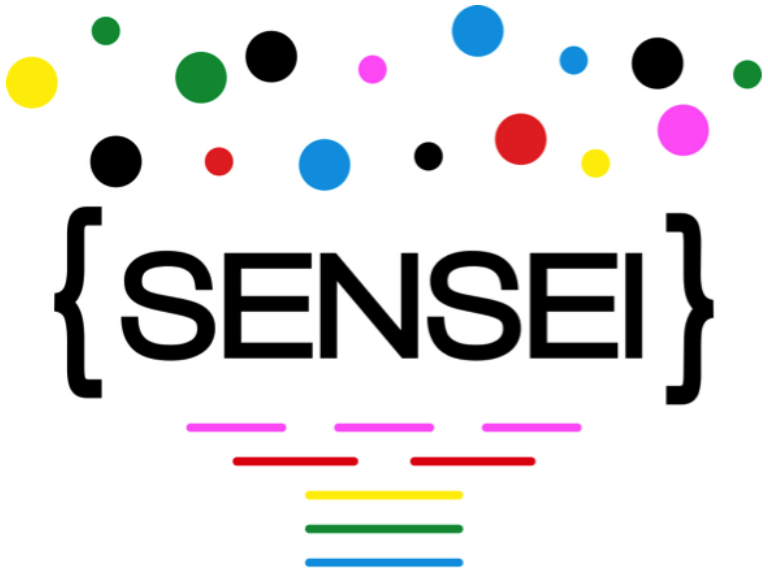


D6.3 – Report on the Rated Questionnaire and Ad-hoc Report Views of the SENSEI Prototype

Document Number	D6.3
Document Title	Report on the Rated Questionnaire and Ad-hoc Report Views of the SENSEI Prototype
Version	1.7
Status	Final
Workpackage	WP6
Deliverable Type	Report
Contractual Date of Delivery	31.10.2016
Actual Date of Delivery	30.10.2016
Responsible Unit	UNITN
Keyword List	Summarization views, user interface, prototype, deployment, software licensing
Dissemination level	PU

Editor

Evgeny A. Stepanov	(University of Trento, UNITN)
Benoit Favre	(Aix Marseille Université, AMU)

Contributors

Evgeny A. Stepanov	(University of Trento, UNITN)
Fabio Celli	(University of Trento, UNITN)
Carmelo Ferrante	(University of Trento, UNITN)
Benoit Favre	(Aix Marseille Université, AMU)
Adam Funk	(University of Sheffield, USFD)
Vincenzo Lanzolla	(Teleperformance, TP)

SENSEI Coordinator

Prof. Giuseppe Riccardi
Department of Information Engineering and Computer Science
University of Trento, Italy
giuseppe.riccardi@unitn.it

Document change record

Version	Date	Status	Author (Unit)	Description
0.1	2016-07-31	Draft	Evgeny A. Stepanov (UNITN)	Initial Outline
0.2	2016-08-26	Draft	Fabio Celli (UNITN)	Brexit Monitoring section
0.3	2016-08-30	Draft	Evgeny A. Stepanov & Carmelo Ferrante (UNITN), Ahmet Aker (USFD)	Added Sections 2.1.1, 3.3
0.4	2016-08-30	Draft	Adam Funk (USFD)	Added Section 3.1, Appendices A and B
0.5	2016-08-30	Draft	Benoit Favre (AMU)	Added Section 5
0.6	2016-09-07	Draft	Vincenzo Lanzolla (TP)	Added Section 3.2
0.7	2016-09-12	Draft	Evgeny A. Stepanov (UNITN)	Update sections
0.8	2016-09-27	Draft	Benoit Favre (AMU)	Add missing sections, cosmetic changes
0.9	2016-09-28	Draft	Evgeny A. Stepanov (UNITN)	Update of some sections
1.0	2016-09-29	Draft	Adam Funk (USFD)	Update to section 3.1
1.1	2016-10-04	Draft	Evgeny A. Stepanov (UNITN)	Added Conclusion
1.2	2016-09-29	Draft	Adam Funk (USFD)	Update to section 3
1.3	2016-10-05	Draft	Benoit Favre (AMU)	Update to section 5
1.4	2016-10-18	Draft	Benoit Favre (AMU)	Update software table
1.5	2016-10-19	Draft	Elisa Chiarani (UNITN) & Massimo Poesio (UESSEX)	Quality check & Scientific review
1.6	2016-10-21	Final	Benoit Favre (AMU) Evgeny A. Stepanov (UNITN)	Corrections
1.7	2016-10-27	Final	Giuseppe Riccardi (UNITN)	Approval for Submission

Executive Summary

This deliverable outlines the activity of the members of the SENSEI project related to the Rated Questionnaire and Ad-hoc Report Views of the prototype, within the context of Work Package 6. The focus of developments reported in this document are twofold: the extrinsic evaluation of summarization technologies in real-world conditions, both in the framework of the speech and social media use cases, and the showcasing of technologies developed in the scope of the project through monitoring of the Brexit referendum. In addition, this report describes the improvements to the core components of the prototype, as well as the distribution of software modules to the community.

Contents

List of Acronyms and Abbreviations

Acronym	Meaning
ACOF	Agent Conversation Observation Form
BART	Beautiful Anaphora Resolution Toolkit
CLI	command-line interface
CRF	Conditional Random Field (a type of machine learning)
CSS	Cascading Style Sheets
DOM	Document Object Model
FBK	Fondazione Bruno Kessler
GATE	General Architecture for Text Engineering
GPL	GNU Public License
GUI	graphical user interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ISF	Interactive Summarization Framework
JAPE	Java Annotation Patterns Engine
JSON	JavaScript object notation
MAE	mean absolute error
ML	machine learning
MMR	maximal marginal relevance
NER	named entity recognition
NLP	natural language processing
PDTB	Penn Discourse Treebank
PHP	PHP Hypertext Preprocessor
PNG	Portable Network Graphics
POS	part of speech
QA	quality assurance
REST	Representational State Transfer
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SQL	Structured Query Language
TBD	to be determined
UI	user interface
URL	Uniform Resource Locator
WADL	Web Application Description Language
WEKA	Waikato Environment for Knowledge Analysis
XML	Extensible Markup Language

1 Introduction

One of the objectives of the SENSEI project is to show that summarization technology created thorough the project is useful in real-life scenarios. To achieve that objective, the project has setup an evaluation framework based on two use cases which aim at an ecological evaluation of the technology, as well as the construction of several artifacts geared towards the community and the general public: a source-code of software modules, and a monitoring and analysis of the Brexit referendum.

The ecological evaluation is based on a protocol created in WP1 and described in deliverables D1.2 through D1.4 for assessing the capability of human subjects to perform a task that matters to them (professionally or personally) with the help of summarization technology. The differential is created by alternatively submitting the subjects to a user interface including summarization technology features, and a baseline interface which replicates currently available features. Then, the performance of the subjects is evaluated in term of indirect factors, such as the time spent to achieve the task, the percentage of tasks completed within a time limit, or direct factors such as a comparison between their creation and a gold standard. In the following, the description of the prototype is separated through the angle of each use case, speech and social media, to focus on the user interaction aspects, although the back-end technology relies on common principles and is already described in deliverables from technical work packages (WP3-5) or from past deliverables of the prototype Work Package (D6.1, D6.2).

In addition to ecological evaluation, SENSEI technology has been exposed to the general public through the creation of a website which monitored the activity of Europeans through their use of social media venues such as Facebook and Twitter. The project has provided technology to capture, analyze and create report based on those sources, and relate the activity to the subject matter of the referendum. The website makes use of mood detection, sentiment analysis, stance detection, syntactic and semantic parsing for generating geographically restricted reports including distribution and evolution in time of the opinions of different populations across Europe.

Finally, following a recommendation from the reviewers of the project and even though it was not part of the original work plan, we have worked towards the release of key components created through the project so that members of the community can replicate some of the experiments performed during the project or build on the technology to create new applications of their own. The software release plan, and its implementation are described in this deliverable.

The design and the results of the ecological evaluation are described in the deliverable D1.4. Here, on the other hand, we describe the prototypes themselves. Section 2 outlines the prototype developments linked to the ecological evaluation. Section 3 describes improvements to core components. Section 4 summarizes the work on the Brexit showcase, and Section 5 focuses on the status of the software release plan.

1.1 Follow-up to Period 2 Activities

In Period 3, we focused on finishing the integration of core components into the prototype framework, and in particular their integration and communication through the conversation repository. We also worked to support the extrinsic evaluation both through the speech and social media use cases. We had to make sure the required core components would feed the prototype with relevant features and analyses, and we made changes to existing UIs or created new ones to support the specifics of the evaluation scenario. In addition, we created the Brexit showcase website, and connected to the other components to setup a near real-time analysis of social-media sources and generate relevant analysis reports. We also spent a substantial effort to stabilize, document and release our software components so that they can be distributed to the community.

2 Prototype Description

This section describes the developments and extensions to the prototype in order to support the extrinsic evaluation of the summarization technologies created by the project. It describes the evaluation scenarios for both use cases and reports some of the implementation details.

2.1 Extrinsic Evaluation Scenarios

Two evaluation scenarios were devised according to the speech and social media use cases. See D1.4 for a complete description and analysis of results.

2.1.1 Speech: Rated Questionnaire Views

The extrinsic evaluation for speech involved quality assurance evaluators from Teleperformance call centers performing tasks relevant to their daily work. The evaluation consisted in two tasks performed on multiple scenarios on both Italian and French data, according to two conditions: using or not using SENSEI-provided technology.

The first task (Task 1) consists in evaluating Agent Conversation Observation Form (ACOF) criteria such as the efficiency of the agent given a conversation recording and automatic transcripts. A set of 10 conversations is presented one by one to the evaluator who has to fill a form corresponding to her analysis. The baseline condition (C1) only gives access to the audio and transcript while the SENSEI condition (C2) also adds synopses and high-level features predicted by the system which could inform the evaluator for her task. A time restriction is enforced so that evaluators have to make use of the features provided by the UI and cannot just listen to the conversation.

The second task (Task 2) is an information retrieval task in which evaluators have access to a whole corpus of conversations through a search engine, and have to determine the proportion of conversations which have a given problem, or list the range of problems addressed in the conversations. In the baseline condition (C1), the evaluators can query the search engine with keywords, which shows the list of conversations that contain the keywords ordered by relevance. The evaluator can chose a conversation by reading snippets which contextualize the search terms in the ASR transcript, and clicking it displays the audio player and the transcript. In the SENSEI condition (C2), the same functionality is available, but evaluators can, in addition, filter the search results with different criteria predicted by our systems, and search results are accompanied by synopses and the same features as for Task 1. Again, the task is timed to ensure that the evaluators do not just listen to all conversations. In this task, the evaluators have to describe their findings own words in a dedicated form.

DECODA For the DECODA corpus, we generated abstractive synopses with the method described in [4] which consists in combining templates extracted from existing synopses and rewriting them to address the specifics of the conversation. When the abstractive synopsis generator was not confident enough, the system backed-off to an extractive system based on Maximal Marginal Relevance (MMR) in order to prevent problems with "good looking" synopses which do not correspond to the topic of the conversation.

The features provided for the UI are as follows:

- Topic: the topic of the conversation.
- Temper/mood: whether the conversation is cold, normal or hot (in which participants shout at each other).
- Politeness: whether or not the participants use inappropriate words or act impolitely by interrupting the other speaker.
- Polarity: whether words expressing negative or positive polarity are used in the conversation.
- Agent polarity: the same but only focused on the agent.
- ACOF: whether the system estimated that the agent would pass all the ACOF questions, or fail on any of them.
- Percentages: those parameters estimated as percentage of the conversation.

The categorical values associated with each features could be used for filtering the search results in Task 2. All the features were predicted by a system described in D3.3 and formatted as JSON for inclusion in the prototype. In the evaluation, ASR transcript with a word error rate between 20% and 30%, provided by LIUM, was used for transcripts, synopsis generation and feature extraction.

LUNA For the LUNA corpus, we first generated abstractive synopses, as described in [3]. However, preliminary assessment revealed that they were hardly readable. Thus, for the extrinsic evaluation extractive synopses were generated from Automatic Speech Recognition output using MMR. Additionally, to filter out turn segments not relevant to the task, the conversations were automatically segmented and labeled for Dialog Act dimensions as described in the deliverable D4.3 on discourse tools.

Similar to DECODA, there are several other conversation descriptions that were provided for the extrinsic evaluation. These include:

- Dialog Polarity: overall sentiment polarity score for the dialog (nominal and as percentage).

- Agent Polarity: sentiment polarity score for the agent turns (nominal and as percentage).
- Caller Polarity: sentiment polarity score for the caller turns (nominal and as percentage).
- Agent Empathy: automatic score to indicate the empathy of the agent, computed from per-turn outputs of the emotion recognition.
- Client Satisfaction: score indicating whether client is satisfied (also from emotion recognition)
- Call Status: (resolved or forwarded) whether the agent solves the problem during the call or forwards it to other department (roughly corresponds to the topic of the call in DECODA).

2.1.2 Social Media: Ad-hoc Report Views

As described in the deliverable D1.4, the evaluation task for the social media scenario is a reading comprehension with respect to a news article and its comments. Participants had to make sense of the comments and to identify the main topics of discussion to be able to answer the questions. The participants are either native or close-to-native English speakers with experience of working as media professionals or members of the public with an interest in online news and/or reading comments.

The evaluation involved three systems, and each system provides a user with a set of comments. The baseline system (A) uses a typical comment presentation view. The systems B and C make use of the features developed within SENSEI project and presents summarization views in the form of clustered comments and pie charts. The extrinsic evaluation for social media was done remotely, through a web platform. For the evaluation, the participants had to perform 2 tasks: using the baseline system and one of the SENSEI enable systems. The detailed description of each system is available in the deliverable D1.4.

The SENSEI enabled system B makes use of a set of comment clusters generated from the source article and comment set (as described in D5.3 Section 3.3) and a set of cluster labels, generated for the set of comment clusters, and for each cluster, the most representative quote from the comments in that cluster.

The SENSEI enabled system C, on top of the system B, additionally uses the template based summary (called 'overview'), mood information and agreement/disagreement information. Both systems comprise of the pie chart, selected quotes and options for viewing clusters of comments.

2.2 Evolution of Views

For both use cases, the UI for performing the evaluation is designed to be as neutral to the technology as possible so that it does not represent a major factor explaining the difference between subject performance.

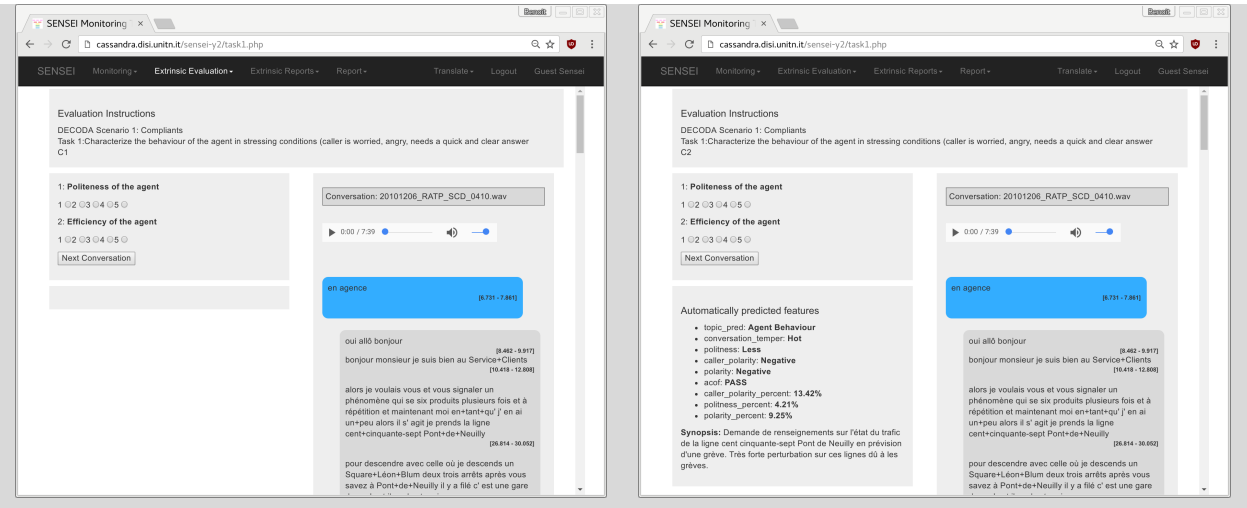


Figure 1: Screenshots of the UI presenting the features for the extrinsic evaluation task 1 on speech (ACOF evaluation). Left is condition C1 (the baseline) and right is condition C2 (SENSEI).

For the speech use case, Figures 1 and 2 show the UI instantiated for the DECODA data. The UI is an evolution of the Kibana interface produced during Y2 of the project. We created a new "evaluation" panel which allows the selection of evaluation conditions, and shows the UI elements provided to subjects for the evaluation. The UI is divided in three parts:

- An instructions part which contains a general description of the task.
- An evaluation form part which contains form elements for input by the subjects.
- A ad-hoc report / summarization view part which contains summarization elements produced by the core components.

The differential between the baseline condition and the SENSEI condition is embodied by the presence of extended UI elements where needed. Both scenarios are implemented according to this framework.

For the social media scenario, Figures 3 and 4 show the views implemented for the extrinsic evaluation. The UI has been significantly updated from Period 2, both in terms of used SENSEI outputs and presentation. The SENSEI enabled UI consists of 3 parts:

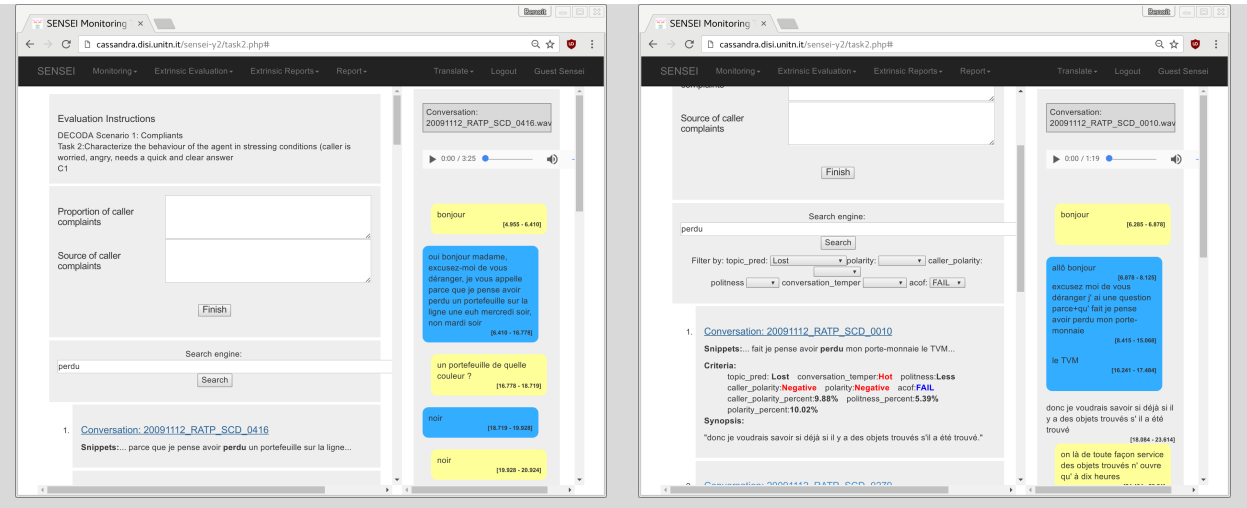


Figure 2: Screenshots of the UI presenting the features for the extrinsic evaluation task 2 on speech (Information retrieval). Left is condition C1 (the baseline) and right is condition C2 (SENSEI).

- On the left column: Pie Chart with the distribution of discussed topics; and for system C also mood and agreement/disagreement information that appears on rollover.
- On the central column: selected comments for each of the comments clusters
- On the right column: panel that displays all the comments in a selected cluster

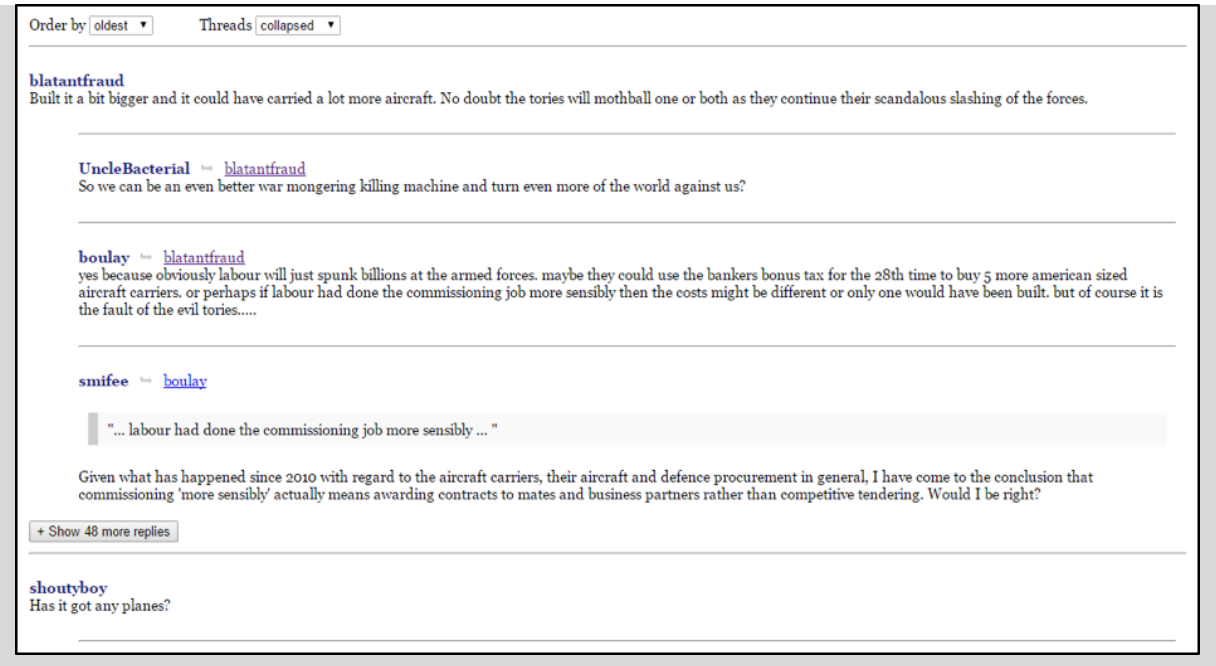


Figure 3: Screenshot of the baseline UI presenting the features for the social media extrinsic evaluation.

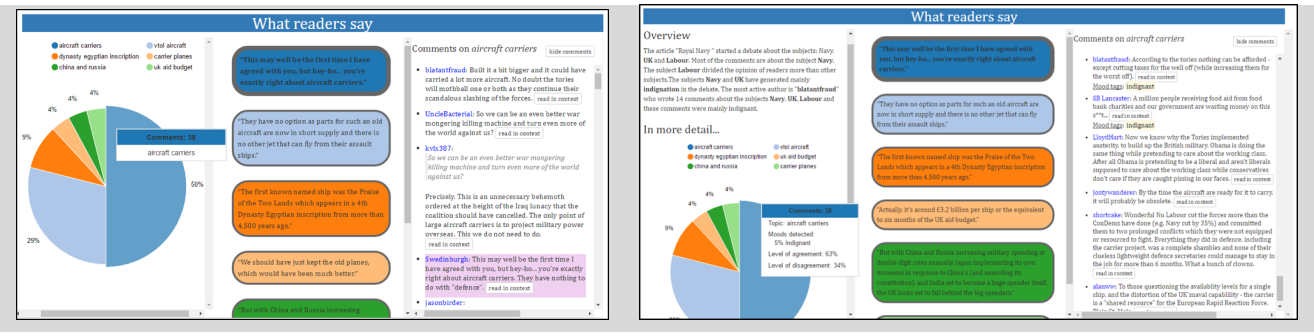


Figure 4: Screenshots of the UI presenting the features for the SENSEI enabled social media extrinsic evaluation: the systems B (left) and C (right).

3 Implementation Details

The SENSEI prototypes are web-based, with an HTTP server supplying rich content to the client's browser. The back-end consists of component modules communicating with each other by REST calls which obtain information from a central repository and add information to it. Most of the back-end modules are "off-line": they process documents and add information to them for presentation later.

The repository serves to feed all the modules with conversational data and store the output of offline modules, while providing fast, convenient access to the web interfaces.

3.1 Updates to the Repository

The repository specified in D5.1 and updated in D6.2 has been successfully used, but some improvements have been carried out as follows in order to better suit the project's needs.

- We added a GET endpoint (`document/{id}/content`) which returns only the content of the specified document (as a string value of `result` wrapped in the usual JSON data format specified in D5.1). This allows clients that need only the plain-text content (e.g., for presentation in a user interface or to carry out NLP from the ground up) to download a smaller payload without the document features and annotation sets.
- We used Spring AOP's target source pooling to create a configurable pool of objects in the intermediate layer between the REST service itself and the MongoDB storage in order to improve the repository's behaviour under heavy loads. [1] The benchmarking results are shown in Appendix B.
- We updated the Jackson and Spring libraries used in the software.

The WADL (Web Application Description Language) for the current version of the repository is included for reference as Appendix A.

3.2 Rated Questionnaire Views

We prepared an interface for the annotation to be done remotely through the web platform. The interface was done to be easily understandable and to be easy to use. The platform allows to dynamically select and extract conversations and to adapt the visualization to the dimensions of the screen of the user. Different views have been created for each type of evaluation and all logs of the user have been saved so that it is possible to analyze and aggregate data. To

retrieve and save data the AJAX technology with PHP, MySQL, Elastic Search and Kibana on the server side has been used.

3.2.1 Extrinsic Evaluation

The Extrinsic Evaluation tool has been designed to manage the evaluation process. In the main menu there is a new option – called "Extrinsic Evaluation" – where users can choose to start the evaluation process. In the first page of the evaluation process, users have to select:

- Service (LUNA or DECODA)
- Evaluation Scenario
- Evaluation Condition
- Evaluation Task

With respect to the selected options, the tool starts the evaluation process showing the related page. The main differences between LUNA and DECODA are in the task and condition options.

Evaluation Scenarios For both services (LUNA and DECODA) there are two different scenarios. When users select a scenario, the tool extract a subset of conversations to be evaluated.

Evaluation Conditions User can select two different conditions: (1) Condition **C1** to use the call listening tools they are currently using to access the source audio of the phone calls. In C1 condition user can read the transcription and listen to the audio file of each conversation, answering the questions and go on to the next conversation until the subset of conversations is completed. At the end, the tool stores in the database the answers with user information and the time to complete the whole evaluation process.The interface in condition C1 appears as described in Section 2.2 on Figure 1 (left).

Condition **C2** to use the SENSEI generated synopsis of the source phone calls In C2 condition, users can read transcription and listen to the audio as in C1, but they have also the SENSEI features and the synopses (see Figure 1 right in Section 2.2).

SENSEI features are stored in the database and are linked to the conversation. To store the features in the database, there is a procedure that reads data from JSON files and stores in the database with the correct conversation ID. The evaluation process is the same as in C1, with time counter of the process and storing the evaluation data in the database.

Evaluation Tasks For each service, scenario and condition, there are 2 tasks. In the Task 1 users have to evaluate a subset of conversations. Users evaluate each conversation and go on until the end of a subset. In Task 2 users can search in all available conversations using a search engine that works differently for each evaluation condition. In condition C1 users have a full-text search engine that uses MySQL full-text search capabilities. Users can search conversations containing keywords and the system returns a subset of conversations that match the search query. They can listen and read each conversation by selecting it from the search results or make any other search they want. When users have finished the task, they have to answer the questions and finalize the evaluation.

In condition C2, users can use the SENSEI features. They can search the conversations with full-text search engine, and filter the results with respect to SENSEI features, such as polarity, politeness, etc.. They can see all the features provided by SENSEI components for each conversation and a snippet with a portion of the text of a conversation transcription, including the keywords they have entered in search engine. Figure 2 (right) in Section 2.2 depicts the interface for the task 2 of the evaluation condition 2.

3.2.2 User Experience Questionnaire

User experience questionnaire allows users to compile a set of questions regarding the experience with the evaluation process. The tool and the DB are changed to support this task, adding new tables to MySQL to manage question answering features. Figure 5 depicts the user experience questionnaire interface.

3.2.3 Reports

A reporting feature has been implemented in the tool to get reports of the extrinsic evaluations and user experience questionnaires quickly and easily. Users can select from the Extrinsic Reports menu which report they want to see and/or download.

Evaluation Reports The evaluation reports are distinct for each task, so users can choose the report for the task they wants to inspect. In the report page there is a tabled report of the evaluation displaying all the evaluation information, such as user, question, answer, evaluation time, date, and others. All columns are filterable and sortable to make search inside report quickly and easily. Reports can be downloaded in Excel format, thanks to the use of the PHPExcel library. Figure 6 depicts the evaluation report interface.

User Experience Reports User experience reports display the results of the user experience questionnaires compiled by the users. Similar to Evaluation Reports, users can filter

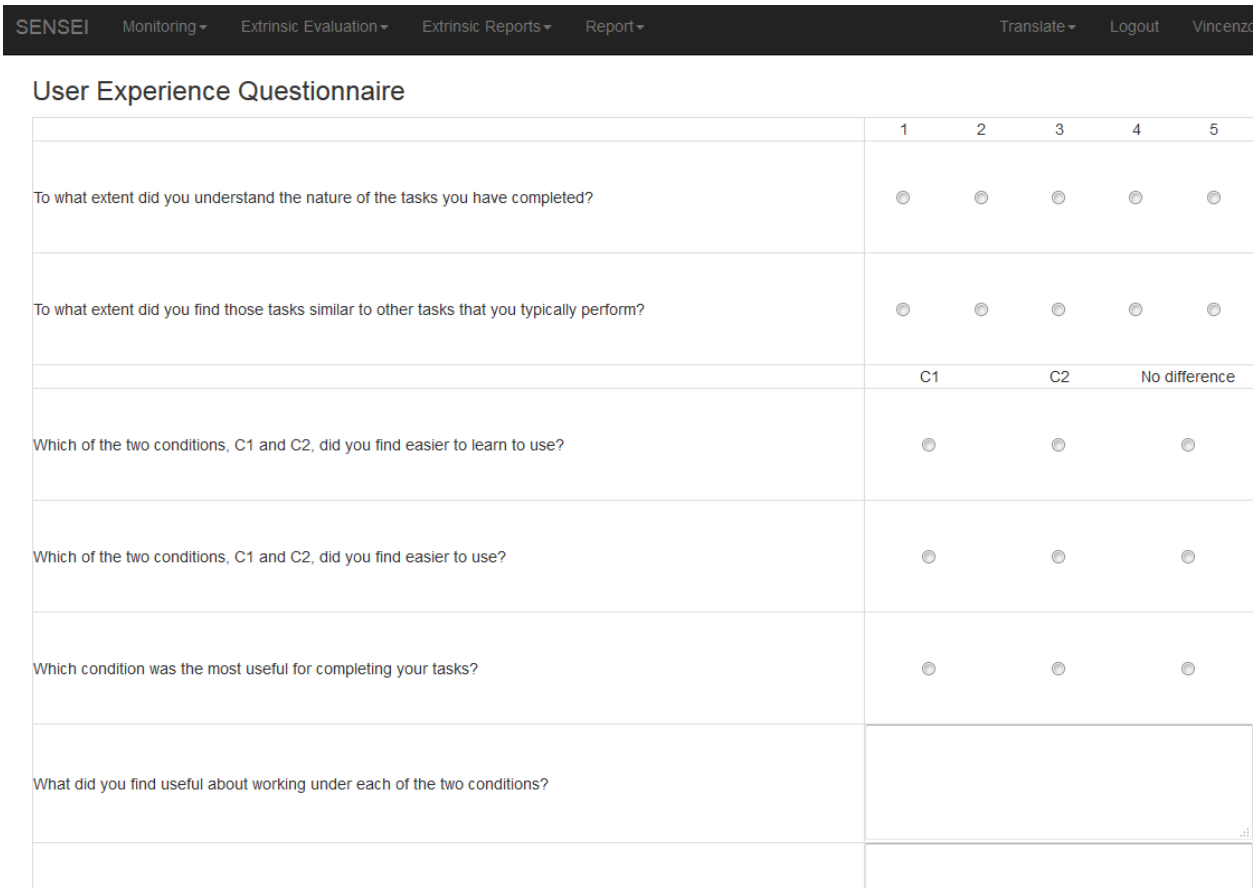


Figure 5: User Experience Questionnaire.

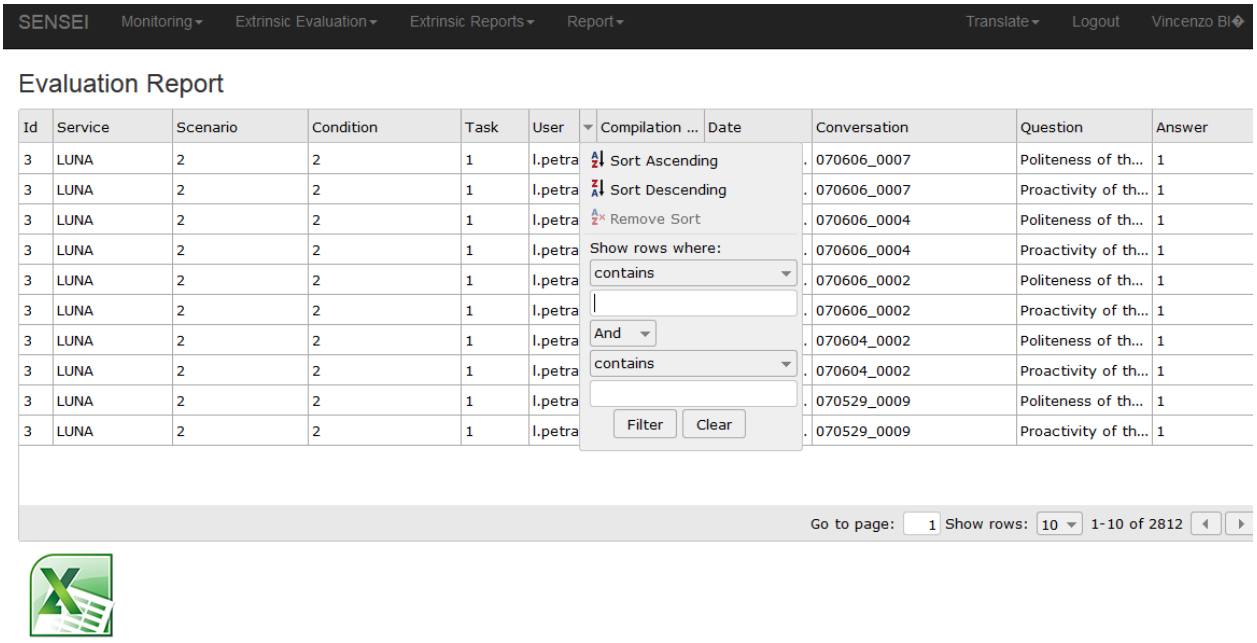


Figure 6: Evaluation report interface.

and sort each column and download the report in Excel format.

3.3 Ad-hoc Report Views

Within the past years of SENSEI USFD developed several software components to analyse social media stream of data. These components include summarization approaches to condense the big amount of user news comments to short summary that gives an overview of the original conversation and is quick to read by users.

The summarization components make use of topic clustering, i.e. producing clusters containing comments addressing the same topic. Our topic clustering component uses various features and supervised machine learning along with graph-based clustering to determine the topic clusters. Note, unlike most clustering approaches our solution does not need to know in advance the number of topic cluster but determines this automatically while performing the clustering task. The summarization components also make use of topic cluster labels – a label is assigned each topic cluster and aims to describe the content of that topic cluster. For this we again went the path of supervised machine learning. Using the USFD gold standard data and feature analysis and extraction we learned a linear model that automatically suggests a label for a given topic cluster. This label is determined from a bigger list of candidate labels – we use terms to represent labels and those terms are extracted from the news article as well as from its comments. The linear model ranks all the candidate labels and we use the highest ranking label to describe the topic cluster. Finally, summaries are constructed based on the generated labels. We generate extractive and abstractive summaries.

For extractive summaries we use the labels to select representative comments from the clusters. The abstractive summaries are generated by extending the labels to full natural sentences using templates and topic cluster meta data information. Individual components are described in detain in the deliverable D5.3.

4 Exploitation of SENSEI Technologies for Monitoring and Predicting Brexit

In this section we describe the system used for monitoring and predicting Brexit campaign. The system is based on Natural Language Processing techniques such as sentiment and argumentation analysis (agreement/disagreement prediction) that produce a set of predictions about authors' support and opposition towards the discussed topics (i.e. stance). The system showcases the success of the NLP techniques developed within the SENSEI project.

4.1 System Infrastructure

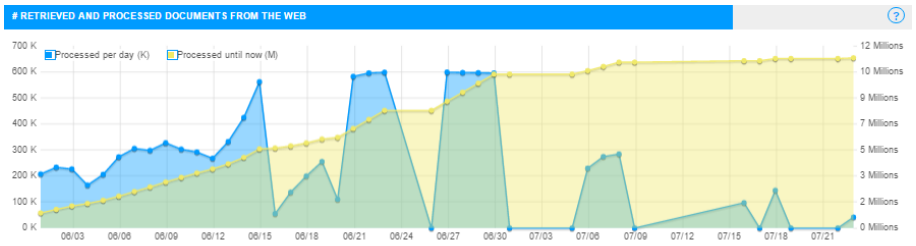
We set up a system for monitoring and predicting Brexit during May 2016. Every night, starting from May 19 to July 19 2016, Websays creates a data dump of the conversations about Brexit monitored from more than 4000 sources (see details in D2.4). About 80% of the data comes from Twitter The data in the dump refers to the day before and is filtered by topic (only posts about Brexit) and tagged with keywords referring to specific topics:

- Brits Abroad
- Brussels
- Immigration
- Leave
- Remain
- Scotland
- Terrorism

The cues for extracting keywords and associate posts to topics are hashtags like #StrongerIn for Remain and #VoteLeave for Leave. In the two-months period from May 19 to July 19 we collected almost 12M posts from social media, as depicted in Figure 7, 7M in the month before the referendum (May 19 to June 21). At 7.00am the server downloads the dump and process the XML file converting it in JSON files compatible with the SENSEI format. The entire dump is converted in about 200K files (one per post or comment).

When the conversion process is finished, files are sent to the UNITN servers for the automatic annotation process and documents are uploaded on the MariaDB database connected to sense-eu.

Figure 7: Graph of the data collected



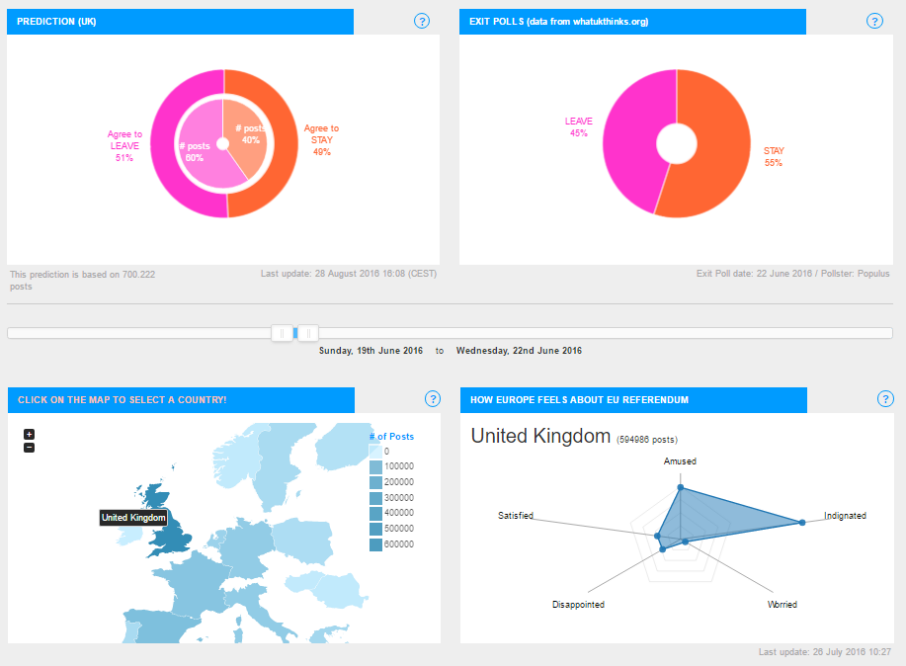
Every 2 or 3 minutes, 2 different servers process blocks of 1.000 files to extract mood and agreement For the prediction of the Brexit referendum we used only a subset of 3M posts, due to processing reasons. Every 1.000 files processed results are uploaded on the MariaDB database connected to the sense-eu website. At 10.00am and 3 PM cache on sense-eu is deleted so that new data on the database can be fetched and show as result. Process repeats every day loading the data of 2 days before. The technical details of the system infrastructure are listed below:

- PHP, JS and CSS based on bootstrap
- MariaDB as database
- Bash scripting, PHP, Java, Perl and Python for processing data
- Cache system self-made
- Server GNU/Linux Ubuntu 16.04 for sense-eu and CentOS for automatic annotation

4.2 Analytic Tools

We created the page `http://sense-eu.info` on May 19,and we maintained it until July 19. Exploiting Agreement/Disagreement and mood technologies developed within the SENSEI project (see D3.4 and D4.3), we developed boxes to display the results of the data analysis to the public. Figure 8 shows the main boxes: The prediction on Brexit, compared to the latest exit polls and to raw prediction based only on the volume of keywords of leave and remain. The prediction pie chart displays the percentage of posts classified in agreement/disagreement with the topics Leave and remain respectively. For example the portion computed as Leave contains posts in agreement with Leave and in disagreement with Remain, and vice-versa for the portion computed as Remain. The System for the automatic labelling of posts with agreement and disagreement labels is described in detail in D4.3, briefly summarizing it makes use of 2. the agreement/disagreement prediction algorithm makes use of cross-language stylometric features such as:

Figure 8: Boxes displaying the results of the analyses on Brexit.



- ratio of internal punctuation
- ratio of final punctuation
- ratio of expression marks
- ratio of quotes
- ratio of open parentheses
- ratio of close parentheses
- ratio of apices
- ratio of numbers
- ratio of operators
- ratio of hashtags
- ratio of mentions
- ratio of urls
- ratio of uppercase letters
- ratio of lowercase letters
- ratio of bigrams (words and chars)
- ratio of trigrams (words and chars)

- ratio of tetragrams (words and chars)
- ratio of positive and negative emoticons

We trained the system using 66% of the CorEA corpus (about 2900 posts in Italian, manually annotated with agree=1, disagree=-1 and neutral/not applicable=0, inter-annotator agreement is k=0.57 on 3 classes and k=0.85 on 2 classes. we used only 1 and -1 as values for learning, not 0 values, that are about 2000 posts. The task was a regression and the distribution was bimodal, but well balanced.). We tested the algorithm on the test split (33% of CorEA), we used a Support Vector Regressor [2] as learning algorithm, and obtained a Mean Absolute Error (MAE) of 0.32.

The mood detection system for social media, described in detail in D3.3, has been used for displaying the mood of conversations about Brexit by EU country in a dynamic map changing on mouse rollover. Other boxes include word clouds, reported in Figure 9 below.

Figure 9: Boxes displaying the word clouds.

We developed a general word-cloud and two specific word-clouds displaying the adjectives found in the context of the topics Leave and Remain respectively.

5 Software Distribution

This section lists the software modules which have been made available to the community. In general, we made available software modules with with source code and models for the languages covered in the project. Some modules are more general than the SENSEI project and are therefore released in that more general context (without SENSEI glue code). We tried to make available as much software as possible for a member of the general community to be able to conduct conversation analysis with a level of precision similar to what was achieved during the project. Due to license restrictions, some of the pieces are not available and have to be filled in by users. The source code is hosted at `https://gitlab.lif.univ-mrs.fr/sensei` when it does not have its own hosting.

5.1 List of software modules

Table 1 lists the software modules released by the SENSEI project. They cover the whole pipeline: data collection, annotation, semantic and para-semantic parsing, discourse analysis, summarization, evaluation and dissemination.

Partner	WP	Module	License
AMU	WP3	Syntactic parser (Macaon)	LGPL v3
AMU	WP3	Semantic parser (Fastsem)	GPL v3
AMU	WP5	Synopsis generator	GPL v3
AMU	WP3	Sentiment analyser (SemEval2016)	GPL v3
AMU	WP5	Abstractive cluster labeller	GPL v3
AMU	WP6	Computation platform	GPL v3
UESSEX	WP4	Coreference resolver (BART)	APL v2
WEBSAYS	WP2	Website parsers	GPL v3
UNITN	WP4	Discourse Parser (CoNLL2016)	LGPL v3
UNITN	WP4	Agreement predictor (ADRian)	MIT
UNITN	WP3	Mood predictor (coMOOD)	LGPL v3
USFD	WP5	Repository	LGPL v3
USFD	WP5	Repository tools	LGPL v3
USFD	WP5	Comment clustering and summarization	GPL v3
USFD	WP3	Event/sentiment detector (GATE)	LGPL v3
USFD	WP6	Social media eval prototype	TBD
TP	WP1	ACOF tool	TBD

Table 1: Table summarizing the software modules provided by the SENSEI project. Note: this table is not final.

5.2 Overview of software modules

This section gives a broad overview of the software modules delivered within the project.

- **Syntactic parser (Macaon)**: This module provides sentence-level part-of-speech tagging and syntactic parses.
  - Models: French, English, Italian
  - License: LGPL
  - Open source: yes
  - Programming languages: C, python
  - Repository integration: yes
  - Dependencies: libxml, glib, openfst, gfsm
  - URL: `https://gitlab.lif.univ-mrs.fr/benoit.favre/macaon`
- **Semantic parser (Fastsem)**: This module performs sentence-level FrameNet semantic analysis of speech and social media text, given syntactic analyses provided by the syntactic parser.
  - Models: French, English, Italian
  - License: GPL
  - Open source: yes
  - Programming languages: python
  - Repository integration: yes
  - Dependencies: liblinear
  - URL: `https://gitlab.lif.univ-mrs.fr/sensei/fastsem`
- **Synopsis generator**: This module generates synopses of speech conversations by recombining training data synopses and filling detected concept slot.
  - Models: French
  - License: GPL
  - Open source: yes
  - Programming languages: python
  - Repository integration: yes
  - Dependencies: icsiboost

- URL: `https://gitlab.lif.univ-mrs.fr/sensei/summarizer`

- **Sentiment analyser (SemEval2016)**: This module can predict the sentiment polarity of short text messages.
  - Models: English
  - License: BSD
  - Open source: yes
  - Programming languages: python
  - Repository integration: no
  - Dependencies: keras
  - URL: `https://gitlab.lif.univ-mrs.fr/mickael.rouvier/SemEval2016`

- **Abstractive cluster labeller**: This rest service will label a textual piece with a label (abstractive) using DBpedia
  - Models: English
  - License: GPL
  - Open source: yes
  - Programming languages: python
  - Repository integration: yes
  - Dependencies: bottle, gensim, rdflib, networkx
  - URL: `https://gitlab.lif.univ-mrs.fr/balamurali.ar/labelling_rest`

- **Computation platform**: This module helps with the integration of the various components as a general pipeline. It provides REST services and repository integration for other components.
  - Models: n/a
  - License: GPL
  - Open source: yes
  - Programming languages: python
  - Repository integration: yes
  - Dependencies: bottle
  - URL: `https://gitlab.lif.univ-mrs.fr/sensei/sensei_pipeline`

- **Coreference resolver (BART)**: This module resolves coreferences in conversations.

- Models: English, French, Italian
- License: APL
- Open source: yes
- Programming languages: java
- Repository integration: yes
- Dependencies: Stanford NER, WEKA, TexPro
- URL: `https://gitlab.lif.univ-mrs.fr/sensei/bart-sensei`

- **Website parsers**: These parsers can be used to crawl a range of websites for online conversations.
  - Models: n/a
  - License: GPL v3
  - Open source: yes
  - Programming languages: java
  - Repository integration: no
  - Dependencies: TBD
  - URL: `https://gitlab.lif.univ-mrs.fr/sensei/website-scrapers`

- **Discourse Parser (CoNLL2016)**: This system provides PDTB-style discourse parsing for written English text.
  - Models: English
  - License: LGPL v3
  - Open source: yes
  - Programming languages: php, perl
  - Repository integration: json schema only
  - URL: `https://github.com/esrel/DP`

- **Agreement predictor (ADRian)**: This system provides agree/disagree classification in social media (used in social media prototype template-based summarisation, sense-eu).
  - Models: Language independent
  - License: MIT
  - Open source: yes
  - Programming languages: php, html, js
  - Repository integration: json schema only

- Dependencies: none
- URL: `http://cassandra.disi.unitn.it/agreement-1.0.tar.gz`

- **Mood predictor (coMOOD)**: This system provides mood prediction (used in social media prototype template-based summarisation, sense-eu)
  - Models: English, French, Italian
  - License: LGPL v3
  - Open source: no
  - Programming languages: perl, java
  - Repository integration: json schema only
  - Dependencies: none
  - URL: `http://www.sense-eu.info/data/Sensei_Mood_V0.2.tar.gz`

- **Repository**: The conversation repository can store conversations and analyses generated by other modules. It is the central synchronization point of the pipeline.
  - Models: n/a
  - License: LGPL v3
  - Open source: yes
  - Programming languages: java
  - Repository integration: n/a
  - Dependencies: Jackson; Apache Commons-Lang; Apache CXF; Spring Framework; Spring MongoDB interface; a MongoDB instance running on the server
  - URL: `http://www.dcs.shef.ac.uk/~adam/stuff/sensei/repository.war-20160602-1520.zip`

- **Repository tools**: Tools for easily accessing the repository from the command line
  - Models: n/a
  - License: LGPL v3
  - Open source: yes
  - Programming languages: python
  - Repository integration: n/a
  - Dependencies: BeautifulSoup
  - URL: `http://www.dcs.shef.ac.uk/~adam/stuff/sensei/sensei-USFD-tools.zip`

- **Comment clustering and summarization**: This system can link comments in a conversation, cluster them and generate a summary from the clusters.
  - Models: English, French, Italian
  - License: GPL v3
  - Open source: yes
  - Programming languages: java
  - Repository integration: yes
  - Dependencies: none
  - URL: `http://www.dcs.shef.ac.uk/~adam/stuff/sensei/SenseiPrototypeTools.zip`
- **Event/sentiment detector (GATE)**: This system provides standard NLP, named-entity recognition, event detection, sentiment detection.
  - Models: English
  - License: LGPL[1]
  - Open source: yes
  - Programming languages: java, groovy, jape
  - Repository integration: yes
  - Dependencies: GATE, ARCOMEM or SentiStrength
  - URL: `http://www.dcs.shef.ac.uk/~adam/stuff/sensei/sensei-USFD-gate-pipelines.zip`
- **Social media eval prototype**: This module contains the evaluation UI for the social media use case.
  - Models: n/a
  - License: LGPL v3
  - Open source: yes
  - Programming languages: php, js, css
  - Repository integration: yes
  - Dependencies: Apache server

[1] The tool includes two pipelines: one using the ARCOMEM sentiment detection software supplied complete, and one using the SentiStrength tool (better results) but it is missing the relevant jar which cannot be distributed by SENSEI.

- URL: `http://www.dcs.shef.ac.uk/~adam/stuff/sensei/sensei-prototype-UI-USFD.zip`

- **ACOF tool**: This module provides an annotation interface for Agent Conversation Observation Form (ACOF) and support for the extrinsic evaluation.
  - Models: French, Italian
  - License: TBD
  - Open source: yes
  - Programming languages: php
  - Repository integration: yes
  - Dependencies: Web server, PHP 5.3, MySQL 5.1
  - URL: `https://gitlab.lif.univ-mrs.fr/sensei/acof`

5.3 SENSEI processing environment

This section presents a formal description of the platform which integrate all SENSEI services. The platform is available at `https://gitlab.lif.univ-mrs.fr/sensei/sensei_pipeline`.

This pipeline is a set of annotation services which can be run against the conversation repository or run as standalone REST services. Example REST services and repository-integrated modules were developed by the project and circulated to the partners in order to ease the elaboration of a comprehensive platform.

The SENSEI processing environment (SPE) is implemented in python based on the bottle.py framework. A full-fledged repository client is provided, it can be used with an API which reflects the REST API of the repository with python-friendly constructs, and provides ssh tunnel management. It can also be completely driven by the command line and it includes a thorough test suite for unit tests and benchmarking. The system provides amenities for polling the repository and executing code when specific document features become available. Already processed documents are marked with additional features. The processing environment includes a generic rest service which can be setup to pipe the output of commands when REST URLs are queried, or when document annotations become available.

6 Conclusions

In this deliverable we have presented SENSEI's final summarization views and their implementation details for both the speech and social media use cases. The presented prototypes are geared towards the extrinsic evaluation described in the deliverable D1.4. Throughout the project, the developed conversation repository was successfully used for sharing data and the output of the models developed within other work packages. The repository was updated with new required features when needed.

Additional to the prototypes for the evaluation of the technology, the project has also setup a website for the monitoring and successfully predicting the outcome of the Brexit campaign. The website showcases the technology developed for social media analysis and makes use of the models developed within WP3-5 such as agreement/disagreement, mood and sentiment prediction. Consequently, the technology developed within the project is presented both to the 'professional' end-users such as call-center employees and media workers and to general public.

Following the reviewers' recommendations the software developed within the project is being released to the community.

References

[1] Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Thomas Risberg, Alef Arendsen, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Rossen Stoyanchev, Phillip Webb, Rob Winch, Brian Clozel, Stephane Nicoll, and Sebastien Deleuze. *Spring Framework Reference Documentation*, 2014.

[2] Shirish Krishnaj Shevade, S Sathiya Keerthi, Chiranjib Bhattacharyya, and Karaturi Radha Krishna Murthy. Improvements to the smo algorithm for svm regression. *Neural Networks, IEEE Transactions on*, 11(5):1188–1193, 2000.

[3] Evgeny A. Stepanov, Benoit Favre, Firoj Alam, Shammur Absar Chowdhury, Karan Singla, Jérémy Trione, Frédéric Béchet, and Giuseppe Riccardi. Automatic summarization of call-center conversations. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) - Demo Papers*, Scottsdale, Arizona, USA, December 2015. IEEE.

[4] Jérémy Trione, Benoit Favre, and Frédéric Béchet. Beyond utterance extraction: summary recombination for speech summarization. In *Interspeech, San Francisco (USA)*, September 2016.

Appendix A Repository WADL

```
<?xml version="1.0"?>
<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <grammars/>
  <resources base="http://localhost:8080/repository/">
    <resource path="/">
      <resource path="doc-feature-query/full">
        <method name="GET">
          <doc>Return documents that match the complex feature query</doc>
          <request/>
          <response>
            <representation mediaType="application/json">
              <doc>list of full documents</doc>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="doc-feature-query/ids">
        <method name="GET">
          <doc>Return IDs of documents that match the complex feature query</doc>
          <request/>
          <response>
            <representation mediaType="application/json">
              <doc>list of document IDs</doc>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="document">
        <method name="POST">
          <doc>Store the document (JSON) and return the ID</doc>
          <request>
            <representation mediaType="application/json"/>
          </request>
          <response>
            <representation mediaType="application/json">
              <doc>newly allocated ID</doc>
            </representation>
          </response>
        </method>
      </resource>
      <resource path="document/plaintext">
        <method name="POST">
          <doc>Store the document (plain text) and return the ID</doc>
          <request>
            <representation mediaType="text/plain">
              <param name="request" style="plain" type="xs:string"/>
```

```
      </representation>
    </request>
    <response>
      <representation mediaType="application/json">
        <doc>newly allocated ID</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="document/websays">
  <method name="POST">
    <doc>Store the document(s) (XML) and return the list of IDs</doc>
    <request>
      <representation mediaType="application/xml"/>
      <representation mediaType="text/xml"/>
    </request>
    <response>
      <representation mediaType="application/json">
        <doc>newly allocated ID</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="document/{docId}/annotation-set/{name}/{annId}">
  <param name="docId" style="template">
    <doc>document ID</doc>
  </param>
  <param name="name" style="template" type="xs:string">
    <doc>annotation set name</doc>
  </param>
  <param name="annId" style="template">
    <doc>annotation ID</doc>
  </param>
  <method name="DELETE">
    <doc>Delete an annotation from the document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="document/{id}">
  <param name="id" style="template"/>
  <method name="DELETE">
    <doc>Delete the specified document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
```

```
  <method name="GET">
    <doc>Retrieve the specified document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="document/{id}/annotation-set/{name}">
  <param name="id" style="template">
    <doc>document ID</doc>
  </param>
  <param name="name" style="template" type="xs:string"/>
  <method name="DELETE">
    <doc>Delete an annotation set from the document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method name="PUT">
    <doc>Add annotations (created the named set if needed) to the document (specified by ID)</doc>
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="document/{id}/annotations">
  <param name="id" style="template"/>
  <method name="PUT">
    <doc>Add annotations (creating sets as needed) from JSON to the document (specified by ID)</doc>
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="document/{id}/content">
  <param name="id" style="template"/>
  <method name="GET">
    <doc>Retrieve the specified document's content</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
```

```
  </method>
</resource>
<resource path="document/{id}/feature/{key}">
  <param name="id" style="template">
    <doc>document ID</doc>
  </param>
  <param name="key" style="template" type="xs:string">
    <doc>feature name</doc>
  </param>
  <method name="DELETE">
    <doc>Delete a feature from the document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="document/{id}/features">
  <param name="id" style="template">
    <doc>document ID</doc>
  </param>
  <method name="DELETE">
    <doc>Delete features from the document</doc>
    <request>
      <representation mediaType="application/json">
        <doc>list of feature names</doc>
      </representation>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method name="GET">
    <doc>Return the complete feature map of the specified (by ID) document</doc>
    <request/>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method name="POST">
    <doc>Return a feature map containing the selected features of the specified (by ID) document</do
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method name="PUT">
    <doc>Add features (from JSON) to the document (specified by ID)</doc>
```

```
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="documents">
  <method name="GET">
    <doc>Return all document IDs in the repository</doc>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="documents/false/feature/{key}">
  <param name="key" style="template" type="xs:string"/>
  <method name="GET">
    <doc>Return documents with the specified feature missing, null, empty (list, string) or zero</do
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="documents/false/feature/{key}/{limit}">
  <param name="key" style="template" type="xs:string"/>
  <param name="limit" style="template" type="xs:int"/>
  <method name="GET">
    <doc>Return documents (up to the limit) with the specified feature missing, null, empty (list, s
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="documents/features">
  <method name="GET">
    <doc>Return documents with features matching the query (flat features only)</doc>
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
```

```
  </method>
</resource>
<resource path="documents/missing/feature/{key}">
  <param name="key" style="template" type="xs:string"/>
  <method name="GET">
    <doc>Return documents without the specified feature</doc>
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="documents/missing/feature/{key}/{limit}">
  <param name="key" style="template" type="xs:string"/>
  <param name="limit" style="template" type="xs:int"/>
  <method name="GET">
    <doc>Return documents (up to the limit) without the specified feature</doc>
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="documents/present/feature/{key}">
  <param name="key" style="template" type="xs:string"/>
  <method name="GET">
    <doc>Return documents with the specified feature (which might be false, zero, or empty)</doc>
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
  </method>
</resource>
<resource path="documents/present/feature/{key}/{limit}">
  <param name="key" style="template" type="xs:string"/>
  <param name="limit" style="template" type="xs:int"/>
  <method name="GET">
    <doc>Return documents (up to the limit) with the specified feature (which might be false, zero,
    <request/>
    <response>
      <representation mediaType="application/json">
        <doc>list of documents</doc>
      </representation>
    </response>
```

```
      </method>
    </resource>
    <resource path="test-doc">
      <method name="GET">
        <doc>Create and store a standard test document</doc>
        <response>
          <representation mediaType="application/json">
            <doc>newly allocated ID</doc>
          </representation>
        </response>
      </method>
      <method name="POST">
        <doc>Create and store a test document</doc>
        <request>
          <representation mediaType="text/plain">
            <param name="request" style="plain" type="xs:string"/>
          </representation>
        </request>
        <response>
          <representation mediaType="application/json">
            <doc>newly allocated ID</doc>
          </representation>
        </response>
      </method>
    </resource>
    <resource path="test-doc/{text}">
      <param name="text" style="template" type="xs:string">
        <doc>text</doc>
      </param>
      <method name="GET">
        <doc>Create and store a test document</doc>
        <request/>
        <response>
          <representation mediaType="application/json">
            <doc>newly allocated ID</doc>
          </representation>
        </response>
      </method>
    </resource>
  </resource>
 </resources>
</application>
```

Appendix B Benchmarking

We tested three endpoint functions with 8 to 128 simultaneous client connections using Apache's benchmarking tool (`ab`). The following results are averaged over 1000 repetitions of each combination of function and number of clients, with a pool of 16 MongoDB wrappers in the repository.

request	nbr of clients	mean ms per request
get 1 document	8	61
	16	136
	32	338
	64	942
	128	1886
list all documents	8	742
	16	1906
	32	3414
	64	6826
	128	13606
query by features	8	92
	16	180
	32	382
	64	764
	128	1544