The SUMMA Platform: A Scalable Infrastructure for Multi-lingual Multi-media Monitoring

Ulrich Germann
University of Edinburgh
ugermann@ed.ac.uk

Renārs Liepiņš
LETA
renars.liepins@leta.lv

Guntis Barzdins
University of Latvia
guntis.barzdins@lu.lv

Didzis Gosko
LETA
didzis.gosko@leta.lv

Sebastião Miranda
Priberam Labs
Sebastiao.Miranda@priberam.pt

David Nogueira
Priberam Labs
David.Nogueira@priberam.pt

Abstract

The open-source SUMMA Platform is a highly scalable distributed architecture for monitoring a large number of media broadcasts in parallel, with a lag behind actual broadcast time of at most a few minutes.

The Platform offers a fully automated media ingestion pipeline capable of recording live broadcasts, detection and transcription of spoken content, translation of all text (original or transcribed) into English, recognition and linking of Named Entities, topic detection, clustering and cross-lingual multi-document summarization of related media items, and last but not least, extraction and storage of factual claims in these news items. Browser-based graphical user interfaces provide humans with aggregated information as well as structured access to individual news items stored in the Platform's database.

This paper describes the intended use cases and provides an overview over the system's implementation.

1 Introduction

SUMMA ("Scalable Understanding of Multilingual Media") is an EU-funded *Research and Innovation Action* that aims to assemble state-of-the-art NLP technologies into a functional media processing pipeline to support large news organizations in their daily work. The project consortium consists of the University of Edinburgh, the Lativan Information Agency (LETA), Idiap Research Institute, Priberam Labs, Qatar Computing Research Institute, University College London, and Sheffield University as technical partners, and BBC Monitoring and Deutsche Welle as use case partners.

1.1 Use Cases

Three use cases drive the technology integration efforts.

1.1.1 External Media Monitoring

BBC Monitoring[1] is a business unit within the British Broadcasting Corporation (BBC). In operation since 1939 and with a staff of currently ca. 300 regional expert journalists, it provides media monitoring and analysis services to the BBC's own news rooms, the British government, and other subscribers.

Each staff journalist monitors up to four live broadcasts in parallel, plus several other information sources such as social media feeds. Assuming work distributed around the clock in three shifts,[2] BBC Monitoring currently has, on average, the capacity to actively monitor about 400 live broadcasts at any given time. At the same time, it has access to over 1,500 TV stations and ca. 1,350 radio stations world-wide via broadcast reception, as well as a myriad of information sources on the internet. The main limiting factor in maintaining (or even extending) monitoring coverage is the cost and availability of the staff required to do so.

Continuous live monitoring of broadcast channels by humans is not the most effective use of their time. Entertainment programming such as movies and sitcoms, commercial breaks, and repetitions of content (e.g., on 24/7 news channels), for example, are of limited interest to monitoring operations. What is of interest are emerging stories, new developments, and shifts in reporting. By automatically recording, monitoring, and indexing media content from a large number of media streams, and storing it in a database within a very short period of time after it has been published, the Platform allows analysts to focus on media content that is most relevant to their work and alleviates them from the tedious task of just monitoring broadcast streams in search of such relevant content.

1.1.2 Internal Monitoring

Deutsche Welle,[3] headquartered in Germany, is an international public broadcaster covering world-

[1] https://monitoring.bbc.co.uk/

[2] Actual staff allocation and thus the amount of media monitoring may vary over the course of the day.

[3] http://www.dw.com

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics-System Demonstrations, pages 99–104
Melbourne, Australia, July 15 - 20, 2018.

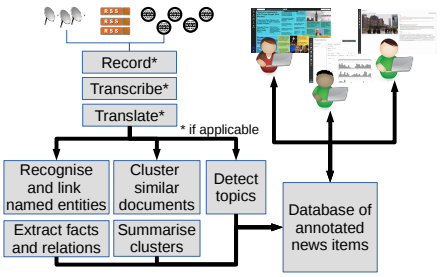

Figure 1: Overview of the SUMMA architecture.

wide news in 30 different languages. Regional news rooms produce and broadcast content independently; journalistic and programming decisions are *not* made by a central authority within Deutsche Welle. This means that it is difficult for the organization at large, from regional managers in charge of several news rooms to top management, to maintain an accurate and up-to-date overview of what is being broadcast, what stories have been covered, and where there is potential for synergies, for example by using journalistic background work that was performed for one story to produce similar content for a different audience in a different broadcast region and language.

The SUMMA Platform's cross-lingual story clustering and summarization module with the corresponding online visualization tool addresses this need. It provides an aggregate view of recent captured broadcasts, with easy access to individual broadcast segments in each cluster.

1.1.3 Data Journalism

The third use case is the use of the platform (or more specifically, it's database API) by journalists and researchers (e.g., political scientists) who want to perform data analyses based on large amounts of news reports. Potential uses are the plotting of events or time series of events on maps, analysis of bias or political slant in news reporting, etc. This use case is the most open-ended and, unlike the other two, has not been a major driving force in the actual design and implementation of the Platform so far.

2 System Architecture

2.1 Design

Figure 1 shows a schematic overview over the Platform's workflow. The Platform comprises three major parts: a data ingestion pipeline build mostly upon existing state-of-the-art NLP technology; a web-based user front-end specifically developed with the intended use cases in mind, and a database at the center that is continuously updated by the data ingestion pipeline and accessed by end users through the web-based GUI, or through a REST API by downstream applications.

Services within the Platform — technical such as the database server or the message queue, and services performing natural language processing tasks — run independently in individual Docker[4] application containers. This encapsulation allows for mostly independent development of the components by the consortium partners responsible for them.

Each NLP service augments incoming media items (represented as `json` structures) with additional information: automatic speech recognition (ASR) services add transcriptions, machine translation (MT) engines add translations, and so forth.

The flow of information within the Platform is realized via a message queues[5]. Each type of task (e.g., speech recognition in a specific language, machine translation from a specific language into English, etc.) has a queue for pending tasks and another queue for completed tasks; task workers pop a message off the input queue, acknowledge it upon successful completion, and push the output of the completed task onto the output queue. A task scheduler (the only component that needs to "know" about the overall structure of the processing pipeline) orchestrates the communication.

The use of a message queues allows for easy scalability of the Platform — if we need more throughput, we add more workers, which all share the same queues.

The central database for orchestration of media item processing is an instance of RethinkDB,[6] a document-oriented database that allows clients to subscribe to a continuous stream of notifications about changes in the database. Each document consists of several fields, such as the URL of the original news item, a transcript for audio sources, or the original text, a translation into English if applicable, entities such as persons, organisations or locations mentioned in the news items, etc.

For the user-facing front end, we use an instance of PostgreSQL[7] that pulls in new arrivals periodically from the RethinkDB. The PostgreSQL database was not part of the original platform design. It was introduced out out of performance concerns, when we noticed at at some point that RethinkDB's reponsiveness was deteriorating rapidly as the database grew. We ultimately determined that this was due to an error in our set-up of the database — we had failed to set up indexing for certain fields in the database, resulting in linear searches through the database that grew longer and longer as the number of items in the database increased. However, we did not realise this until after we had migrated the user front-ends to interact

[4] www.docker.com
[5] www.rabbitmq.com
[6] www.rethinkdb.com
[7] www.postgresql.org

with a PostgreSQL database. The Platform also provides an export mechanism into ElasticSearch[8] databases.

The web-based graphical user front-end was developed specifically for SUMMA, based on wireframe designs from our use case partners. It is implemented in the Aurelia JavaScript Client Framework[9].

In the following section, we briefly present the individual components of the data ingestion pipeline.

3 Data Ingestion Pipeline Components

3.1 Live Stream Recorder and Chunker

The recorder and chunker monitors one or more live streams via their respective URLs. Broadcast signals received via satellite are converted into transport streams suitable for streaming via the internet and provided via a web interface. All data received by the Recorder is recorded to disk und chunked into 5-minutes segments for further processing. Within the Platform infrastructure, the recorder and chunker also serves as the internal video server for recorded transitory material.

Once downloaded and chunked, a document stub with the internal video URL is entered into the database, which then schedules them for downstream processing.

Video and audio files that are not transitory but provided by the original sources in more persistent forms (i.e., served from a permanent location), are currently not recorded but retrieved from the original source when needed.

3.2 Other Data Feeds

Text-based data is retrieved by data feed modules that poll the providing source at regular intervals for new data. The data is downloaded and entered into the database, which then schedules the new arrivals for downstream processing.

In addition to a generic RSS feed monitor, we use custom data monitors that are tailored to specific data sources, e.g. the specific APIs that broadcaster-specific news apps use for updates. The main task of these specialized modules is to map between data fields of the source API's specific (json) response and the data fields used within the Platform.

3.3 Automatic Speech Recognition

The ASR modules within the Platform are built on top of CloudASR (Klejch et al., 2015); the underlying speech recognition models are trained with the Kaldi toolkit (Povey et al., 2011). Punctuation is added using a neural MT engine that was trained to translate from un-punctuated text to punctuation. The training data for the punctuation module is created by stripping punctuation from an existing corpus of news texts. The MT engine used for punctuation insertion uses the same software components as the MT engines used for language translation. Currently, the Platform supports ASR of English, German, Arabic, Russian, Spanish, and Latvian; systems for Farsi, Portuguese and Ukranian are planned.

3.4 Machine Translation

The machine translation engines for language translation use the Marian decoder (Junczys-Dowmunt et al., 2016) for translation, with neural models trained with the Nematus toolkit (Sennrich et al., 2017). In the near future, we plan to switch to Marian entirely for training and translation. Currently supported translation directions are from German, Arabic, Russian, Spanish, and Latvian into English. Systems for translation from Farsi, Portuguese and Ukranian into English are planned.

3.5 Topic Classification

The topic classifier uses a hierarchical attention model for document classification (Yang et al., 2016) trained on nearly 600K manually annotated documents in 8 languages.[10]

3.6 Storyline Clustering and Cluster Summarization

Incoming stories are clustered into storylines with the online clustering algorithm by Aggarwal and Yu (2006). The resulting storylines are summarized with the extractive summarization algorithm by Almeida and Martins (2013).

3.7 Named Entity Recognition and Linking

For Named Entity Recognition, we use TurboEntityRecognizer, a component within TurboParser[11] (Martins et al., 2009). Recognized entities and relations between them (or propositions about them) are linked to a knowledge base of facts using AMR techniques developed by Paikens et al. (2016).

3.8 Trained systems and their performance

Space contraints prevent us from discussing the NLP components in more detail here. Detailed information about the various components can be found in the project deliverables 3.1, 4.1, and 5.1,

[8] www.elastic.co
[9] aurelia.io
[10] The Platform currently is designed to handle 9 languages: English, Arabic, Farsi, German, Latvian, Portuguese, Russian, Spanish, and Ukrainian.
[11] https://github.com/andre-martins/TurboParser

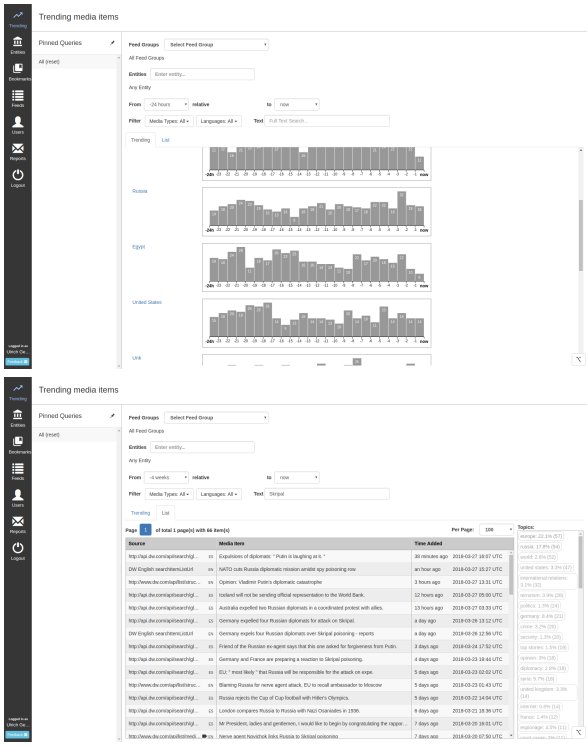

Figure 2: Trending media item views: histogram (top), and list (bottom)

which are available from the SUMMA project's web page.[12]

4 User Interfaces

The platform provides several user interfaces ("views"), to accommodate different user's needs.

The trending media item views (Fig 2) rely on recognised Named Entities. For each entity recognized by the Named Entity recognizer, the histogram view (top) shows a bar chart with the number of media items that mention said entity within each hour for the past n hours. The order of the list of histograms corresponds to the recent prominence of the respective entity. In the list view (bottom), recent arrivals for a specific Named Entity are sorted most-recent-first. The screenshot shows the result of a full text search.

Figure 3 shows the view of a recorded video segment. The video player panel is on the left. On the right, we see topic labels (top), a summary of the transcript (middle), and the original transcript. The transcript is synchronized with the video, so that recognized speech is automatically highlighted in the transcript as the video is played, and a click into the transcript takes the user to the corresponding position in the video.

The document cluster view (Fig. 4) gives a bird eye's view of the clustered media items; the area occupied by each cluster corresponding to the number of items within the cluster. A click on the respective tile shows a multi-document summary of the cluster and a list of related media items.

The trending media item views correspond most to our first use case: journalists keeping track of specific stories linked to specific entities and identifying emerging topics.

The document cluster ciew corresponds best to the needs of our second use case: internal monitoring of a media organisation's output.

In addition to these visual user interfaces, the platform can also expose a database API, so that users wishing to access the database directly with their own analysis or visualisation tools can do so. A number of ideas for such tools were explored recently at BBC's SUMMA-related #NewsHack event in London in November 2017.[13] They include a slackbot that allows journalists to query the database in natural language via a chat interface, automatic generation of short videos capturing and summarizing the news of the days in a series of captioned images, or contrasting the coverage by a specific media outlet against a larger pool of information sources.

5 Hardware Requirements and Performance Analysis

When designing the system, we made a deliberate decision to avoid reliance on GPUs for processing, due to the high cost of the hardware, especially when rented from cloud computing services. This constraint does not apply to the training of the underlying models, which we assume to be done offline. For most components, pre-trained models are available for download.

5.1 Disk Space

A basic installation of the platform with models for 6 languages / 5 translation directions currently requires about 150GB of disk space. Recording of live video streams requires approximately 10–25 GB per stream per day, depending on the resolution of the underlying video. We do not transcode incoming videos to a lower resolution, due to the high processing cost of transcoding.

5.2 CPU Usage and Memory Requirements

Table 1 shows a snapshot of performance monitoring on a singe-host deployment of the Platform that we use for user interface testing and user experience evaluation, measured over a period of about 20 minutes. The deployment's host is a large server

[12] www.summa-project.eu/deliverables

[13] http://bbcnewslabs.co.uk/2017/11/24/summa-roundup. The event attracted 63 registrations; subtracting no-shows, the event had ca. 50 attendees.

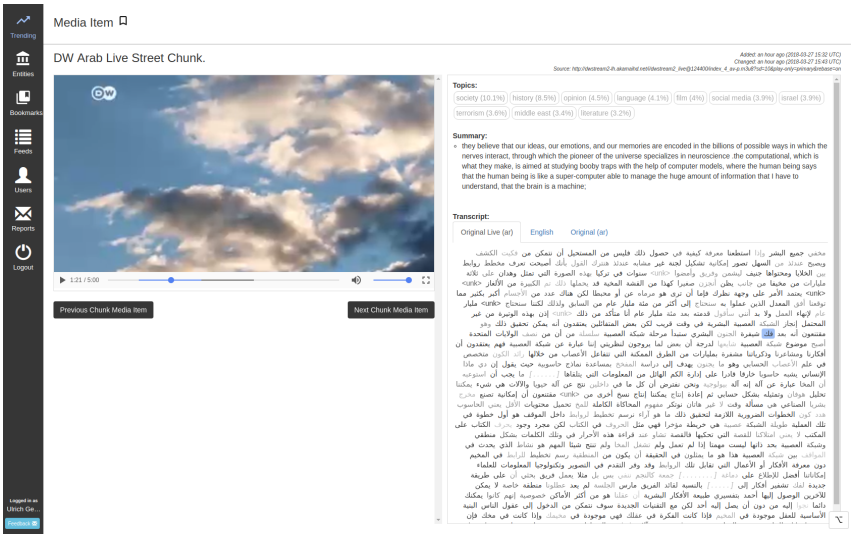

Figure 3: Single media item view

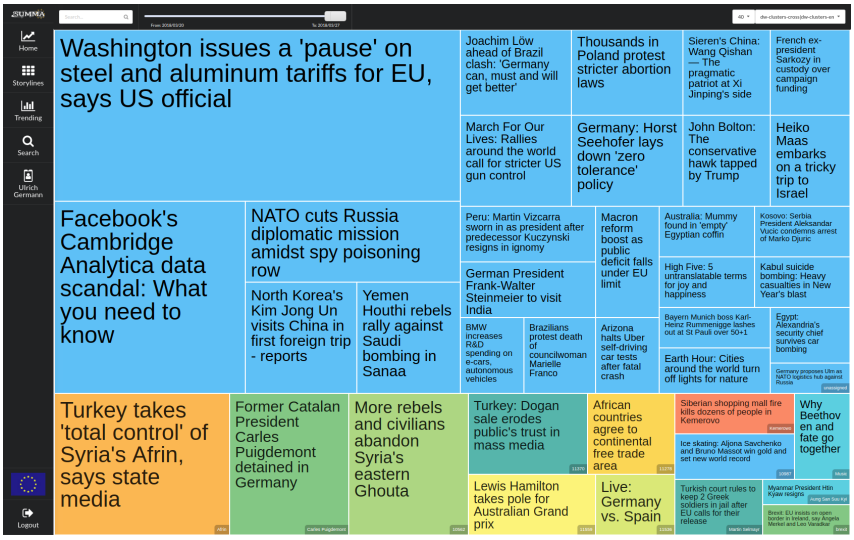

Figure 4: Document cluster view

with 32 CPU cores and 287 GB of memory, of which ca. 120GB are currently used by the Platform, supporting transcription in 6 languages and translation from 5 into English. Not shown in the table are the ASR, Punctuation and MT components for Latvian and German, as they were idle at the time.

Notice that except for the Named Entity recognizer, no component has a peak memory use of more than 4 GB, so that individual components can be run on much smaller machines in a distributed set-up.

Speech recognition is clearly the bottleneck and resource hog in the pipeline. The ASR engine currently is a single-threaded process that can transcribe speech at about half the "natural" speaking rate.[14] Each segment is transcribed in one chunk; we use multiple workers (2-3) per channel to keep up with the speed of data arrival.

The multi-threaded MT engines use all available cores to translate in parallel and are able to translate some 500 words per minute per thread.[15] For comparison: the typical speaking rate of an English-speaking news speaker is about 160 words per minute. A single MT engine can thus easily accommodate transcripts from multiple transcribed live streams.

[14] For segments without speech, the processing speed may be about 8 times real time, as the speech recognizer gets confused and explores a much larger search space.

[15] Junczys-Dowmunt et al. (2016) report 141 words per second on a 16-thread machine; this means ca. 528 words per minute per thread.

Table 1: Relative use of CPU time and peak memory use (per container) for various tasks in the NLP processing pipeline.

Task	CPU	RAM
ASR (ar)	27.26%	1359 MiB
ASR (en)	26.53%	2594 MiB
ASR (ru)	22.40%	2044 MiB
ASR (es)	13.44%	2800 MiB
MT (ar)	3.34%	926 MiB
MT (ru)	3.14%	1242 MiB
Summarization	1.21%	3609 MiB
Semantic parsing	0.97%	3290 MiB
MT (es)	0.34%	1801 MiB
Job queue	0.30%	326 MiB
Topic identification	0.27%	1418 MiB
Named entity recognition	0.18%	12625 MiB
Punctuation (en)	0.16%	362 MiB
Database	0.16%	3021 MiB
Recording and chunking	0.15%	337 MiB
Punctuation (ru)	0.04%	285 MiB
Punctuation (ar)	0.03%	276 MiB
DB REST interface	0.02%	153 MiB
Clustering	0.01%	1262 MiB
Data feed (not streamed)	0.01%	141 MiB
Punctuation (es)	0.01%	240 MiB
Task creation	0.01%	1076 MiB
Result writer (msg. queue to DB)	0.01%	1003 MiB

In a successful scalability test with docker swarm in January 2018 we were able to ingest 400 TV streams simultaneously on Amazon's AWS EC2 service, allocating 2 t2.2xlarge virtual machines (8 CPUs, 32GB RAM) per stream, although 1 machine per stream probably would have be sufficient. For this test, we deployed several clones of the entire platform for data ingestion, all feeding into a single user-facing PostgreSQL database instance.

6 Conclusion

The SUMMA Platform successfully integrates a multitude of state-of-the-art NLP components to provide an integrated platform for multi-lingual media monitoring and analysis. The Platform design provides for easy scalability and facilitates the integration of additional NLP analysis modules that augment existing data.

Availability

The Platform infrastructure and most of its components are currently scheduled to be released as open-source software in late August 2018 and will available through the project home page at http://www.summa-project.eu

Acknowledgements


This work was conducted within the scope of the Research and Innovation Action *SUMMA*, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688139.

References

Aggarwal, Charu C and Philip S Yu. 2006. "A framework for clustering massive text and categorical data streams." *SIAM Int'l. Conf. on Data Mining*, 479–483. Boston, MA, USA.

Almeida, Miguel B and Andre FT Martins. 2013. "Fast and robust compressive summarization with dual decomposition and multi-task learning." *ACL*, 196–206. Sofia, Bulgaria.

Junczys-Dowmunt, Marcin, Tomasz Dwojak, and Hieu Hoang. 2016. "Is neural machine translation ready for deployment? A case study on 30 translation directions." *International Workshop on Spoken Language Translation*. Seattle, WA, USA.

Klejch, Ondřej, Ondřej Plátek, Lukáš Žilka, and Filip Jurčíček. 2015. "CloudASR: platform and service." *Int'l. Conf. on Text, Speech, and Dialogue*, 334–341. Pilsen, Czech Republic.

Martins, André FT, Noah A Smith, and Eric P Xing. 2009. "Concise integer linear programming formulations for dependency parsing." *ACL-IJCNLP*, 342–350. Singapore.

Paikens, Peteris, Guntis Barzdins, Afonso Mendes, Daniel Ferreira, Samuel Broscheit, Mariana S. C. Almeida, Sebastião Miranda, David Nogueira, Pedro Balage, and André F. T. Martins. 2016. "SUMMA at TAC knowledge base population task 2016." *TAC*. Gaithersburg, Maryland, USA.

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. 2011. "The Kaldi speech recognition toolkit." *ASRU*. Waikoloa, HI, USA.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. "Nematus: a toolkit for neural machine translation." *EACL Demonstration Session*. Valencia, Spain.

Yang, Zichao, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. "Hierarchical attention networks for document classification." *NAACL*. San Diego, CA, USA.