] and it can be used in the GATE graphical user interface or as a java library making it suitable for the creation of standalone or web applications. In order to carry out extractive text summarization in SKATER we rely on two components for basic linguistic analysis. On the one hand we can use the components available through the GATE system [Maynard et al., 2002] which we use mainly for processing English texts, on the other hand we make use of the FreeLing system [Padró and Stanilovsky, 2012] mainly for processing Spanish, Catalan, and other languages targeted in SKATER. When using FreeLing we produce for the input text either an XML file which will be compatible with GATE or we call the FreeLing analyser and read and store the linguistic information into the document we are processing.

The different language functionalities are represented at Figure 1 as different modules and are implemented following the service oriented architecture (SOA). Therefore all the pipelines (one for each language) have been implemented as

web services and may be requested to produce different levels of analysis (e.g. tokenization, lemmatization, NERC, parsing, relation extraction, etc.).

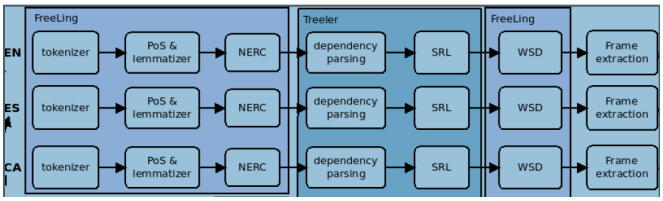

Figure 1: Language Processing Architecture

For **Dependency Parsing**, we use Treeler[1], a library that implements several methods, among other statistical methods for tagging and parsing based on [Carreras,],[Koo et al., 2008], [Carreras et al., 2008].

As with syntactic parsing, **Semantic Role Labeling** methods are developed with the Treeler library. In order to train models, the treebanks made available by the CoNLL-2009 shared task are used. The method implemented follows a pipeline architecture described in [Lluís et al., 2013].

The Freeling includes a **Word Sense Disambiguation** engine based in WN synsets, UKB (AgirreSoroa09).

The final step allows to convert all the gathered linguistic information into a semantic representation. Our method is based on the notion of frame: **a semantic frame** is a schematic representation of a situation involving various participants. In a frame, each participant plays a role. There is a direct correspondence between roles in a frame and semantic roles; namely, frames correspond to predicates, and participants correspond to the arguments of the predicate. We distinguish three types of participants: entities, words, and frames. For example, in the sentence in Figure 2, we can find three frames:

```
 1 Acme        acme        NP00000 B-PER  8 SBJ    _          _       A1     A0     A0
 2 ,           ,           Fc      O      1 P      _          _       _      _      _
 3 based       base        VBN     O      1 APPO   00636888-v base.01 _      _      _
 4 in          in          IN      O      3 LOC    _          _       AM-LOC _      _
 5 New_York    new_york    NP00G00 B-LOC  4 PMOD   09119277-n _       _      _      _
 6 ,           ,           Fc      O      1 P      _          _       _      _      _
 7 now         now         RB      O      8 TMP    09119277-n _       _      AM-TMP _
 8 plans       plan        VBZ     O      0 ROOT   00704690-v plan.01 _      _      _
 9 to          to          TO      O      8 OPRD   _          _       _      A1     _
10 make        make        VB      O      9 IM     01617192-v make.01 _      _      _
11 computer    computer    NN      O     10 OBJ    03082979-n _       _      _      A1
12 and         and         CC      O     11 COORD  _          _       _      _      _
13 electronic  electronic  JJ      O     14 NMOD   02718497-a _       _      _      _
14 products    product     NNS     O     12 CONJ   04007894-n _       _      _      _
15 .           .           Fp      O      8 P      _          _       _      _      _
```

Figure 2: Output of the analyzers for the sentence Acme, based in New York, now plans to make computer and electronic products.

- Base: A person or organization being established or grounded somewhere. This frame has two participants: Acme, a participant of type entity playing the theme role (the thing being based), and New York, a participant of type entity playing the role of location.
- Plan: A person or organization planning some activity. This frame has three participants: Acme, a participant of type entity playing the agent

[1] http://treeler.lsi.upc.edu

role, now, a participant of type word playing the role of time, and make, a participant of type frame playing the theme role (i.e. the activity being planned).

- Make: A person or organization creating or producing something. Participants in this frame are: Acme, entity playing the agent role, and products, a participant of type word playing the theme role (i.e. the thing being created).

It is important to note that frames are a more general representation than SVO-triples. While SVO-triples represent a binary relation between two participants, frames can represent any n-ary relation. For example, the frame for plan is a ternary relation because it includes a temporal modifier. It is also important to note that frames can naturally represent higher-order relations: the theme of the frame plan is itself a frame, namely make. A graphical representation of the example sentence is presented in Figure 3

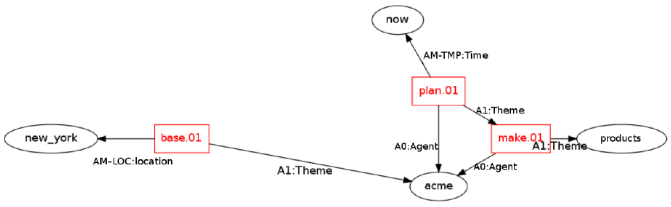

Figure 3: Graphical representation of frames in the example sentence.

The SUMMA system contains a series of processing resources – implemented algorithms – to compute among other things sentence relevance features. It also contains a number of language resources – data – to be used by a number of components.

1.2 Processing Resources

There are over twenty processing resources in SUMMA from which we choose a set of most representative ones to describe in this deliverable. The SUMMA Web site describe all resources[2]. The resources are used to: (i) carry out textual analysis from the linguistic information provided by different text processors, and to (ii) compute features for sentence relevance assessment. A number of classical features for computing sentence relevance described in the literature [Mani, 2001] have been implemented in SUMMA. Both single and multi-document summarization can be produced with the components described here.

1.3 Statistical Computation

The **SUMMA NEs Statistics** module is used to produce term frequency values for each word (or expression) in the document. It additionally uses a table of inverted document frequencies to produce tf*idf values for each word in the document. The idf table is a language resource instantiated from a

[2] `http://www.taln.upf.edu/pages/summa.upf/`

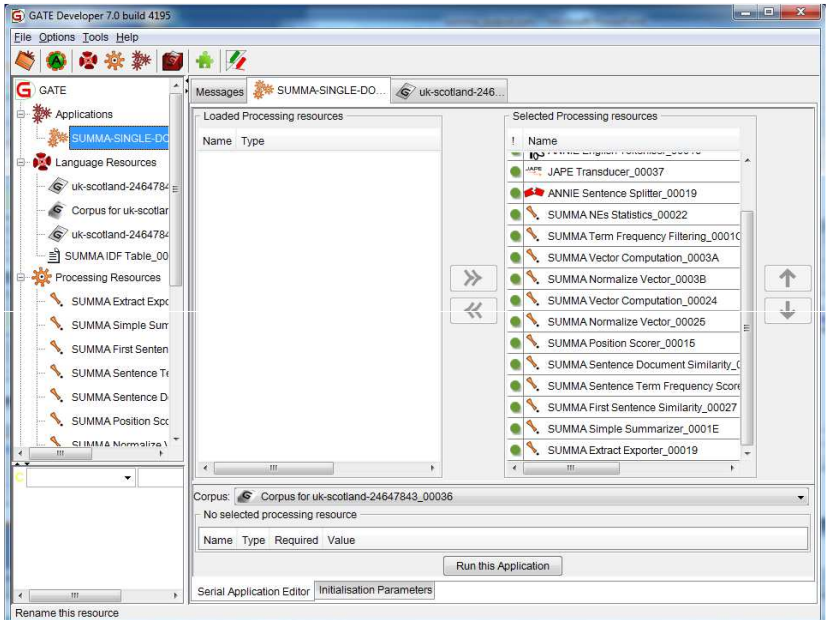

Figure 4: Summarization Application in the GATE GUI

pre-computed table or a table created on-line from a corpus. The statistics produced for each word or expression can be filtered out (nullified) using the componente **SUMMA Term Frequency Filtering**. The combination of tf*idf computation and filtering can effectively used to implement a keyword method [Luhn, 1957].

1.4 Vector Representation and Similarity Computation

Several sentence relevance features can be produced by comparing the content of sentences with other document or external units (e.g. query, title, centroid). The module **SUMMA Vector Computation** creates vector representations for textual units to support cosine-similarity computation. The dimentions of the vectors are the sentence words (or terms) and the values of the dimentions statistics such as tf*idf. The **SUMMA Title Sentence Similarity** is a flexible module which can be used to compare two document units (e.g., a sentence and the title of the document) and produce a similarity value for each sentence in the document. The comparison is implemented as the cosine of the angle between the two vectors being compared. In addition to vectors, the **SUMMA N-Gram Computation** can be used to compute customized n-grams for evaluation purposes (SUMMA has an implementation of the ROUGE metrics [Lin, 2004]) and for detecting n-gram redundancy in a multidocument summarization setting. The **SUMMA Centroid Computation** module is used to create a vector representing the centroid of all document vectors in a corpus.

1.5 Sentence Relevance Scorers

SUMMA scorers implement various traditional summarization features used by extractive systems:

- **SUMMA Position Scorer** scores sentences relying on customized weights for the position of the sentence in the document.
- **SUMMA Paragraph Scorer** scores sentences relying on customized weights for the position of the sentence in the paragraph.
- **SUMMA Cue Phrase Scorer** implements a cue-based relevance feature relying on a pre-computed list of cue-phrases and weigths
- **SUMMA Sentence Document Similarity** is used to compute the relevance of a sentence with respect to the full document.
- **SUMMA Query Method Scorer** implements a relevance metric based on the similarity of a sentence to a user query. The query itself is a document which contains a vector to be compared to.
- **SUMMA Term Frequency Scorer** implements a relevance scoring function based on term distribution. It uses the statistics computed by the statistcal computation module.
- **SUMMA Semantic Scorer** computes a relevance feature for a sentence which is based on the distribution of named entities it contanis. The named entities to be considered can be customized by the user.

These components produce features (with numerical values) that can be combined to produce a sentence scoring function implemented by a dedicated component – the **SUMMA Simple Summarizer**. Sentences' scores are used as the basis for ranking and selecting content units for the summary. Figure 4 shows the components in a summarization application.

The component **SUMMA Simple Multi-document Summarizer** can be used for extracting relevant sentences from multiple "related" documents filtering out redundant information. A multidocument relevance feature based on centroid computation can be produced using the **SUMMA Centroid Sentence Computation**.

1.6 Language Resources

SUMMA implements language resources needed by several of its components. For example in order to perform statistical analysis, SUMMA relies on inverse document frequency tables [Salton and McGill, 1983] which can be created from external resources or directly with a SUMMA component. Tables for stop words and categories are also implemented to support term filtering functionalities. In order to implement our cue phrase component a "cue" word lists is also provided through a Gazetteer lists implementation available with GATE.

2 Creating Summarization Applications

Summarization applications can be created as GATE pipelines combining processors for text analysis and components for sentence scoring and summarization. The applications can be saved and used in any environment which contains the libraries and resources used to create the application.

Summarization applications to support single document summarization of news articles in Spanish and Catalan have been developed and deployed in a server. The interface can be seen at `http://summaweb.upf.edu/`. This application can create stand alone summaries or sentence highlights. This application is one of the prototypical examples which we will follow to demonstrate SKATER summarization functionalities.

In addition to the basic building blocks to create NLP pipelines, two ready made applications implemented in *Bash* script files are made available for single and multi-document summarization. These applications can be easily customized by setting up a fixed number of parameters in the script files. Example input files are also provided to test the applications.

3 Installation of the Summarization Infrastructure

The SUMMA software is distributed through a dedicated Web site at `http://www.taln.upf.edu/pages/summa.upf/`. The site includes all documentation on processing resources required to use the software as well as Java examples of how to create standalone applications. The instalation procedure is very simple, it only requires the user to copy the distribution directory to disk. To run the summarization infrastructure within GATE the software can be loaded as a plug-in. To run the infrastructure in a standalone application the GATE and SUMMA libraries must be included in a Java application.

4 Evaluation Components

SUMMA implements the ROUGE evaluation framework based on n-gram comparison. The module **SUMMA Rouge Evaluation** takes a document and a set of reference summaries and produces ROUGE values. Both the system summary to evaluate and the ideal summaries must have the appropriate n-grams computed before ROUGE can be applied. SUMMA also implements summary comparison using the Bleu MT evaluation [Pastra and Saggion, 2003] and cosine-similarity [Donaway et al., 2000].

5 Related Work

Research on text summarization has progressed steadly in recent years, however, and contrary to what occurs with NLP tools such as parsers, named entity recognizers, POS taggers, etc. which can be obtained from well known packages

such as GATE, OpenNLP, etc., not many laboratories make their summarization tools freely available. The MEAD system [Radev et al., 2004] is a well known example of a tool which has been widely used for comparison purposes, however it has fewer components and customization possibilities than SUMMA. Compendium [Lloret, 2012] is a text summarization architecture which includes components for relevance computation and redundancy removal, however appart from an on-line demo, it is not freely available.

6 Conclusions

This report described the summarization infrastructure for extractive summarization to be used in SKATER. The summarization components can be applied to the output of FreeLing making it appropriate for integration in SKATER and for the development of different summarization solutions for the languages addressed by the project. The infrastructure is easily extensible to accomodate new relevance computation features including those relying on deeper linguistic processing (e.g. semantic interpretation, word sense disambiguation).

References

[Carreras,] Carreras, X. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

[Carreras et al., 2008] Carreras, X., Collins, M., and Koo, T. (2008). Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16, Manchester, England. Coling 2008 Organizing Committee.

[Donaway et al., 2000] Donaway, R. L., Drummey, K. W., and Mather, L. A. (2000). A comparison of rankings produced by summarization evaluation measures. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, pages 69–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Koo et al., 2008] Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio. Association for Computational Linguistics.

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Lloret, 2012] Lloret, E. (2012). *Text Summarisation based on Human Language Technologies and its Applications*. PhD thesis, Universidad de Alicante, Spain.

[Lluís et al., 2013] Lluís, X., Carreras, X., and Márquez, L. (2013). Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):219,230.

[Luhn, 1957] Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1:309–317.

[Mani, 2001] Mani, I. (2001). *Automatic Summarization*. John Benjamins Publishing Company.

[Maynard et al., 2002] Maynard, D., Tablan, V., Cunningham, H., Ursu, C., Saggion, H., Bontcheva, K., and Wilks, Y. (2002). Architectural Elements of Language Engineering Robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*, 8(2/3):257–274.

[Padró and Stanilovsky, 2012] Padró, L. and Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. In *Language Resources and Evaluation Conference (LREC 2012)*. ELRA.

[Pastra and Saggion, 2003] Pastra, K. and Saggion, H. (2003). Colouring summaries bleu. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: Are Evaluation Methods, Metrics and Resources Reusable?*, pages 35–42, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Radev et al., 2004] Radev, D. R., Allison, T., Blair-Goldensohn, S., Blitzer, J., Çelebi, A., Dimitrov, S., Drábek, E., Hakim, A., Lam, W., Liu, D., Otterbacher, J., Qi, H., Saggion, H., Teufel, S., Topper, M., Winkel, A., and Zhang, Z. (2004). Mead - a platform for multidocument multilingual text summarization. In *LREC*.

[Saggion, 2008] Saggion, H. (2008). SUMMA: A Robust and Adaptable Summarization Tool. *Traitement Automatique des Langues*, 49(2):103–125.

[Saggion and Poibeau, 2013] Saggion, H. and Poibeau, T. (2013). Automatic text summarization: Past, present, and future. In Poibeau, T., Saggion, H., Piskorski, J., and Yangarber, R., editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing. Springer.

[Salton and McGill, 1983] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.