Scalable Understanding of Multilingual MediA (SUMMA)

http://www.summa-project.eu

H2020 Research and Innovation Action
Number: 688139
D5.2 – Release of storyline level, frame-semantic and AMR parsing capabilities

Nature	Other	**Work Package**	WP5
Due Date	31/07/2017	**Submission Date**	31/07/2017
Main authors	Shay Cohen (UEDIN)		
Co-authors	Sebastião Miranda (PRIB), Shashi Narayan (UEDIN)		
Reviewers	Afonso Mendes (PRIB)		
Keywords	Semantic parsing, AMR parsing, Summarization		
Version Control			
v0.1	**Status**	Draft	20/07/2017
v0.2	**Status**	Reviewed	27/07/2017
v1.0	**Status**	Final	31/07/2017

Contents

Abstract

Semantic parsing enables rich processing of natural language into structures representing the meaning of the text, enabling many other natural language understanding tasks, such as automatic text summarization. In the context of the SUMMA project, several semantic parsing approaches are being researched and developed by leveraging different semantic formalisms (such as Abstract Meaning Representation (AMR) and PropBank semantic role labeling) for multiple languages. In this deliverable we report the progress made in developing these semantic parsing and summarization components from a software tool release perspective and how these tools integrate in the SUMMA platform.

1 Introduction

In the context of the SUMMA project, several NLP modules are being developed which benefit from a structured representation of natural language text. Semantic parsers enable this rich processing of text by extracting a semantic structure representing the meaning of natural language, typically a graph. In the context of the SUMMA project, several semantic parsing components are being developed for multiple languages and in different semantic formalisms, such as PropBank (Kingsbury et al., 2002) and Abstract Meaning Representation (AMR) (Banarescu et al., 2013).

The main goal of the current report is to describe these components from a software tool release perspective, whereas a full research and development overview is to be reported in Deliverable D5.1. Additionally, we also report the software release of summarization-related components, which are the main use-case of the semantic parsing capabilities. Briefly, the software components released in this report are:

- AMREager, a multilingual AMR parser developed by UEDIN.
- TurboParser, a PropBank semantic role labeling parser maintained by Priberam.
- CAMR wrapper, an English AMR parser maintained by LETA.
- AMR-to-text, a module which does text generation from AMR graphs developed by LETA.
- Extractive neural summarization module developed by UEDIN.
- Multi-document extractive summarization module developed by Priberam.

The semantic parsing modules do not have a direct output to the SUMMA platform interface. Instead, the semantic structures computed by these parsers serve as features for other NLP components, such as summarization. To ease integration, some of these modules have already been shared within SUMMA as docker components (www.docker.com) and integrate with the rest of the platform via a RESTful API. In the following sections, we describe each component in more detail.

Note: This document is not intended as a full technical description of the technologies developed in WP5 – that information is contained in Deliverable D5.1 (M18). The current deliverable details only the aforementioned semantic parsing and summarization components of WP5. This deliverable is related to Deliverable D8.3, which lists components for IP purposes at a coarser granularity.

2 Software release by component

2.1 TurboParser

TurboParser (Das et al., 2014; Martins and Almeida, 2014) is a semantic parser actively being developed and maintained at Priberam, which employs a feature-rich linear model with first and second-order dependencies where decoding is performed in a global manner by solving a linear relaxation with alternating directions dual decomposition AD^3 (Martins et al., 2011). The semantic graphs extracted by this component have been used in summarization experiments as reported in deliverable D5.1. Currently, this parser is released within SUMMA as a Docker component, featuring semantic parsing models for English and Spanish. During the context of this project, this module received improvements in terms of computational speed and the ability to run in a multiprocessing environment, in order to scale for the SUMMA platform. Currently, the single-core performance is 2702 tokens/sec.

2.2 AMREager

AMREager (Damonte et al., 2017) is an abstract meaning representation parser (AMR) being developed and maintained by the University of Edinburgh (UEDIN). The parser is an incremental left-to-right parser that works by scanning a sentence, and incrementally creating a graph that represents the meaning of the sentence.

The parser was released on github,[1] and crosslingual extensions for it are provided (ongoing work). The crosslingual mechanism permits the rapid development of AMR parsers for any language, with only basic linguistic processing pipeline and parallel corpus needed for that language. The crosslingual parser is described in the preprint by Damonte and Cohen (2017).

The parser is transition-based, which means that it has a controller (based on a neural network) that chooses among a set of actions to apply in the current state of the parser to incrementally build the AMR graph. As opposed to previous AMR parsers, this parser has a linear asymptotic computational complexity. As such, it recovers AMR parses quickly.

We plan to provide the parser as a docker image as well, which will enable a simple deployment of the parser. We note, however, that the github source code requires minimal installation and should work well in any Python-enabled environment.

2.3 Multi-document Extractive Summarization

Extractive summarization consists in extracting the most relevant parts (such as sentences) from an input document (or collection of documents) and organising it into a summary. Within the SUMMA project, several summarization approaches are being actively researched, and some experiments attempt to leverage semantic graphs, such as the ones produced by the other modules mentioned in this release. The summarization system currently deployed in SUMMA and part of this software release is Priberam's multi-document Extractive Summarizer (based on Almeida and Martins (2013)), which selects the most relevant sentences from an input collection of documents. One of the inputs of this module are the multi-document clusters generated by the SUMMA task T3.4 (WP3).

[1] https://github.com/mdtux89/amr-eager

The SUMMA team carefully designed a flexible RESTful API interface and several object models in JSON schema in order to enable fast and scalable development across all SUMMA modules. The end-point is available for usage within SUMMA at 213.63.185.148/StorylineSummarization/v1. This API receives a document (or a collection of documents) and returns a summary of a given length (e.g., 150 words) representing what the system considers the most relevant information within the input collection of texts.

2.4 SideNet: A Neural Document Summarizer

SideNet is a neural extractive document summarization system that constructs the summary of a given document by extracting relevant sentences from the document. It incorporates the use of side information such as the title and the image captions in the document. It has been used in summarization experiments as reported in deliverable D5.1. It is being developed and maintained by the University of Edinburgh. For more details on the system, we refer the reader to Narayan et al. (2017).

The summarizer is written in Python using TensorFlow, an open-source software library for Machine Intelligence. We plan to soon release our summarizer on github.[2] A running demonstration of SideNet using Flask can be found at http://cohort.inf.ed.ac.uk/sidenet.html. While we have not performed yet a full analysis of the performance of SideNet, we can confirm that it analyses a given document in a few milliseconds on a machine with 3.2GHz CPU.

In addition to the github page, we plan to provide the summarizer as a docker image, which will enable its simple deployment.

2.5 AMR-to-text

The purpose of this module is generating text from the semantic graphs produced by the AMR parsing modules, such as the one described earlier. In our work (further detailed in Gruzitis et al. (2017); Gruzitis and Barzdins (2016)), we use Grammatical Framework (GF) (Ranta, 2011) as the intermediate tree representation of a document, in order to later generate text.

A key feature of GF is its wide-coverage resource grammar library (RGL) with a shared abstract syntax. The idea is to transform AMR graphs (story highlights) to GF abstract syntax trees (AST), leaving the linearization of the acquired ASTs to the existing English resource grammar. The linearization of an AST resolves the word order, word forms (syntactic agreement), function words, etc. Since RGL supports many more languages (30+), this approach is relatively easily extensible to multilingual AMR-to-text generation, given a wide-coverage translation lexicon (currently available for 15+ languages).

In addition to GF, the generator uses Tregex and Tsurgeon packages from the Stanford JavaNLP library, supporting tree pattern-matching and transformation operations. The system processes 20 AMR graphs per second (50 milliseconds per graph) on a machine with 2.5GHz CPU (single core). The source code is available at https://github.com/GrammaticalFramework/gf-contrib/tree/master/AMR/AMR-to-text. We plan to provide the generator as a docker image, which will enable its simple deployment.

[2] https://github.com/shashiongithub/sidenet

Name	Copyright	Licence	Docker
English SRL (TurboParser)	André Martins & Priberam	LGPL v3.0	Y
Advanced	URL: https://github.com/andre-martins/TurboParser		
Spanish SRL (TurboParser)	André Martins & Priberam	LGPL v3.0	Y
Advanced	URL: https://github.com/andre-martins/TurboParser		
English AMR (AMREager)	Marco Damonte & Shay B. Cohen	BSD 2-clause	N
Advanced	URL: https://github.com/mdtux89/amr-eager		
Spanish AMR (AMREager)	Marco Damonte & Shay B. Cohen	BSD 2-clause	N
Advanced	URL: https://github.com/mdtux89/amr-eager		
German AMR (AMREager)	Marco Damonte & Shay B. Cohen	BSD 2-clause	N
Advanced	URL: https://github.com/mdtux89/amr-eager		
English AMR (CAMR wrapper)	Didzis Gosko & Guntis Barzdins	GPL v2.0	Y
Advanced	URL: https://github.com/didzis/CAMR/tree/wrapper		
English Neural Summarizer (SideNet)	UEDIN contributors[a]	BSD 2-clause	N
Advanced	URL: https://github.com/shashiongithub/sidenet		
English Extractive Summarizer	Priberam	SUMMA only	N
Advanced			
Spanish Extractive Summarizer	Priberam	SUMMA only	N
Advanced			
Portuguese Extractive Summarizer	Priberam	SUMMA only	N
Advanced			
AMR-to-text generator	LETA contributors[b]	GPL v2 (Tregex)	N
Initial	Github URL[c]	BSD (rest)	

[a] Shashi Narayan & Nikos Papasarantopoulos & Shay B. Cohen
[b] Normunds Gruzitis & Didzis Gosko & Guntis Barzdins
[c] https://github.com/GrammaticalFramework/gf-contrib/tree/master/AMR/AMR-to-text

Table 1: Semantic Parsing and Summarization components.

3 Release summary

To better track down the current status of development, we define the following component categories:

- **Initial:** A working system, but not necessarily expected to work well in SUMMA due to small or mismatched training data.
- **Advanced:** A system that works well, and may also be expected to work well in SUMMA, for instance because it was trained on broadcast data.
- **Tuned:** A system that has been trained on or adapted to SUMMA-specific data.

Our current system implementations are either initial or advanced, currently covering the English, German and Spanish languages. Table 1 lists the tools that have been made available to the other partners either to support the development of other SUMMA technologies or to serve as direct outputs to the platform.

Since we are still in a development phase, more than one different system is available for some tasks and some systems are not yet dockerized. Further in the development of the projects we will gradually integrate more systems as docker components.

4 Future plans

The current software release already offers a reasonable coverage of features to enable development of the SUMMA platform towards it's ultimate goals. In order to approach this objective, we plan to continue improving our current systems in future software releases. To this end, we envisage two classes of future actions:

1. **First priority: Moving more components into the Docker architecture.** Some components are not yet available as Docker modules; this is indicated in the component table above. Release of a component as a Docker module is taken as that component being usable by the wider project.
2. **Second priority: Iterative development of better models.** We'll continue to pursue the research and development of better models for semantic parsing and summarization. For summarization we'll mostly focus on the neural approaches described in deliverable D5.1.

References

Almeida, M. B. and Martins, A. F. (2013). Fast and robust compressive summarization with dual decomposition and multi-task learning. In *ACL (1)*, pages 196–206.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Damonte, M. and Cohen, S. B. (2017). Cross-lingual abstract meaning representation parsing. *arXiv preprint arXiv:1704.04539*.

Damonte, M., Cohen, S. B., and Satta, G. (2017). An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

Das, D., Chen, D., Martins, A. F., Schneider, N., and Smith, N. A. (2014). Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Gruzitis, N. and Barzdins, G. (2016). The role of cnl and amr in scalable abstractive summarization for multilingual media monitoring. In *Controlled Natural Language*, volume 9767. Springer.

Gruzitis, N., Gosko, D., and Barzdins, G. (2017). Rigotrio at semeval-2017 task 9: Combining machine learning and grammar engineering for amr parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 924–928, Vancouver, Canada. Association for Computational Linguistics.

Kingsbury, P., Palmer, M., and Marcus, M. (2002). Adding semantic annotation to the penn treebank. In *Proceedings of HLT-02*, San Diego.

Martins, A. F. and Almeida, M. S. (2014). Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476.

Martins, A. F., Figueiredo, M. A., Aguiar, P. M., Smith, N. A., and Xing, E. P. (2011). An augmented lagrangian approach to constrained map inference. In *International Conference on Machine Learning (ICML'11)*, Bellevue, Washington, USA.

Narayan, S., Papasarantopoulos, N., Lapata, M., and Cohen, S. B. (2017). Neural extractive summarization with side information. *CoRR*, abs/1704.04530.

Ranta, A. (2011). *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

ENDPAGE

SUMMA

H2020-ICT-2015 688139

D5.2 Release of storyline level, frame-semantic and AMR parsing capabilities