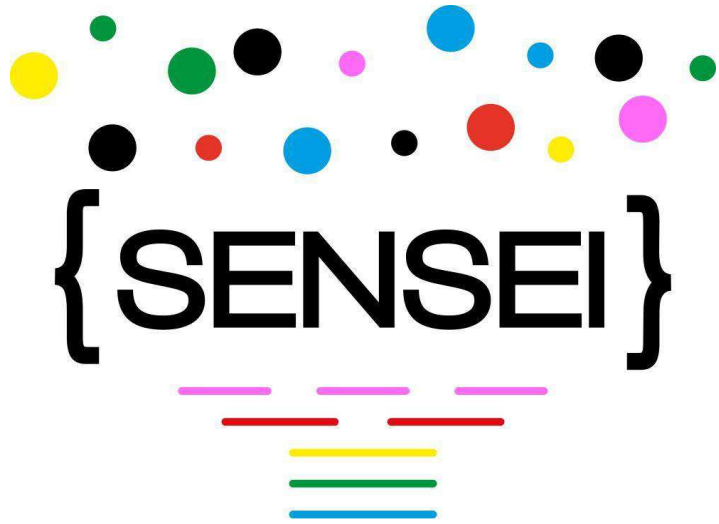


D6.1 – Report on the architecture of SENSEI conversation summarization prototype

Document Number	D6.1
Document Title	Report on the architecture of SENSEI conversation summarization prototype
Version	2.0
Status	Final
Work Package	WP6
Deliverable Type	Report
Contractual Date of Delivery	31.10.2014
Actual Date of Delivery	31.10.2014
Responsible Unit	AMU
Keyword List	Prototype, evaluation, requirements, specifications
Dissemination level	PU

Editor

Benoit Favre (AMU)

Contributors

Ahmet Aker (USFD)
Benoit Favre (AMU)
Carmelo Ferrante (UNITN)
Adam Funk (USFD)
Vincenzo Lanzolla (TP)
Giuseppe Riccardi (UNITN)

SENSEI Coordinator

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

Document change record

Version	Date	Status	Author (Unit)	Description
0.1	07/22/2014	Draft	Benoit Favre (AMU)	Table of Content
0.2	08/25/2014	Draft	Benoit Favre (AMU)	Added evaluation description Added user interface stubs
0.2a	08/25/2014	Draft	Giuseppe Riccardi (UNITN)	Add initial requirements
0.2b	08/25/2014	Draft	Carmelo Ferrante (UNITN)	Add software list
0.3	08/31/2014	Draft	Benoit Favre (AMU)	Add appendix A, requirements, developer process, backend modules, schemas
0.4	09/01/2014	Draft	Benoit Favre (AMU)	Add REST specification, mockups
0.5	09/11/2014	Draft	Benoit Favre (AMU)	More screenshots and mockups
0.6	09/14/2014	Draft	Benoit Favre (AMU)	Refine REST protocol, update scenario section
0.7	09/15/2014	Draft	Benoit Favre (AMU)	Add social-media use case scenarios
0.8	09/16/2014	Draft	Benoit Favre (AMU)	REST URLs reverted to http except for gateway
0.9	09/18/2014	Draft	Adam Funk (Sheffield)	Update conversation repository description
0.9b	09/18/2014	Draft	Ahmet Aker (Sheffield)	Add clustering and linking module
0.9c	09/18/2014	Draft	Benoit Favre (AMU)	Update backend modules
0.10	09/19/2014	Draft	Benoit Favre (AMU)	Rewrite UI modules, add conclusion
0.11	09/19/2014	Draft	Vincenzo Lanzolla (TP)	AOF module
0.12	09/26/2014	Draft	Benoit Favre (AMU)	Add SM scenario
1.0	09/26/2014	Draft	Benoit Favre (AMU)	Coherence with other WPs. Version ready for scientific review and quality check
1.1	09/29/2014	Draft	Giuseppe Riccardi (UNITN)	Review

1.2	09/29/2014	Draft	Elisa Chiarani (UNITN)	Quality Check
1.3	09/29/2014	Draft	Benoit Favre (AMU)	Improve diagram quality, align wording with D1.2 and D5.1, some editing,
1.4	10/15/2014	Draft	Adam Funk (Sheffield)	Scientific review
1.5	10/15/2014	Draft	Benoit Favre (AMU)	Modifications required by scientific review
2.0	10/20/2014	Final	Elisa Chiarani, Giuseppe Riccardi	Final version

Executive summary

This document describes the specifications of the prototype that will support the evaluation of conversation summarization approaches proposed in the course of the SENSEI project.

For the speech use case, the evaluation scenarios consist in helping QA professionals fill agent observation forms, as well as find uncommon situations in conversations.

For the social media use case, they consist in generating town hall summaries of comments related to news articles, finding editor picks, and supporting comment writers.

Functional analysis of these scenarios revealed that the prototype should contain a collection browser, conversation and an agent/reader views, a specialised search engine and means of inputting evaluation material. Technically, the prototype will be a server hosting backend REST modules, and exposing a web-based client. The development of the modules will draw from core technology developed in WP3, WP4 and WP5 and follow adequate quality assurance through a single deployment target virtual machine (VM), unit testing and source control.

Table of Content

Executive summary 5
1. Introduction 8
1.1 Overview 8
1.2 Objectives 8
1.3 Evaluation scenarios 8
2. Requirements 12
2.1 Organization 12
2.2 Usability 12
2.3 Software 12
2.4 Hardware 12
2.5 Latency 12
2.6 Security 12
2.7 Quality 13
3. Development process 14
3.1 Development practices 14
3.2 Source control 14
3.3 Deployment 14
3.4 Quality insurance 15
4. Detailed design 16
4.1 Guidelines 16
4.2 Architecture 16
4.3 Backend modules 16
4.4. User interfaces 22
5. Conclusion 26
5.1. Roadmap 26
Appendix A: Virtual machine user guide. 27
A.1. Virtual machine description 27
A.2. Downloading the VM 27
A.3. Networking 27
A.4. How to get an account 28
A.5. Installed software 28
A.6. Updating 28
Appendix B: mock-ups and screenshots 29
B.1. Study of existing interfaces 29
B.2. Collection browser 30

B.3. Conversation view 31
B.4. Search 31
B.5. Auditor observation form 32
B.6. Semantic concordancer 32
B.7. Statistics 32
B.8. Evaluation input 33
B.9. Agent/Blogger view 34
Appendix C: REST/JSON tutorials 35

1. Introduction

1.1 Overview

One of the objectives of the SENSEI project is to run user-oriented evaluations of the proposed approaches to conversation summarization. The SENSEI prototype encapsulates research technologies in a coherent package for performing these evaluations. It showcases conversation-oriented summaries, blogger-oriented summaries, rated questionnaire summaries, and ad-hoc reports. This document presents the requirements and specifications of the SENSEI prototype.

1.2 Objectives

The SENSEI project will perform an evaluation of the approaches to conversation summarization proposed in the course of the project in an ecological environment where professional users perform a real-life task. The evaluation focuses on the differential generated by the technology in achieving the targeted tasks. In order to support that experiment, the SENSEI project will develop a prototype which will integrate research technologies for conversation understanding into instrumented, intuitive user interfaces destined to the subjects of the evaluation: professionals targeted by the Speech and Social Media use cases.

The objectives of this document are to:

- List the requirements driving the evaluation and the development of the prototype.
- Specify the architecture and the interfaces between the components of the prototype.
- Define a software development framework, including software and hardware environment, as well as engineering guidelines.

This document is not a detailed specification of the software in order to keep the necessary agility to account for refinement of both the evaluation scenario and the research technologies during the development of the prototype.

1.3 Evaluation scenarios

Evaluation will be performed on the speech and social media use cases. The speech use case takes place in a call centre where quality assurance (QA) professionals (the users of the SENSEI prototype) are tasked with collecting agent and corpus-level statistics and atypical examples of calls for monitoring and improving the call centre. The social-media use case targets journalists and web comment readers and writers who have to tackle hundreds of reactions to news releases. The journalists' task is to gather and interpret trends, in order to steer the creation of follow-up articles while commenters shall collect information and produce new content through social interactions. The following describes the main lines of the evaluation scenario, which is defined in details in D1.2.

The objective of the prototype is to support ecological evaluation of SENSEI-created technology and approaches. In the evaluation, subjects have to perform a task which is non-trivial, matters for them and is solved in the most realistic environment, in other words a task that they already

do in their daily activities. Therefore, in both use cases, we will evaluate how well the experiment subjects (QA professionals, journalists, commenters) perform a task with existing technology and methodology, versus how well they perform it with SENSEI tools. Such evaluation is different from intrinsic evaluation which compares the output of a system to a gold standard without measuring how this system affects human behaviour. While the SENSEI project will run intrinsic evaluations on the output of systems for most subtasks, the prototype specification focuses on the extrinsic evaluation of the system. For each tasks tackled by the extrinsic evaluation, SENSEI systems will have to be adapted in order to provide helpful output for evaluation subjects.

1.1.1 Speech use case scenarios

The speech use case is embodied by the work of QA professionals in call centres, monitoring how agents handle inbound and outbound calls. As detailed in D1.2, several use cases have been selected for designing the evaluation scenario: (1) the automatic generation of call surveys and (2) conversation oriented summaries. In the first use case, QA professionals listen to a sample of conversations from a given agent and fill an Agent Observation Form (AOF) about agent behaviour, such as communication skills, politeness, respect of the script, etc. These forms are used to monitor how agents perform their work and perform targeted training when needed. In the second use case, QA professionals study conversations from a caller perspective, gathering tackled topics, problems, solutions, etc. The outcome is used, for instance, to improve conversation scripts and agent training material.

The extrinsic evaluation is elaborated in the framework of these use cases. It will be supported by the RATP-DECODA and LUNA datasets which consist of a set of conversations recorded in call centres. The evaluation can be cast as two general scenarios, even though both might not be evaluated (see D1.2 for scenario details).

The first scenario is “agent observation form-filling” a task relying on rated questionnaire summaries. Subjects have to analyse a set of conversations and fill an AOF for a given agent. For each question of the AOF, they have to give supportive statements of their decision. We compare how they perform with and without the SENSEI tools. In this scenario, the evaluation could be run in the following way (numbers are placeholders, actual values will be defined by WP1). A set of QA professionals are given 20 conversations each, from a single agent. The time limit for filing the AOF is 20% of the total duration of the conversations. A first subgroup, the control group, has access to current technology: an audio player for each conversation, the corresponding transcript with a text search function. The second group has access to the same functionalities as the first group, plus prefilled AOFs at the agent level and conversation level, synopses of the conversations for better browsing, advanced search including semantics, argumentative structure and emotions. After the time is elapsed, their forms are compared to reference forms carefully filled by experts.

The second scenario is “collection-wide conversation retrieval,” a task relying on conversation summaries and ad-hoc reports. Subjects perform a collection-level information retrieval task. We compare how well they perform with and without the SENSEI tools. This scenario could be run as follows. Subjects are given an information need: for instance “find conversations where the caller is asking for an itinerary.” Then, they have a limited time (10 minutes, for instance) to

find all conversations in the collection which a relevant to the query, and explain in their own words why they are relevant. The control group is given each conversation with an audio player, the transcripts and a search function. The other group is given the SENSEI conversation browser which includes advanced search and display of the conversations. After the time is elapsed, subjects are rated according to the number of relevant conversations they have been able to retrieve, compared to expert-labelled conversations.

1.1.2 Social media scenarios

The social-media use case consists in supporting journalists who post articles online, as well as comment readers and writers (both called commenters) with tools for exploiting comment threads. Journalists may want to gather reader testimonies from the comments, direct readers towards insightful comments, and simply connect with their readers. Commenters may want to have all the elements for posting constructive comments, that is know what is being talked about in the thread, know who the other commenters are, etc. The two use cases selected in D1.1 are "Town Hall Summaries (THM)," summaries of what is talked about in comment threads, by whom, what are the arguments and opinions, and "identifying trends in readers' comments", identifying topics with a target volume and polarized opinions. Again, the extrinsic evaluation will be carried out in this framework.

The data for supporting the extrinsic evaluation is a set of news articles with accompanying comment threads, collected by WP2. The articles and comments have been extracted from the website of the Guardian (see D2.2). Three scenarios for ecological evaluation have been selected. More details about the scenarios can be found in D1.2.

The first scenario is "A comment editor preparing a summary of the contents of a news article and associated comments," a task relying on conversation summaries. In this scenario, subjects (comment editors) have to write an article which summaries the comments to a given news article, selected for its particularly interesting commenters' conversation. Under a limited time (i.e. 20 minutes), subjects have to write a town-hall summary of the comment threads, answering questions such as "What's the lead? Who took part? What were the people talking about? What issues did people feel strongly about? What issues did people agree/disagree about?" The control group is given the article, the comments (as presented on the Guardian website) and a search function. The other group can use the thread view proposed by SENSEI which shows the argument structure, emotions, and semantic analyses. An advanced search function is available. Subjects are rated in different ways which will be defined by WP1, for instance by comparing their production with gold standard town hall summaries, created without a time limit. Subjects are also asked to fill a post-hoc questionnaire.

The second scenario consists in "A Comment Editor selecting 'editor picks' from a set of comments," a task using conversation summaries, blogger-oriented summaries as well as ad-hoc reports. In this scenario subjects have to trawl through comment threads and retrieve high quality comments that would have been selected as "editor picks" for their particularly useful content. The control group performs the task with a user interface similar to the current Guardian website (found in Appendix B), including a text search function. The other group has access to the SENSEI thread view, and commenter details, as well as advanced search. Subjects are rated

according to various metrics, such as the number of gold-standard editor picks they can retrieve in a limited time. Subjects are also asked to fill a post-hoc questionnaire.

The last scenario is “A comment provider writing a new comment,” a task relying on ad-hoc reports and conversation summaries. In this scenario, comment providers are given a news article and comment threads. Under a time constraint, they are required to read the threads and decide or not to create a new thread, and/or post a reply to a comment in an existing thread. The control group has access to a baseline threaded view (for example the current Guardian website), while the test group has access to SENSEI technology such as advanced search, and conversation structuring. Subjects are evaluated on the quality of their production, as well as asked to fill post-hoc questionnaires on their understanding of the content of the comments.

1.1.3 Functional analysis

After describing both the social-media and speech scenarios, it is possible to outline functionalities that will be common to both scenarios and functionalities which need to be scenario-specific. We can define a list of views and the corresponding functionality:

- Collection browser: subjects shall be able to browse the list of evaluation-targeted conversations. This view might benefit from conversation summaries for locating quickly relevant information.
- Conversation browser: for a single conversation, the subjects should be able to look at the transcript/text, as well as listen to the audio. This view shall involve conversation summaries.
- Agent/commenter: allows displaying aggregate information about an agent or commenter. This view shall involve blogger-oriented summaries as well as rated questionnaires, depending on the use case.
- Basic text search: allows retrieving a list of conversations (or items from conversations such as speech turns, single comments).
- Advanced search: allows retrieving conversations according to more complex criteria, fed by SENSEI technology. This view will be central to ad-hoc reports.
- Evaluation input: allows subjects to complete the scenario task by filling in evaluation fields, as well as explaining their motivation for doing so.

2. Requirements

In the following, the high-level requirements for the prototype specification are given as guidelines for the choices made in the rest of the document.

2.1 Organization

Software development and processes should not incur a demanding overhead and should divert as little resources as possible from the research aspects of the project. Software components supporting the various evaluation scenarios should be factorized as much as possible, specifically to draw from the similarity between the speech and social media use cases.

2.2 Usability

The prototype shall be easy to use for the targeted evaluation subjects. In particular, it should rely on proven, well accepted technologies such as web sites. In order for the evaluation to measure a technological difference and not a usability difference, great care should be taken to create intuitive, ergonomic and instrumentable user interfaces. The user interfaces should be appealing, in order to motivate evaluation subjects and serve demonstration purposes. They should be integrated in a single, coherent package, and follow the same style guide.

2.3 Software

Development should prioritize the use of open-source, well supported software. Programming languages and libraries shall be widespread, highly documented, well accepted and reasonably efficient.

2.4 Hardware

The prototype shall not depend on specific hardware. In particular, it should be hostable on virtual hardware in order to be able to replicate as closely as possible the development conditions on all sites of the project partners. A central platform for running the prototype shall be made available by one of the partners.

2.5 Latency

Prototype components shall not have latency higher than what is accepted by professionals targeted by the evaluation. In particular, components shall be divided into offline modules which are run beforehand and the result of which is cached, and online modules which are responsive to user input.

2.6 Security

The prototype shall not allow access to unauthorized individuals. Categories of users should be allowed to act according to their capabilities (such as viewing, modifying, parameterizing). All accesses to the system shall be logged.

2.7 Quality

Source code as well as data shall be versioned. Version tags should be used to denote compatible modules and data. Under reasonable development overhead, components shall be unit-tested, in particular at the interface level between components (for instance the server API).

3. Development process

This section describes the development process adopted for the prototype.

3.1 Development practices

Developers shall write well documented, easy to understand and formatted code, with good naming conventions. They should focus on providing stable interfaces to the modules they have in charge. They should test their code before sharing it with other partners.

3.2 Source control

All source code and data produced for the prototype shall be versioned in order to ease the collaboration process and track changes during the project.

As outlined in D2.1, SENSEI unprocessed data is versioned under subversion, hosted at Websays.

The prototype source code and data is versioned in a git repository, available at https://gitlab.lif.univ-mrs.fr/benoit.favre/sensei-proto. Access requires a user account and password, available from the WP6 leader. Gitlab is an open source development collaboration suite which provides git repository management, code reviewing, issue tracking, activity tracking and documentation through wikis. In addition, gitlab can be interfaced with gitlab-ci, a continuous integration server which tracks the success of build and test suites after each git commit.

All developers working on the SENSEI prototype should get push rights to the gitlab central repository and deliver their software this way. They should follow git good practices, as described in http://sethrobertson.github.io/GitBestPractices, which include using meaningful commit messages and not publishing changes that break the running version of the prototype.

3.3 Deployment

In order to help developers work on the prototype, a reference deployment platform has been defined. This definition includes the version of the operating system, libraries and programming languages the developers can use. For all dependencies, only one version is preferred in order to increase the coherence and reduce the friction of the development process. All developers shall use the reference deployment platform for testing their code prior to sharing it with other partners. The platform consists in[1]:

- Operating system: Ubuntu 14.04
- Programming languages: Python 2.7, Python 3, PHP 5.3.2, Java 7, Perl 5, C/C++ with gcc/g++
- Databases: MySQL, MongoDB
- Web server: Apache2 with mod_rewrite, Apache Tomcat 6.0.39
- Web browser: Google Chrome / Chromium browser
- Web frameworks: HTML5, jQuery, AngularJS, Bootstrap, HighCharts.

[1] A full description of the installed packages can be found in https://gitlab.lif.univ-mrs.fr/benoit.favre/sensei-proto/blob/master/deploy.sh (access requires developer credentials).

In addition to defining the reference deployment platform, project members can access a virtual machine (for VirtualBox or VMWare) preinstalled with the reference deployment platform software, to be used as server for local tests. The VM contains a 200GB hard drive, 2GB of memory and is available as a ~5GB download. The content of the VM can be updated from the sensei-proto git repository using a provided script. Appendix A gives quick start documentation to using the VM for developers.

In addition to the VM, a reference deployment server runs at AMU (139.124.22.35 / sensei-proto.lif.univ-mrs.fr). Every time a developer pushes changes to the git repository, the machine will run the update script so that it hosts the latest version of the prototype. It will only be accessible to authorized project members. For security reasons, installing new software on the reference server will require manual intervention. A frozen, evaluation-specific deployment will be used during evaluations.

3.4 Quality insurance

Having a centralized and synchronized development process already ensures the coherence of the developed software modules. In addition, a gitlab-ci continuous integration server is being run to provide developers with amenities for automatically testing their code after each commit. Even though unit testing is encouraged, specifically at the interface between modules, it is not mandatory in order not to hamper developer productivity.

4. Detailed design

4.1 Guidelines

The detailed design contains high-level description of the components necessary for the prototype. The descriptions focus on functionalities instead of implementation details, which we need to keep flexible in order to foster interaction opportunities between research and development.

4.2 Architecture

The prototype will be a web service with a centralized server running databases and serving files, as well as a client side written in HTML and making use of modern browser features for intuitive and swift user interactions. The user interface views are run client-side while the backend modules are run server-side.

4.3 Backend modules

Backend modules run on the server. For maximum flexibility, they are written in PHP, Python, Java, Perl or C/C++ and communicate through representational state transfer (REST) interfaces.

Arbitration between the modules is performed by a Gateway module. The conversation repository provides access to data, the synopsis generator, AOF generator, etc provide SENSEI-powered high-level conversation analyses, the evaluation recorder module stores the data captured for the evaluation. Other modules feed the different components of the user interface.

In the following, so-called “Offline” modules might run for a long time and are therefore fed in an offline manner with precomputed resources or run beforehand to generate those resources (they are not subject to latency constraints). These resources are cached in the conversation repository. “Online” modules are subject to user input and compute their output on the fly under the latency constraint.

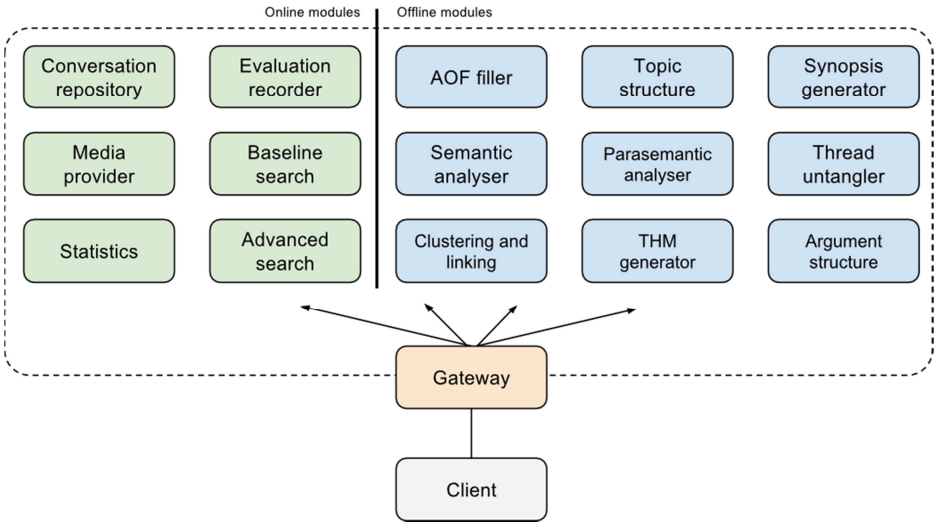


Figure 1: Server-side modules. Blue modules are run offline, green modules are online, the gateway mediates the access to other components.

The REST APIs of the prototype shall be stateless, cacheable, make use of uniform naming (use nouns instead of verbs), be implemented as HTTP queries using common error codes and reply as JSON, XML or CSV (modules should at least implement JSON). Appendix C lists tutorials for implementing REST services in various programming languages. The gateway is the only server exposed to clients, it redirect queries to other modules. For instance, the following URL:

https://sensei-proto/synopsis-generator/synopses/20091112_RATP_SCD_0020

is forwarded to the synopsis generation module running on the same server at the URL:

http://localhost:port/synopses/20091112_RATP_SCD_0020.

Note that only the gateway uses https.

In the rest of the document, except for the gateway module which is exposed to the outside world, module APIs will be given in the form http://localhost:port/ where the module name does not appear.

The REST protocol is defined for all server-side modules as follows. The definition includes for each service, an HTTP action (GET, POST, PUT,...), a URL, a definition of the action, the format of the output, the list of possible errors and associated error codes as well as associated content. In particular, the following conventions shall be used.

HTTP verbs:

- GET: retrieve a resource
- POST: create a resource
- PUT: update a resource
- DELETE: remove a resource

HTTP error codes:

- 200 Success (when a query is successful), 201 Created (when a resource had to be created)
- 304 Not modified (when a resource did not need to be created because it already existed)
- 400 Bad request (when the parameters of a query don't make sense), 401 Unauthorized (when credentials do not permit the query), 404 Not found (when a resource does not exist)
- 500 Internal server error (when a service yielded an exception), 501 Not implemented (when a service is not yet implemented)

Query parameters:

- approach: specify a method for generating the resource
- from, to: specify a range
- format: xml, json, csv (for modules that support multiple formats)

JSON output in case of error:

{success: false, error: "message", code: error-code}

JSON output in case of success:

{success: true, result: {content}, code: error-code}

Authentication and credentials are managed by the gateway module (described thereafter), other modules (1) do not listen for connections outside of localhost, and therefore (2) don't have to deal with credentials.

Example of API URLs:

GET http://localhost:port/documents (retrieve full list of documents)

GET http://localhost:port/documents/1 (retrieve first document)

GET http://localhost:port/approaches (retrieve list of valid approaches)

GET http://localhost:port/parameters (retrieve list of available parameters)

For each module, the specification document lists the behaviour of the module and its API. Note that the API is not binding and might be changed to accommodate for evolutions during the project. It will be fully specified in D6.2.

4.3.1 Gateway

This modules arbitrates the communication between the different modules. It also serves the files for the web interface. It is an https service exposed to the outside, so it also makes sure that clients are properly authentified. URLs of the form:

GET https://sensei-proto/<module-name>/<resource>/<id>...

are forwarded to the corresponding module (given adequate credentials). Other modules than the gateway listen in http on the local network (for instance localhost) on non-privileged port numbers, possibly in the range 8000-8100. The mapping between host:port and module names is defined in a configuration file, loaded by the gateway. For convenience, the gateway also listens on localhost for other modules' use:

GET https://sensei-proto/<module-name>/<resource>/<id> is rewritten as

GET http://localhost:port/<resource>/<id> with the port of the corresponding module.

In addition, the gateway hosts the login service which given a user/password combination returns an identification token which must be used to access other resources.

POST https://sensei-proto/login/<user>

with the password as POST parameter. This URL returns an identification token which must be passed to identify all subsequent communications. Any other query without the authentication token is redirected to the login UI.

This module is an online module.

4.3.2 Conversation repository

This module enables other modules to store, annotate, and access the content of structure conversations. It is fully described in D5.1. It shall respond to various queries (see D5.1) including the following fundamental ones:

GET http://localhost:port/documents lists all document ids.

GET/PUT/DELETE http://localhost:port/document/<doc_id> returns the content of a document, updates it, or deletes it.

POST http://localhost:port/document stores a new document and returns its unique id (generated by the repository).

POST http://localhost:port/document/content stores the textual content of a document, returns its id.

POST http://localhost:port/annotations/<doc_id> adds annotations to a document.

POST http://localhost:port/features/<doc_id> stores features related to a document.

DELETE http://localhost:port/annotations/<doc_id>/<set_name>/<ann_id> deletes a set of annotations from document

DELETE http://localhost:port/annotations/<doc_id>/<set_name>/<ann_id> deletes specific annotations from document.

DELETE http://localhost:port/features/<doc_id>/<feature_name> deletes a feature from the document.

GET http://localhoost:port/documents?<feature0>=<value0>&<feature1>=<value1>... returns a list of documents that have feature specified.

POST http://localhost:port/documents returns a list of identifiers of documents that have feature-value pairs specified as POST data.

A document is an object consisting of content text, a map of document features, and sets of annotations, generally represented as JSON. D5.1 fully explains the structure of a document.

This module is an online module.

1.4.3.3. Synopsis generator

The synopsis generator module encapsulates the synopsis generation approaches developed in WP5 and can generate synopses for each of the spoken conversation. The API should allow to select the type of synopsis (text, list of descriptive sentences attributed to sets of speech turns, semantic representation) as well as the approach for generating it (baseline, MMR, etc). It should allow the offline generation of synopses and store them in the conversation repository. The module shall reply to the following query:

GET http://localhost:port/synopses/<conversation-id>?type=<type>&approach=<approach> returns the synopsis for a given conversation, for a given type and approach. The default type is a textual synopsis; the default approach will be determined in WP5.

GET http://localhost:port/approaches lists valid approaches.

This module is an offline module.

4.3.4 THM Generator

This module generates structured town-hall-meeting summaries for supporting the social media use case evaluation. It will be developed by WP5. It supports the following API:

GET http://localhost:port/summary/<document-id>?type=<type>&approach=<appraoch>

Generates a properly structured summary for a given social-media conversation (news article and comments).

GET http://localhost:port/approaches lists valid approaches.

4.3.5 Agent observation form filler

The AOF filler processes spoken conversations in order to reply the questions of the agent observation form. This module encapsulates the technology developed in WP3, WP4 and WP5 for conversation analysis. For each question, it should return a score for each of the possible answer, the highest score being the decision of the module. For each question, it should be possible to select the approach for generating the decision. It should also allow the offline generation of AOF answer. The module supplies the following queries:

GET http://localhost:port/forms/<conversation-id>/<question-id>?approach=<approach> returns the vector of (answer, score) couples for a given question. The default approach will be defined by WP3.

GET http://localhost:port/questions returns the list of question ids.

This module is an offline module.

4.3.6 Topic structure generator

This module encapsulates WP3 approaches which extract the topical structure of conversations. For each conversation, it generates a segmentation in topics as well as topic labels, allowing to select the approach being used. It allows offline processing. The module provides the following queries:

GET http://localhost:port/topics/<conversation-id>?approach=<approach> returns the topic structure for a given conversation. The default approach will be defined by WP3.

This module is an offline module.

4.3.7 Argument structure generator

This module generates the argumentative structure of a conversation for both use cases. It will be developed in the course of WP4 and follows this API:

GET http://localhost:port/arguments/<conversation-id>?approach=<approach>

GET http://localhost:port/coreferences/<conversation-id>?approach=<approach>

The first method returns the argument structure while the second one returns the coreference pairs of the document.

This module is an offline module.

4.3.8 Thread untangler

This module inputs complex threaded conversations such as those found in social media and returns untangled conversations according to various criteria such as topic, commenters, emotions, etc. It may precompute potential thread boundaries. It encapsulates approaches developed in WP3 and WP4, and draws specifically from the argument structure. The module provides the following queries:

GET http://localhost:port/subthreads/<conversation-id>?approach=<approach>&criterion=<criterion> returns the subthreads of a conversation according to given criteria. The default approach and criteria will be defined in WP4.

This module is an online module.

4.3.9 Semantic analyser

This module encapsulates the semantic analysis approaches developed in WP3. It inputs conversations and outputs semantic structures. The module provides the following API:

GET http://localhost:port/semantics/<conversation-id>?approach=<approach> returns the semantic analysis of a conversation.

Approaches and output format will be defined in WP3.

This module is an offline module.

4.3.10 Parasemantic analyser

This module encapsulates the parasemantic analysis approaches developed in WP3. It inputs conversations and outputs parasemantic classes and analyses. The module provides the following API:

GET http://localhost:port/parasemantics/<conversation-id>?approach=<approach> returns the parasemantic analysis of a conversation.

Approaches and output format will be defined in WP3.

This module is an offline module.

4.3.11 Clustering and linking module

This module provides clustering of comments/speech turns into topically coherent sets. In addition, it provides comment/speech turn linking. This module will be provided by WP5. It follows this API:

GET http://localhost:port/clusters/<document-id> returns a list of clusters for the given document id.

GET http://localhost:port/links/<utterance-id> returns a list of utterances linked to the given utterance, with link labels.

This module is an offline module.

4.3.12 Baseline indexing and search

This module provides baseline word search from conversation transcripts. It can be implemented with SOLR. The module implements the following API:

GET http://localhost:port/search/<query>?from=<from>&to=<to> returns the ordered set of conversations that match a search query, optionally from the <from> to the <to> result. By default, returns 20 results. A query is a sequence of words separated by spaces.

This module is an online module.

4.3.13 Advanced Indexing and search

This module provides instant search from conversation transcripts, structural, semantic and parasemantic tags. It can be implemented with SOLR. The module implements the following API:

GET http://localhost:port/search/<query>?from=<from>&to=<to> returns the ordered set of conversations that match a search query, optionally from the <from> to the <to> result. By default, returns 20 results. A query is a sequence of words, structural, semantic and parasemantic tags.

This module is an online module.

4.3.14 Statistics

This module provides the ability to extract various statistics from conversations, such as the distribution of conversation length against emotions and other multivariate analyses. It is fed with a set of variable names and returns statistics and histograms. The module implements the following API:

GET http://localhost:port/multivariates/<variable1>&<variable2>&<variable3…> returns the statistics for a set of variables.

GET http://localhost:port/variables returns the list of valid variable names.

GET http://localhost:port/histogram?variable=<variable>&bins=<bins> returns the histograms for a variable, in a number of bins.

This module is an online module.

4.3.15 Media provider

This module provides access to audio files. In its basic implementation, it is just an HTTP file server, queried from the browser through HTML5 audio. The API of this module is as follows:

GET http://localhost:port/media/<conversation-id> returns an mp3-encoded audio file for a given conversation. This module supports HTTP media streaming (can be implement through Apache).

This module is an online module.

4.3.16 Evaluation recorder

The evaluation recorder is the module in charge of logging all evaluation data, including nature and timing of interactions, and evaluation form data. It can be queried in order to display statistics and evaluation outcome. This module provides the following API:

PUT http://localhost:port/evaluations/<conversation-id>?user=<user>&condition=<condition>&field=<field>&value=<value> records an evaluation parameter.

GET http://localhost:port/evaluations/<conversation-id> retrieves the outcome of the evaluation.

The above URLs will be redesigned to accommodate for the specifics of the use cases.

This module is an online module.

4.4. User interfaces

The user interface consists in a number of views which can be used by subjects in order to perform the task given to them. All views shall be instrumented in order to record the timing of user input such as clicks and text entry. User interfaces are implemented as a web client using HTML5, jQuery, AngularJS, Bootstrap.

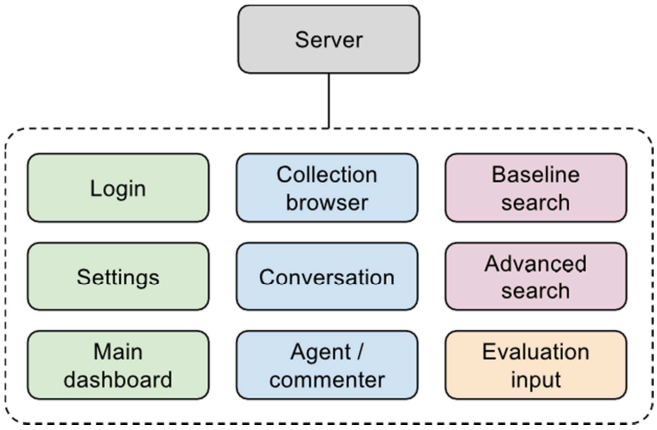


Figure 2: client side views

4.4.1 Login

This view allows the user to enter appropriate credentials. Users can have three roles: administrator, evaluator, and regular. Interactions from evaluator are logged and they have access to evaluation input forms. Administrator can add new users and change roles. Regular users can only browser the content of the conversation collection.

4.4.2 Main dashboard

This view shows an overview of the possible interactions, database-level statistics, as well as instructions for the evaluation subjects.

4.4.3 Collection browser

This view shall show the content of the collection. It shows conversations, agents/commenters and global statistics. The list of conversations and the list of agent/commenters can be organized by various criteria, such as topics, argument structure, semantics, parasemantics. For each conversation, it shows a summary and factual descriptors. For each agent/commenter, it shows factual descriptors. Global statistics can be histograms and curves and support use cases (for instance, average time of a call, number of replies to a news article…) This view can be used to generate ad-hoc reports, and takes advantage of conversation summaries and blogger-oriented summaries for qualifying contents of the collection.

Selecting a conversation or agent/commenter leads to the corresponding view. This view can be restricted to baseline display for supporting the control group evaluation.

This view is supported by the following backend modules: conversation repository, statistics, topic structure, synopsis and THM generators, semantic, parasemantic, and argument structure generators, and clustering and linking.

4.4.4 Conversation view

The conversation view shall display a single call centre conversation or comment threads from a news article. The conversation is structured by topic, argument, semantics, parasemantics in order to help quickly browsing to relevant information. It is supported by conversation summaries, such as THM or synopses. Annotations can be selected for searching other conversations

or displaying them in other views. For audio conversations, a synchronized player is available. For long conversations such as in social media, subthreads can be untangled. The view can be used for finding similar / redundant conversations.

This view can be restricted to baseline display (transcript, comment threads) for supporting the control group evaluation.

This view is supported by the following backend modules: conversation repository, media provider, topic structure, synopsis and THM generators, semantic, parasemantic, and argument structure generators, and clustering and linking.

4.4.5 Agent / commenter view

This view shall display details about an agent or commenter: factual information, statistics along semantic, structural, parasemantic, topical axes, list of contributions, agent observation form, friends and foes, emotion flowers, etc. This view embodies blogger-oriented summaries in the social-media use case, as well as rated questionnaires in the speech use case.

This view can be restricted to baseline display (factual information and contributions) in order to support the control group evaluation.

This view is supported by the following backend modules: conversation repository, statistics, topic structure, semantic, parasemantic, and argument structure generators, and clustering and linking.

4.4.6 Baseline search view

The baseline search view provides text search on transcripts / comment threads. It implements instant search suggestions (display results as soon as text is typed), and displays results as a list of passages ranked by relevance. Each passage is qualified by extracted snippets showing the matches. This view essentially mimics a web search engine as can be found on many websites.

This view relies on the conversation repository, and baseline search. It could be implemented using the SOLR search engine.

4.4.7 Advanced search view

This view shall allow searching the collection by keywords, and all other annotations. It implements instant suggestions, and recognizes topics, argument structure, semantic and parasemantic elements. The results are displayed as a list showing a summary of the conversation, relevant snippets or keywords, highlighting found items. It also shows factual descriptors (such as duration, concepts). Search results are ordered by relevance. The search results displayed in this view are at the heat of ad-hoc reports.

This view is supported by the conversation repository, advanced search, topic structure, synopsis generation, semantic analyser, parasemantic analyser, thread untangler, clustering and linking, THM generator and argument structure extractor.

4.4.8 Evaluation input

This view allows subjects to input the outcome of their task. Given different constraints of the evaluation scenarios for each use case, this view provides adequate controls for inputting subject production: text fields for writing summaries, forms for filling AOFs, etc. The AOF filling view is already developed as part of WP2, and will be integrated for evaluation input. This view cannot be accessed by regular users. For evaluators, it is always available besides other views, so that they can enter their production at any time.

This view relies on the evaluation recorder.

4.4.9 Settings

This view shows the settings of the interface. In particular, evaluation scenarios and modalities can be selected, users can be managed, and evaluation reports can be viewed. This view enables administrators to create evaluation scenarios: select how an evaluator will perform tasks under a set of constraints, using which collection and in which order.

5. Conclusion

This document presents the specification of the prototype which will be used for running the extrinsic evaluation in the SENSEI project. The scenarios are AOF filling, spoken conversation retrieval (for the speech use case), THM summary writing, and staff picks selection (for the social media use case). The prototype will be a website where subjects can perform the task using SENSEI technology while a control group uses baseline features. The service side is composed of REST services which expose modules from core technology WPs and an HTML5 user interface for running the scenarios. Development guidelines recommend that adequate tools are used for quality assurance and to maximize the flexibility of the design while minimizing the drag on scientific work.

5.1. Roadmap

The next steps include the writing of the detailed design of the extrinsic evaluation in D1.3, which will allow finalizing the prototype design document for D6.2. The responsibilities for the development of the prototype are the following:

- WP3 will develop the semantic and parasemantic modules.
- WP4 will develop the argumentative structure module.
- WP5 will develop the summarization modules, the clustering and topic extraction modules, the conversation repository (already described in D5.1).
- WP6 will lead the integration of the modules and UI.

Appendix A: Virtual machine user guide.

This section lists technical information about the virtual machine available for developers of the SENSEI prototype as a reference platform. All modules developed for the prototype should be tested on this virtual machine before being deployed to the actual server.

A.1. Virtual machine description

The VM is a VirtualBox virtual machine. It contains a 200GB HD, 2GB of RAM, with Ubuntu 14.04 server installed.

A.2. Downloading the VM

- The latest version of the virtual machine can be downloaded as an archive from the following URL: http://pageperso.lif.univ-mrs.fr/~benoit.favre/sensei/SENSEI-VM.ova (access limited to SENSEI developers)
    - Note that the extension package from Oracle for USB-2.0 support for VirtualBox should be installed.
    - Installing: import SENSEI-VM.ova in VirtualBox
- Changelog:
    - 20140716: pull to latest git, remove network adapter, add vbox additions
    - 20140520: initial release

A.3. Networking

- Networking uses NAT. If you want to expose a service to the outside, you need to setup port forwarding in the network tab of VirtualBox settings
    - 8080 is forwarded to 8080 (you can open a browser to http://localhost:8080 to hit tomcat).
    - 2222 is forwarded to 22 (ssh -p 2222 sensei@localhost to access the VM through ssh)
- Mount shared directory in VM:
    - add shared folder in VirtualBox UI, remember how you named the folder
    - sudo mount -t vboxsf -ouid=sensei shared-folder-name /mnt/shared-folder
    - For more details, see http://devtidbits.com/2010/03/11/virtualbox-shared-folders-with-ubuntu-server-guest/
- Fix "waiting for network configuration" at boot time: edit /etc/network/interface and comment out the lines about eth1

A.4. How to get an account

Send an email to benoit.favre@lif.univ-mrs.fr with a name and email address. The person will be contacted through that email address. Then, she has to login to https://gitlab.lif.univ-mrs.fr. This will create her user in the gitlab database. After, contact Benoit Favre again in order to get access to the SENSEI-proto git repository. User and password for accessing the VM must be requested to the leader of WP6.

A.5. Installed software

- Development: build-essential make subversion git autotools-dev automake libtool
- Languages: perl openjdk-7-jre g++ python
- Web: lamp-server^ solr-tomcat (version 3.6.2)
- Editors: vim emacs
- Utils: ssh sox htop tmux screen
- Libraries: zlib-bin zlib1g zlib1g-dev zlibc libatlas-base-dev libatlas-dev

A.6. Updating

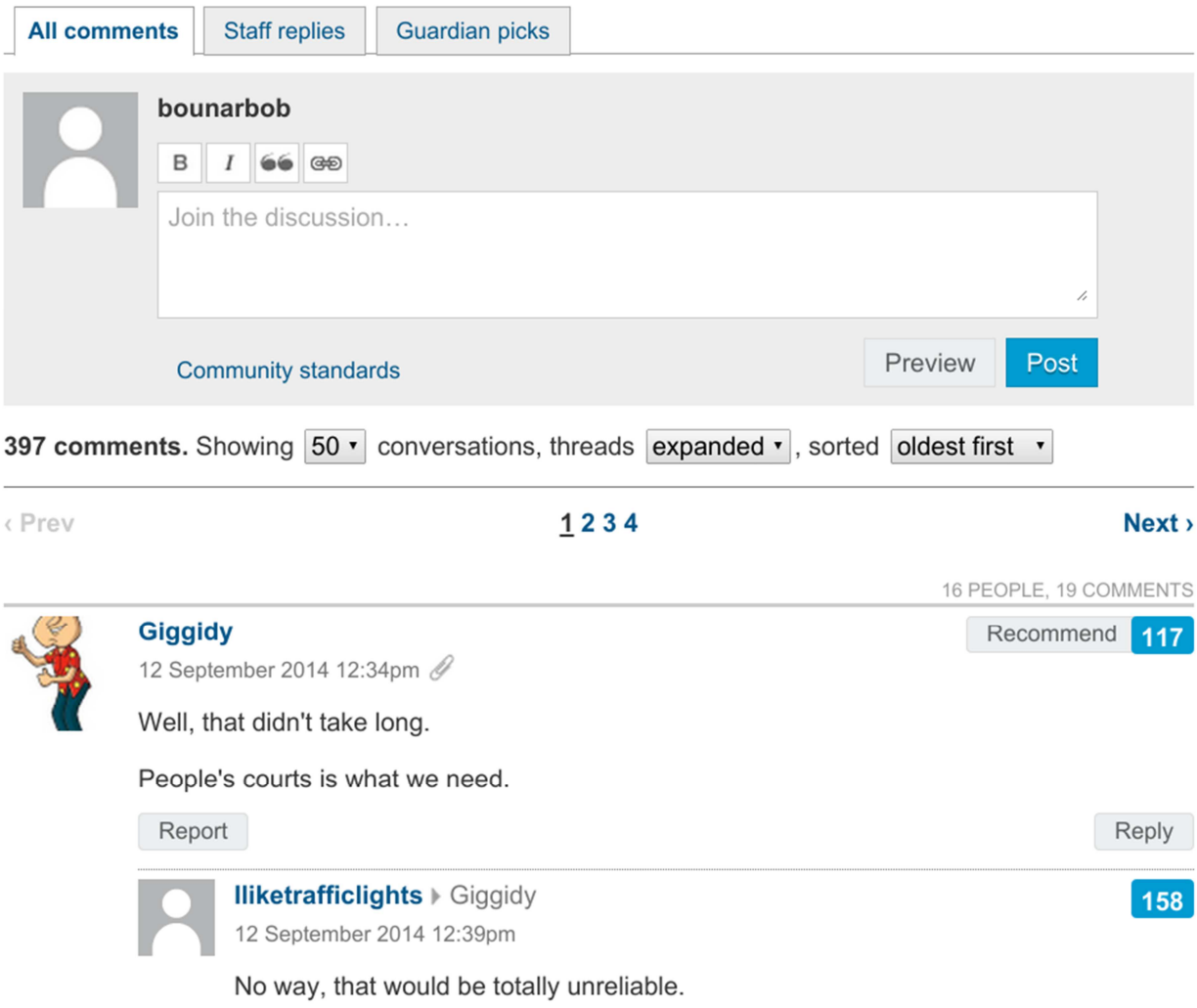
The prototype is stored in the following git repository hosted at AMU: https://gitlab.lif.univ-mrs.fr/benoit.favre/sensei-proto. The repository is cloned in the VM under ./sensei-proto. The VM can be updated by running the ./sensei-proto/update.sh script. This script pulls from the git repository and runs the deploy.sh scripts which installs latest version of software and runs required services.

Appendix B: mock-ups and screenshots

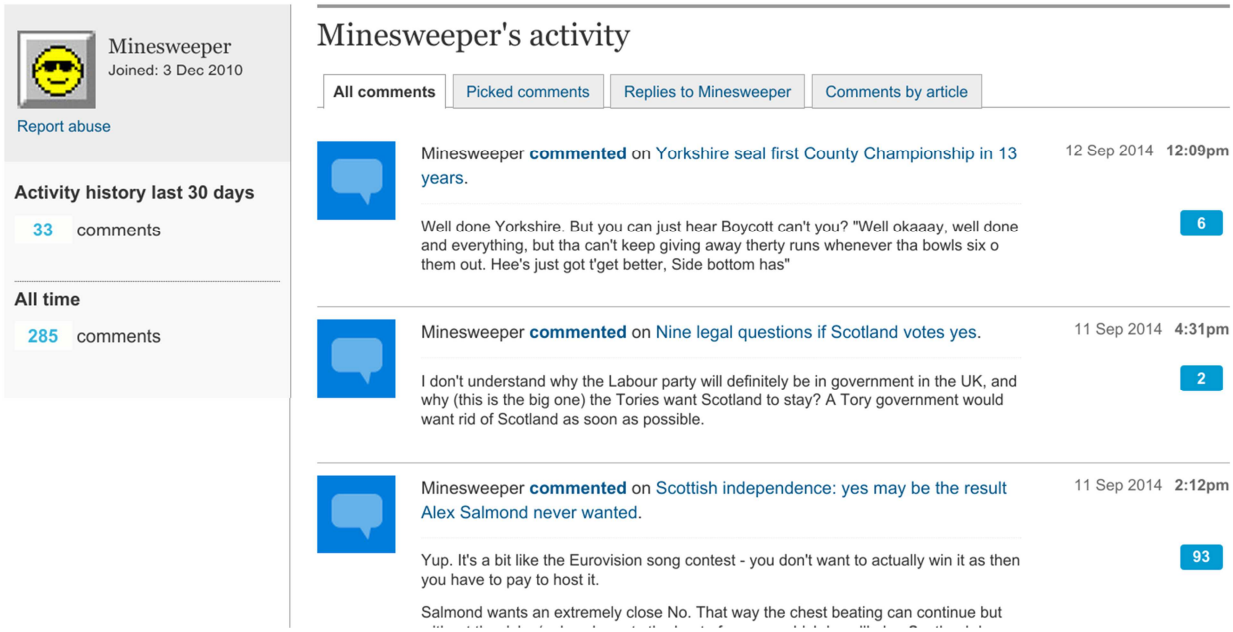
This section lists several early mock-ups of the user interface elements leveraged in the design of the prototype.

B.1. Study of existing interfaces

This is the current comment section of articles on the Guardian website. It shows tabs for selecting subcategories of comments, as well as the number of comments and layout options. For each comment, the author, avatar, date and text are shown. In addition, it is possible to report abuse, recommend comments and reply.



The following screenshot is the commenter oriented view on the Guardian website. It shows the history of a commenter, in categorized tabs.



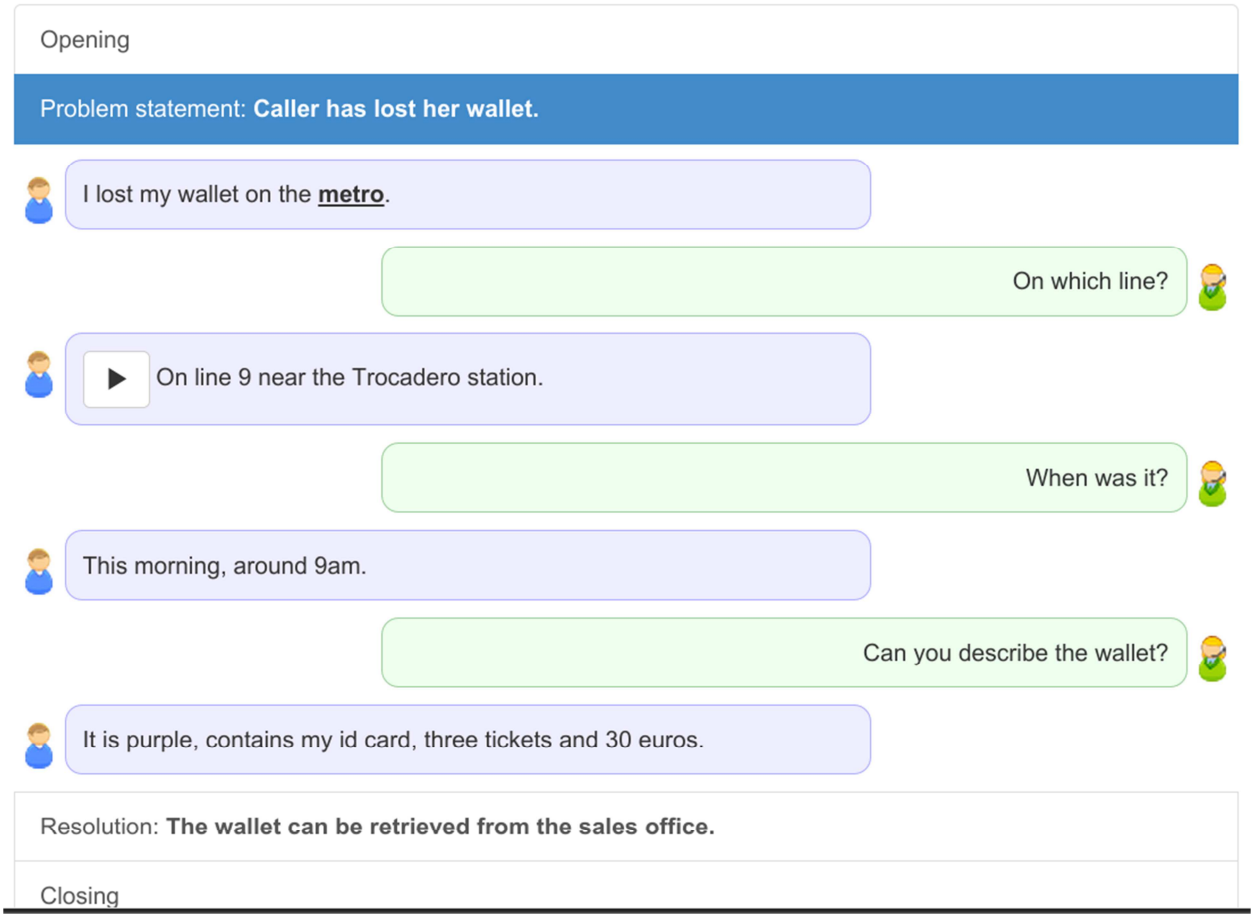
B.2. Collection browser

- schedule
  - **Decoda-345:** The caller asks for the metro schedule changes due to the strike. He complains about the impact of the strike on his work. The agent is sorry but cannot be more accurate.
    Duration: 3m30, 2 speakers
- lost and found
  - **Decoda-177:** The caller has lost his wallet on metro line 5. The wallet can be recovered at the main station.
    Duration: 1m50, 3 speakers
- itinerary
  - **Decoda-921:** The caller requires an itinerary for going from Montparnasse to Invalides without using the metro. The agent tells him to use bus 123.
    Duration: 2m10, 2 speakers

B.3. Conversation view

Decoda-156: The caller has lost his wallet on metro line 9. The wallet can be recovered from the sales office.

lost and found



B.4. Search

Results

1. **Decoda-345:** The caller asks for the metro schedule changes due to the strike. He complains about the impact of the strike on his work. The agent is sorry but cannot be more accurate.
   Duration: 3m30, 2 speakers, negative emotions schedule
2. **Decoda-177:** The caller has lost his wallet on metro line 5. The wallet can be recovered at the main station.
   Duration: 1m50, 3 speakers, lost and found
3. **Decoda-921:** The caller requires an itinerary for going from Montparnasse to Invalides without using the metro. The agent tells him to use bus 123.
   Duration: 2m10, 2 speakers, itinerary

B.5. Auditor observation form

Conversation	Opening	Problem	Solution	Proposals	Closing	Communication
Decoda-221	A1 A2 A3	B1 B2 B3	C1 C2 C3	D1 D2 D3	E1 E2 E3	F1 F2 F3 F4
Decoda-389	A1 A2 A3	B1 B2 B3	C1 C2 C3	D1 D2 D3	E1 E2 E3	F1 F2 F3 F4
Decoda-735	A1 A2 A3	B1 B2 B3	C1 C2 C3	D1 D2 D3	E1 E2 E3	F1 F2 F3 F4

B.6. Semantic concordancer

Conversation	Displacement	From	To	Mean	Goods	Time
Decoda-221	to go	Gare-de-Lyon	Versailles	Bus	I	Today

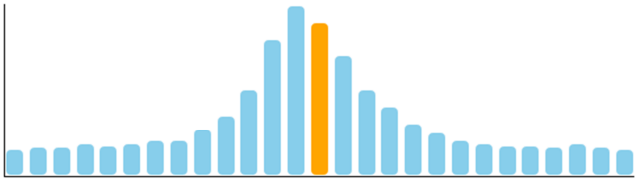
I would like to go from Gare-de-Lyon to Versailles ... The bus would be more convenient ... I need to go today ...

B.7. Statistics

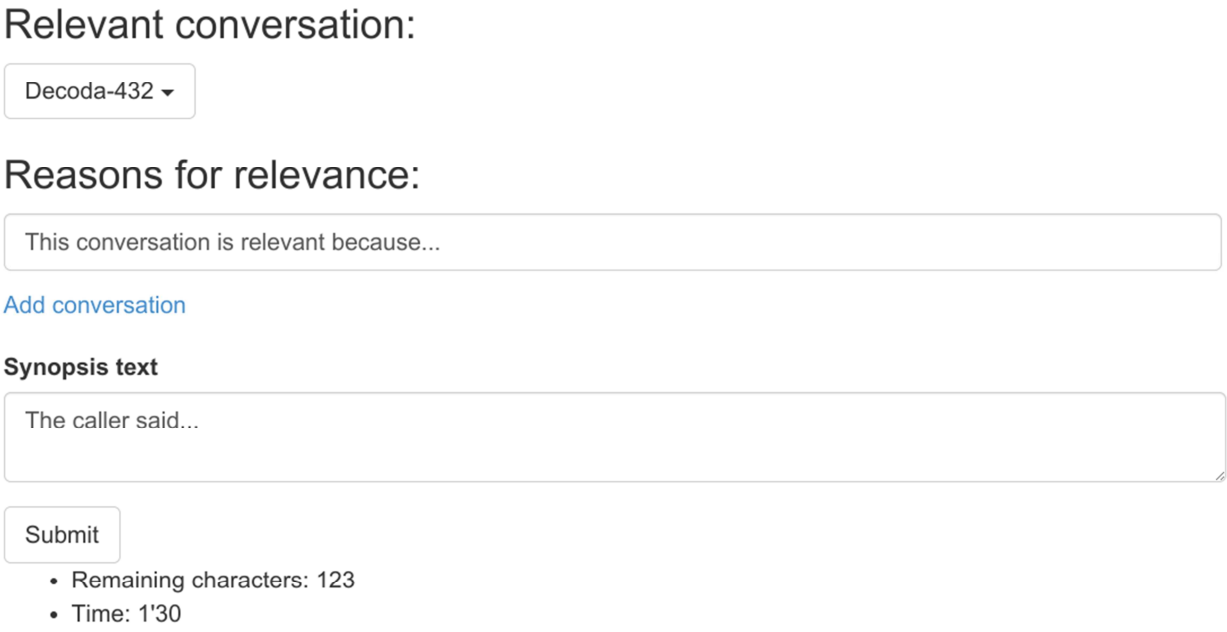
Conversation duration

Emotion intensity

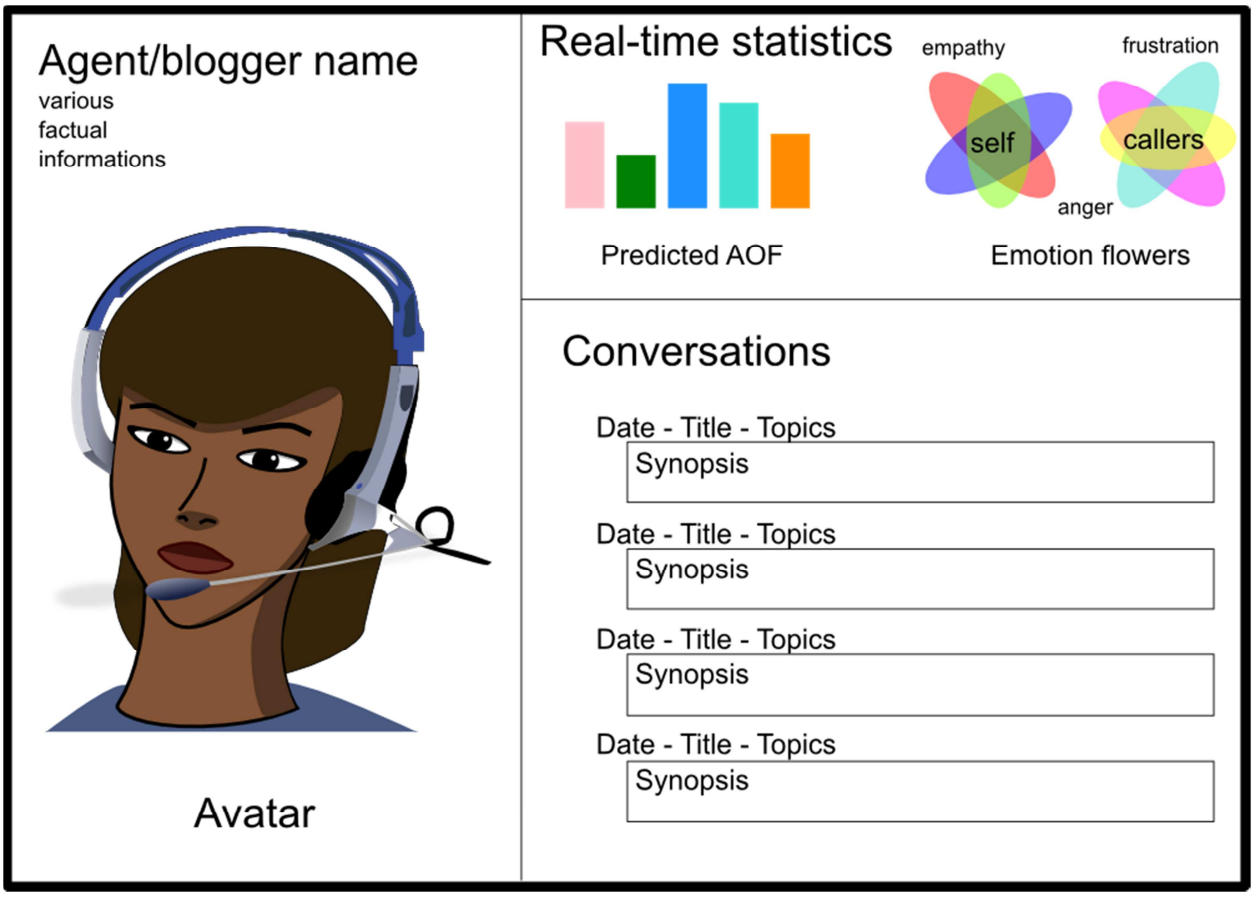
Emotion polarity



B.8. Evaluation input



B.9. Agent/Blogger view



Appendix C: REST/JSON tutorials

- clients in different languages: http://rest.elkstein.org/2008/02/rest-examples-in-different-languages.html
- python server: http://gotofritz.net/blog/weekly-challenge/restful-python-api-bottle/
- JSON encoding of java objects: https://code.google.com/p/google-gson/
- perl server: http://mojolicio.us/
- C++ server and client: http://nipun-linuxtips.blogspot.fr/2012/09/a-simple-rest-framework-on-cc.html
- Java server: http://cxf.apache.org/docs/writing-a-service-with-spring.html