This project is partially funded by the European Commission under the Seventh (FP7-2007-2013) Framework Programme for Research and Technological Development. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

D6.3 SOFTWARE PROTOTYPE V2

Author: Pranay Puri

Affiliation: IWEB

Date: 30 September 2013

Document Number: FIRST_D6.3_20130930

Status/Version: Final / v1.2

Distribution Level: Public

Project Reference	287607
Project Acronym	FIRST
Project Full Title	A Flexible Interactive Reading Support Tool
Distribution Level	Public
Contractual Date of Delivery	30 Sept 2013
Actual Date of Delivery	30 Sept 2013
Document Number	FIRST_D6.3_20131108
Status & Version	Final / v1.2
Number of Pages	20
WP Contributing to the Deliverable	WP2, WP3, WP4, WP6
WP Task responsible	IWEB
Authors	Pranay Puri
Other Contributors	Dilyana Krushkova, Iustin Dornescu, Paloma Moreda, Lea Canales, Isabel Moreno, Elena Lloret
Reviewer	Constantin Orăsan
EC Project Officer	Magdalena Szwochertowska
Keywords:	Software Prototype V2
Abstract:	
This report presents an overview of work done in Open Book Prototype V2 since V1.	

Index

Introduction

This document presents an overview of the Open Book prototype v2, a key second stage deliverable as part of the ongoing Open Book development.

The Open Book application has evolved since the last prototype and now supports features for both standard users and carers. The biggest improvement in this prototype has been around the carer's user interface which now gives carers the ability to re-view the suggestions of the system. They can either use/embed simplified information suggested by NLP services or override the simplified text completely and freely edit the contents of the document. Currently the following types of obstacles are supported by the prototype:

1. Anaphora Resolution related obstacles for English and Spanish
2. Disambiguation of rare or polysemous and Wikipedia disambiguation for some specialized words
3. Offline and Online Image retrieval functions that allow visual depiction of complex concepts and the ability to embed the graphic aide element within the document
4. Dictionary consultation for complex words and substituting appropriate words with explicative text: synonyms and/or definitions
5. Idiom detection
6. Syntactic Simplification (for long sentences)

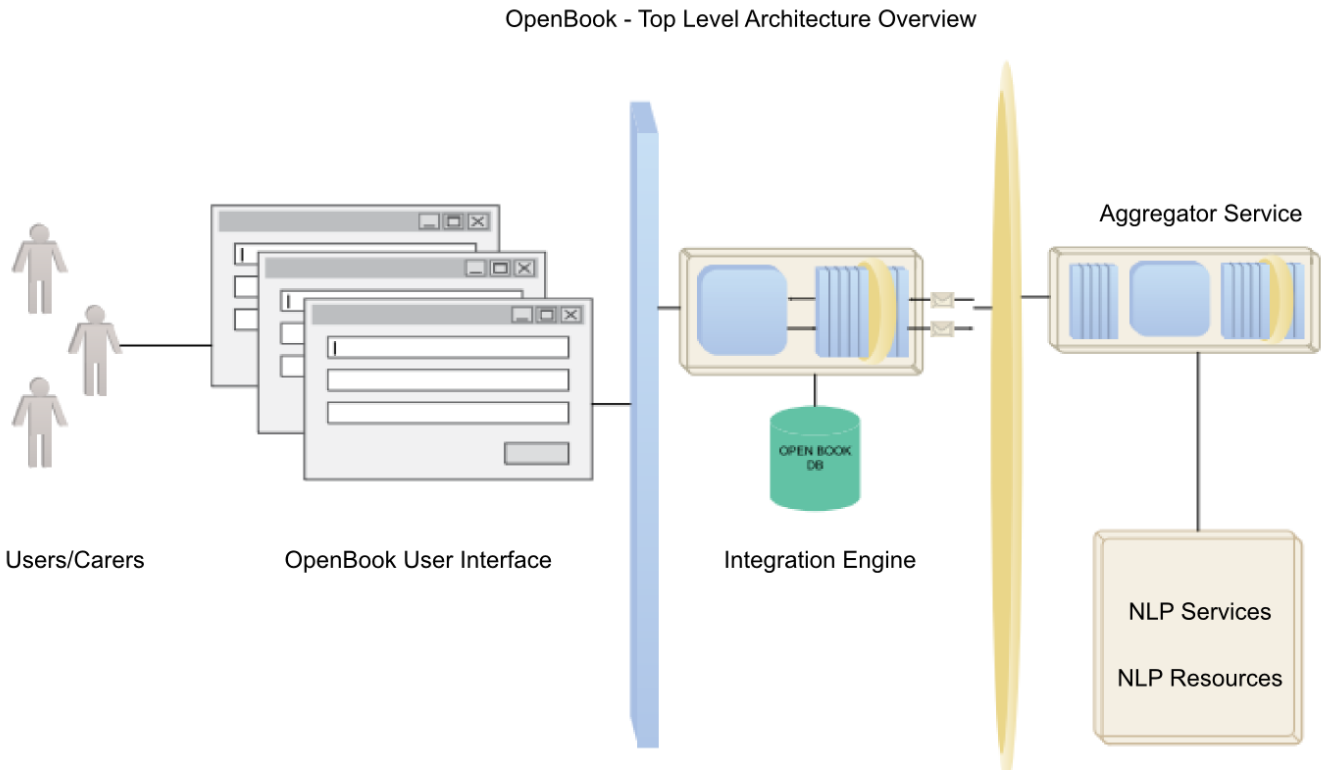
Summary of Changes

The changes that have been made to this prototype have been categorised as follows: architectural and process flow changes, functional changes and changes to NLP Components.

Architectural & Process Flow Changes

Further to an architectural review by all technical members the following changes were proposed and subsequently implemented:

- A new web service façade (Aggregator Web Service) was implemented outside the boundary of the integration engine that accepts request from the integration engine containing text specified in GATE structure format and language code. The façade acts as a single gateway with a sole purpose of calling each NLP web service, aggregating response output containing obstacle specific annotation sets. The reason why this was decoupled from the integration engine is to consolidate all components that naturally fall within NLP and use frameworks already currently used/developed for NLP services that would naturally aide in handling/producing NLP specific technical info quickly and efficiently. Another reason for this change is the situation where different components detect obstacles affecting the same span of text. Correctly resolving any possible conflict requires linguistic knowledge and naturally fits the NLP layer of the system. This allows the Integration Engine Tier to be language agnostic.



- Process flow change in the integration engine to accommodate the new service. The text fragment originally specified by the user/carer is converted into a GATE request that is then passed to the aggregator service. This interacts with back-end NLP services consolidating outputs

from each NLP service into a single response in GATE format back to the integration engine. The integration engine will then convert this final GATE response to HTML in the format the user interface is expecting.

- The integration engine records all metadata about sentence and word entities (including punctuation) along with obstacle information (such as obstacle type and NLP suggestions) in the back-end Open Book database.
- Any look up (for example retrieval of definition, synonym or image info) on a given string (word or sentence) requested by the User or Carer id done on the basis of the entity identifier stored in the HTML document. The User Interface passes the entity identifier of the selected term/phrase and the integration engine will query the database for any obstacle information which is then sent to the User Interface.
- UI acts as a thin client that contains no implementation relevant to any NLP logic; this approach is taken to enable support for low end mobile devices rather than limit the platform to traditional browsers (desktops and laptops).

Functional Changes

The functionality of the user's part of Open Book was enriched with the following features:

- Underline – new button added in the simplification page;
- Bold - new button added in the simplification page;
- Download – provides opportunity to download the simplified text as a plain text (.txt), .pdf file or .html file.
- Text font – user can select amongst the following fonts: Verdana, Tahoma, Calibri, Times New Roman. This functionality was added to the Preferences page.

These features were added to fulfil requirements of ASD users as described in D2.2.

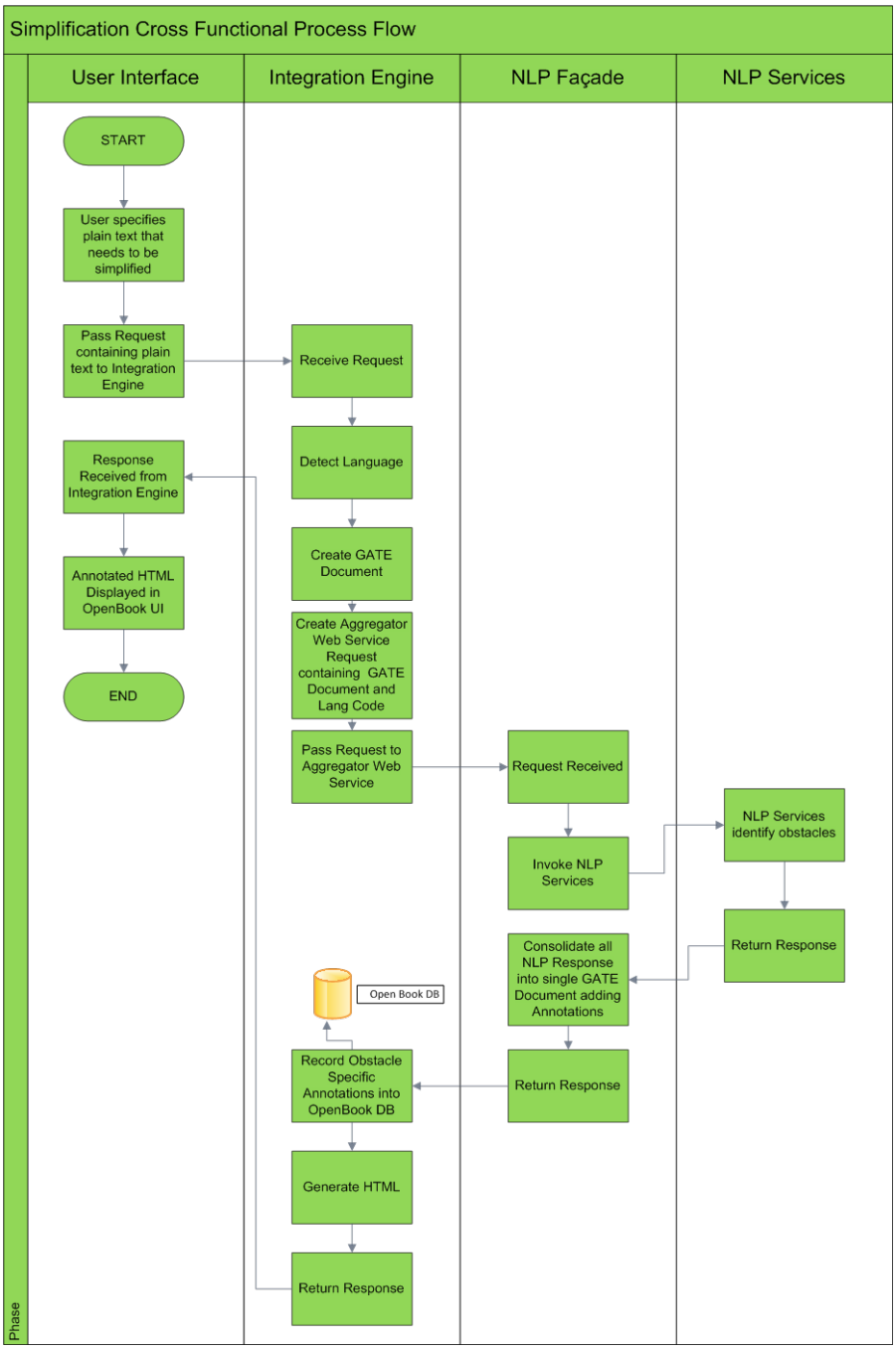
All carer features are introduced for the first time in Open Book v2. The functionality of the Carer's UI includes:

- Administration of users:
  - Register new user;
  - Edit user's profile and preferences;
  - Delete user.
- Notifications – carer can respond to notifications, sent by the user, to sort them by user and by date and to delete notifications;
- Simplify, review and edit documents for a user. Carer can:
  - Review alternative structures of the selected sentence from the simplified text and select the best one;
  - Review the words, considered as obstacles by the system;
  - Replace long/complicated words with synonyms;
  - Insert image to depict the meaning of a word;
  - Insert definitions/hyperlinks for ambiguous terms;
  - Make free changes to the sentence.
- Change his own profile.

Changes to existing features

Simplification

This version of Open Book contains changes to how integration engine accepts information from the User Interface, interacts with NLP resources and then generates output expected by the User Interface. The whole process is illustrated in the diagram below:



The main processing steps (1-6) are listed below.

1) Text Fragment specified, captured and passed to the integration engine

2) Language Detection

This module is responsible for accurately detecting the language of the text fragment passed by the UI. The integration engine uses a text classification library (NtextCat) which uses statistical models built using Wikipedia to detect the language of any given text The library supports UTF-8, UTF-16 and UTF-32 encodings. The module thus detects the most likely language that a text was written in. It returns either English ("en"), Spanish ("es") or Bulgarian ("bg") values.
Note: In order to accurately detect the language code the minimum length of text fragment that can be specified has to be greater or equal to 10 characters. The longer the text, the better the prediction accuracy.

3) Creating GATE document

The next process is to create a GATE document conforming to the following XML schema below where the text fragment is embedded within the TextWithNodes xml node.

```
<?xml version='1.0' encoding='UTF-8'?>
<GateDocument>
<!-- The document's features-->

<GateDocumentFeatures>
<Feature>
  <Name className="java.lang.String">MimeType</Name>
  <Value className="java.lang.String">text/plain</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">gate.SourceURL</Name>
  <Value className="java.lang.String">file:/Users/Eduard/Documents/MyPerlCode/
FirstRelatedScripts/Annotated-GATE-Files/TextFiles/metaphor-1.txt</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">docNewLineType</Name>
  <Value className="java.lang.String">LF</Value>
</Feature>
</GateDocumentFeatures>
<!-- The document content area with serialized nodes -->

<TextWithNodes>{text fragment goes here}</TextWithNodes>
<AnnotationSet>
</AnnotationSet>
</GateDocument>
```

4) Aggregator Service Invocation

The Aggregator web service contains clients for all developed NLP web services. It performed two operations:

- It calls the partner web services and aggregates the output computing the corresponding offsets in the final GATE document.

- It performs tokenization and sentence detection. This information is then used at the level of user interface to properly display the document.

5) The language code along with GATE document containing the text fragment is passed to the aggregator service which is the main NLP façade. The NLP façade invokes each specialized NLP service that is responsible for identifying a specific set of obstacles. Any output from the NLP services which is in GATE structure is consolidated into a single GATE response which is then returned to the integration engine.
6) HTML Generation: once a valid response is returned from the NLP façade service, the integration engine begins the process of recording metadata information for each sentence and word in the sentence starting from the top sentence fragment into the application back-end database. The information that is recorded consists of:

Entity Type, i.e., sentence or word
Entity Unique Identifier: Unique identifier that will be used by the UI to refer to the entity in question
Entity Value: Original Value of the entity specified initially
IsObstacle: Whether the entity has been identified as an obstacle
Obstacle Type: Suggested NLP value (e.g alt sentence, definition, synonym or image url value)
Simplified Value: Simplified equivalent of the Entity Value

Once all relevant information is read from the GATE document and stored in the database, the next process of generating the required HTML output begins.

7) Generating HTML format

This process is a conversion from the GATE format used by the NLP services into HTML.

In order to offer highlighting abilities within the UI to denote where obstacles of different types exist within a text fragment, the integration engine returns an html fragment for each entity type in the form of span tags annotated with `class="sentence"` or `class="word"` along with a unique identifier. In cases where an obstacle has been detected the returned span will contain `class="sentence obstacle"` or `class="word obstacle"` which then instructs the UI to present the entity in question in a different manner aesthetically.

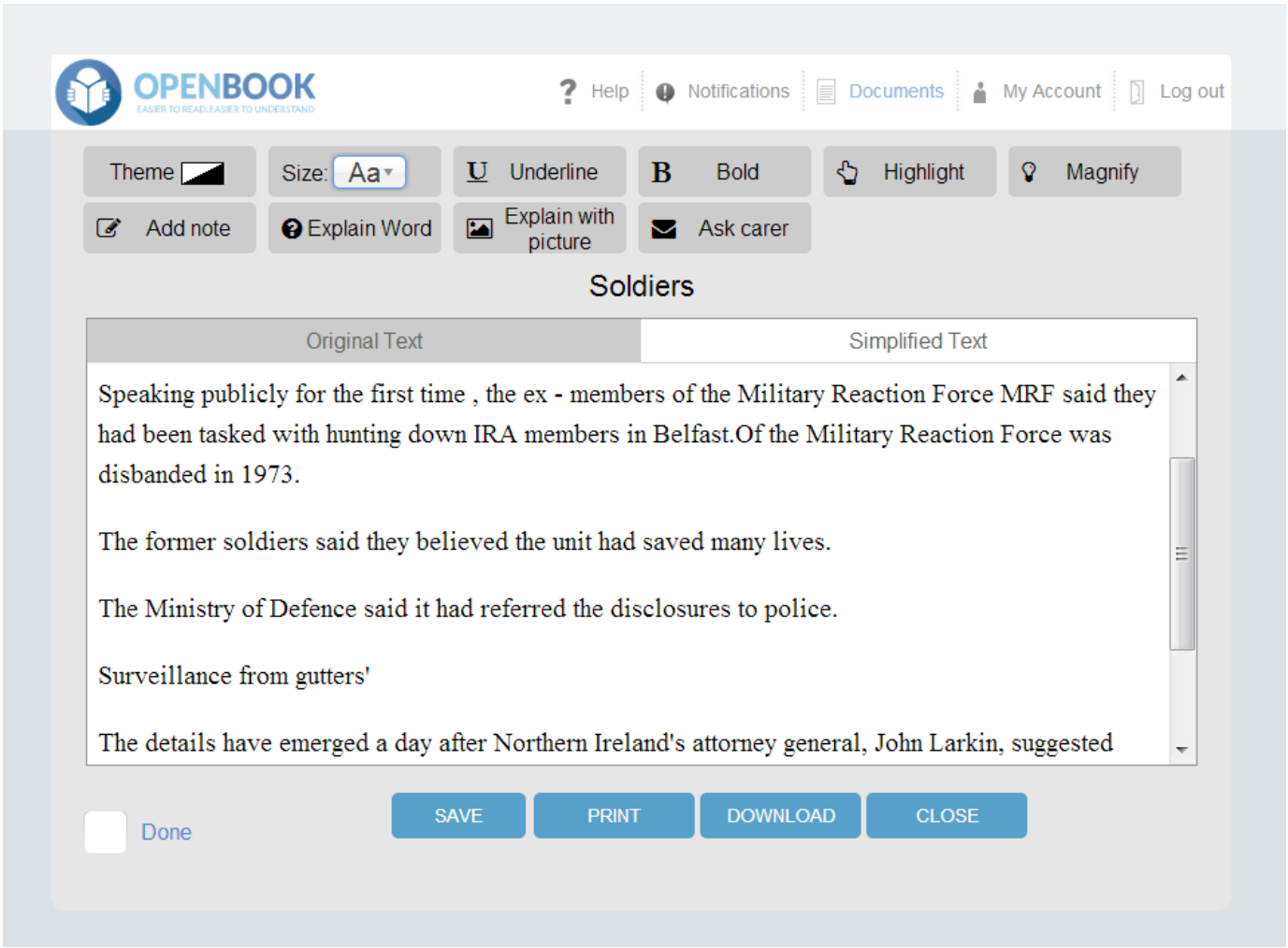
Format of a sentence fragment where no obstacle was detected:

```
<span class="sentence" data-id="30640">
  <span class="word" data-id="30649">The</span>
  <span class="word" data-id="30648"> </span>
  <span class="word" data-id="30647">Shock</span>
  <span class="word" data-id="30644"> </span>
  <span class="word" data-id="30643">of</span>
  <span class="word" data-id="30642"> </span>
  <span class="word" data-id="30641">the</span>
  <span class="word" data-id="30646"> </span>
  <span class="word" data-id="30645">Truth</span>
</span>
```

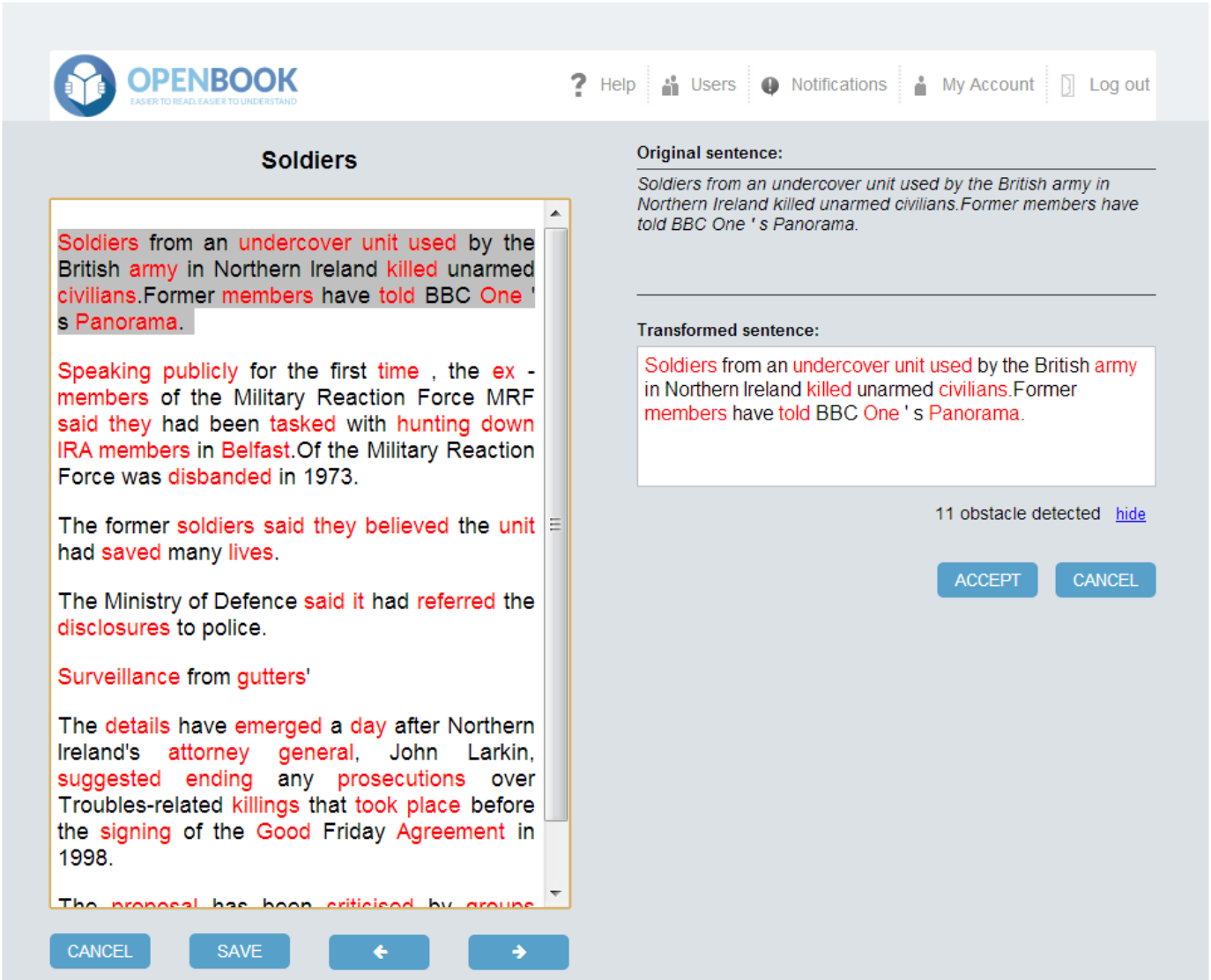
Format of a sentence fragment where obstacles exist at sentence and word level:

```
<span class="sentence obstacle" data-id="30640">
  <span class="word" data-id="30649">The</span>
  <span class="word" data-id="30648"> </span>
  <span class="word obstacle" data-id="30647">Shock</span>
  <span class="word" data-id="30644"> </span>
  <span class="word" data-id="30643">of</span>
  <span class="word" data-id="30642"> </span>
  <span class="word" data-id="30641">the</span>
  <span class="word" data-id="30646"> </span>
  <span class="word obstacle" data-id="30645">Truth</span>
</span>
```

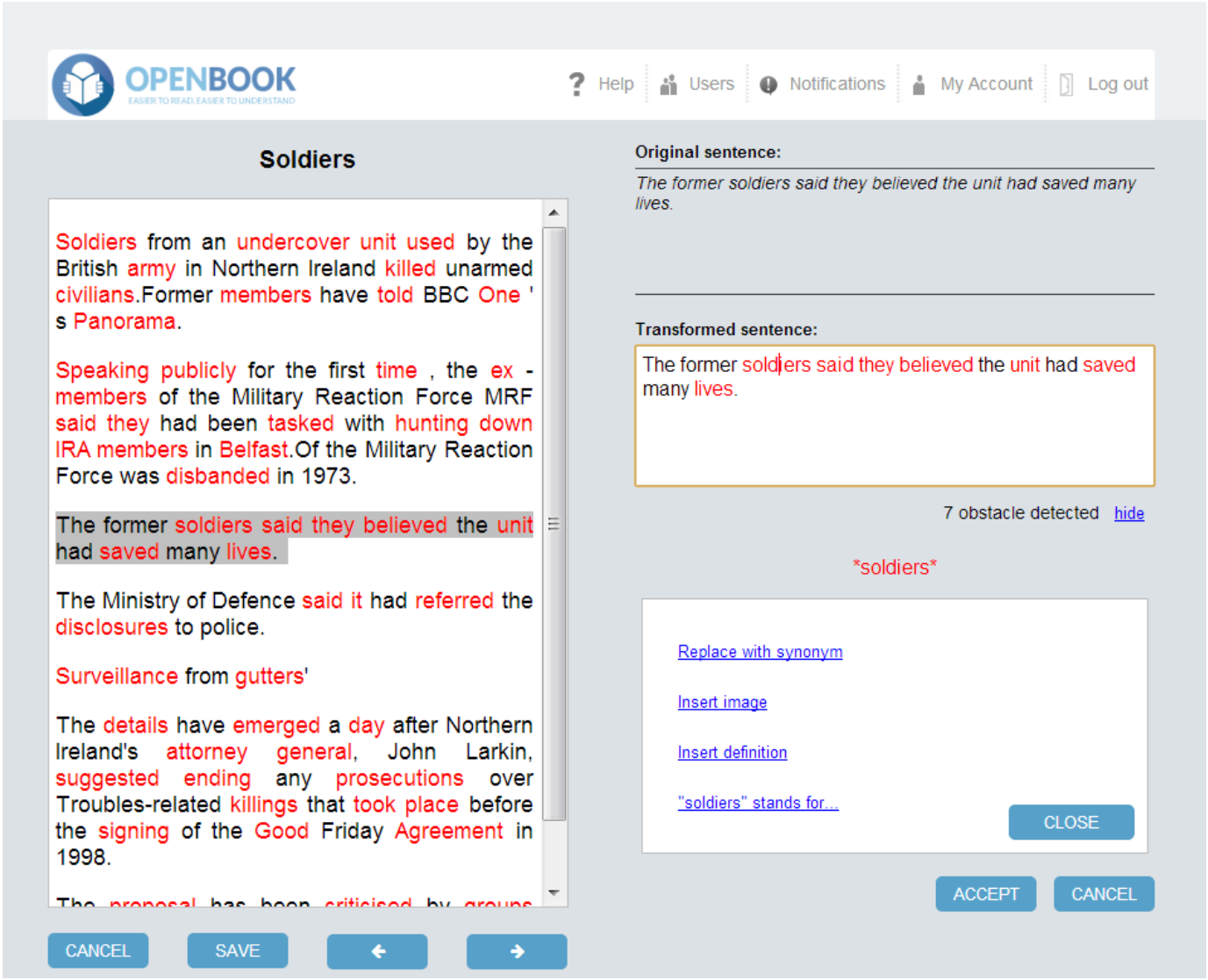
8) Display Output for Users: the display preferences are applied (styles, theme)



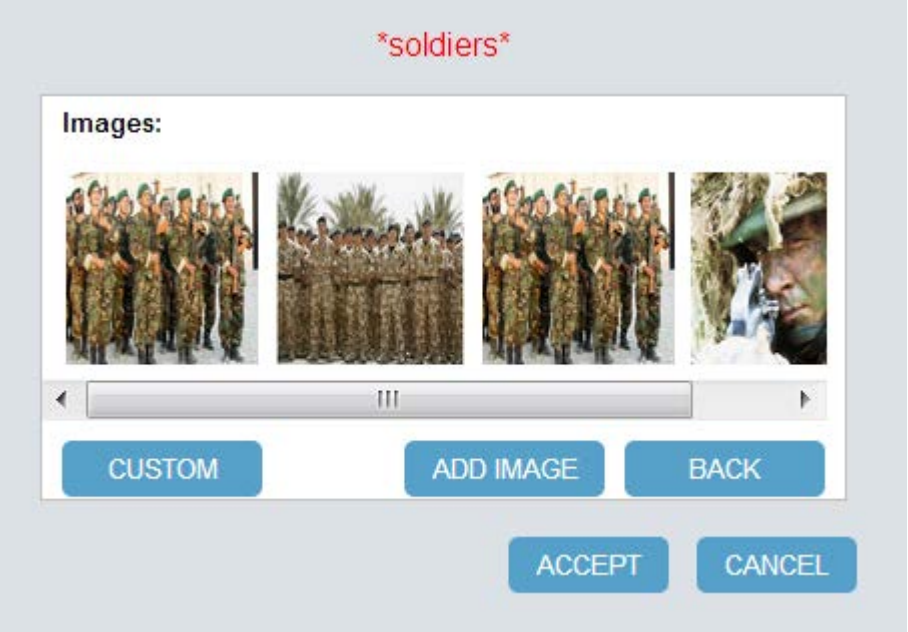
9) Display Output for Carers: The editor has to main panels: the document panel on the left and the review panel on the right. The carer can select any sentence of the document (using the mouse or navigating sentence by sentence). The current selected sentence is highlighted (in grey) and its contents and associated obstacle information is displayed in the review panel. The carer can freely edit the text of the sentence and accept or cancel these edits before moving to another sentence. Additional assistive elements are also available (i.e. reviewing assistive information produced by the system).



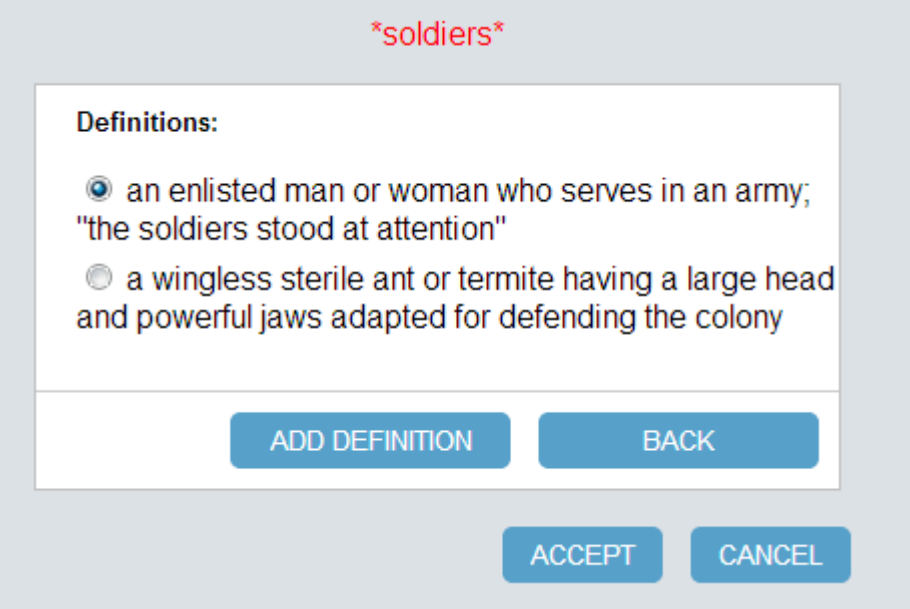
10) Display Output in the Carer Interface for a selected obstacle: when the carer select a word in the review panel, e.g. soldiers, then obstacle information associated to it is displayed in a panel on the bottom right side. This panel will contain links for definitions, synonyms, images or other assistive information relevant to this particular term/phrase. As new types of obstacles are supported by the system, this panel will be used to present any available obstacle removing tools/actions.



11) Images displayed for the word “soldiers”: the carer can chose to illustrate a concept visually by the inclusion of an image. Clicking on “Insert Image” will reveal a list of images that can be inserted in the document as an inline image.



12) Definitions displayed for the word “soldiers”: in a similar fashion, the carer can review and select a suitable definition. This is added to the sentence immediately after the term surrounded by parentheses. The carer can then directly edit the text of the definition or change its position in the sentence.



Changes to NLP Features

Anaphora Resolution

a. Spanish language (ES)

The Anaphora Resolution web service for Spanish developed within the FIRST project is able to detect and resolve pronominal anaphora, definite description and ellipsis. More information about the process of detecting and resolving anaphors can be found in D4.ii and D4.iii.

b. English language (EN)

The coreference resolver for English supports resolution for both nouns and pronouns, but currently only obstacles related to ambiguous pronominal references are supported in the Open Book prototype v2.

c. Bulgarian language (BG)

An anaphora resolution module is in development by the University of Wolverhampton and will be included in the next development phase (months 25-30);

Example: the output format for the coreference resolver

```
<Annotation EndNode="322" Id="2522" StartNode="319" Type="PronounAnaphora">
        <Feature>
                <Name className="java.lang.String">complexity</Name>
                <Value className="java.lang.String">0.9</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">StartNodeAntecedent</Name>
                <Value className="java.lang.String">298|313</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">EndNodeAntecedent</Name>
                <Value className="java.lang.String">309|318</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">confidence</Name>
                <Value className="java.lang.String">0.59723|0.59663</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">AntecedentId</Name>
                <Value className="java.lang.String">2520|2521</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">chain</Name>
                <Value className="java.lang.String">gruesa capa|hielo</Value>
        </Feature>
</Annotation>
```

The annotations produced include:

a) Information about the obstacle complexity
b) List of possible antecedents, for each one: the annotation ID its position in the text (start and end offset) and the solutions' confidence
c) Information to assist carers to remove the obstacle, i.e., a feature 'chain' listing names which could be used to replace the pronoun (similar to how a carer selects a synonym to replace a complex word)

Future work: in the next months the parameters of this web service will be changed into JSON format to simplify the compatibility of all web services and increase flexibility. This change will enable the inclusion of personalisation parameters to be considered by the service when processing a document. This will allow the service to filter out certain obstacles which are deemed irrelevant to a particular user. This change will be operated in the third year of the project.

Disambiguation obstacles

This NLP component processes a text with the aim of detecting and resolving different types of difficult words that could be problematic for people with Autism Spectrum Disorders. Such difficult words were proposed in D2.2, and include several types: polysemic words, mental verbs, less common words, specialized words, infrequent slang and acronyms. More information about the process followed for detecting and resolving each of them can be found in D4.ii and D4.iii.

This component provides the best definition and synonyms for a word within the context this word is being used. Additionally, other type of information is also provided (e.g., part of speech category of the word, or the type of obstacle).

Depending on the type of obstacle, it would be necessary to apply a word sense disambiguation process as an intermediate stage between the detection and resolution stages. This occurs for: polysemic words, mental verbs, less common words and specialized words. For acronyms and infrequent slangs no word sense disambiguation process is necessary, since for them, the web services just provides the expansion in the case of acronyms and the normalized word for infrequent slang.

Example 1: Output format for obstacles that need the word sense disambiguation stage (polysemic words, mental verbs, less common words and specialized words):

```
<Annotation EndNode="1181" Id="3895" StartNode="1170" Type="Definition">
        <Feature>
                <Name className="java.lang.String">definition</Name>
                <Value className="java.lang.String">a physicist who studies astronomy|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">synonyms</Name>
                <Value className="java.lang.String">astronomer, uranologist, stargazer|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">token</Name>
                <Value className="java.lang.String">astronomers</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">synonym</Name>
                <Value className="java.lang.String">astronomer|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">idWN</Name>
                <Value className="java.lang.String">09818343n|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">typeToken</Name>
                <Value className="java.lang.String">RARE</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">lemma</Name>
                <Value className="java.lang.String">astronomer</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">POS</Name>
                <Value className="java.lang.String">N</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">complexity</Name>
                <Value className="java.lang.String">0.6</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">confidence</Name>
                <Value className="java.lang.String">1.0|</Value>
        </Feature>
</Annotation>
```

This format includes the following information (not all linguistic features are available to users/carers):

- **Definition**: the definition of a detected word being an obstacle.
- **Synonyms**: the list of synonyms of the detected word.
- **Token**: the detected word.
- **Synonym**: the simplest synonym computed taking into account the guidelines in D2.2.
- IdWN: the identifier of WordNet for English, MultiWordNet for Spanish and Balkanet for Bulgarian. WordNet, MultiWordNet and Balkanet are semantic resources that are used within the detection and resolution process of this type of obstacles. More information about these resources can be found in D4.ii and D4.iii.
- TypeToken: the type (e.g., mental verb, polysemic).
- Lemma: the lemma of the detected word.
- POS: the grammar category of the detected word.
- Complexity: the obstacle's complexity
- Confidence: the probability to obtain the correct solution automatically according to the model

Example 2: output format for the obstacles that do not need disambiguation (acronyms and infrequent slang).

```
<Annotation EndNode="583" Id="2204" StartNode="577" Type="Definition">
        <Feature>
                <Name className="java.lang.String">definition</Name>
                <Value className="java.lang.String">United Nation's International Children's
        Emergency Fund (Fondo Internacional de las Naciones Unidas para Emergencias de la
        Infancia)</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">token</Name>
                <Value className="java.lang.String">UNICEF</Value>
        </Feature>
</Annotation>
```

This output only returns the **token** detected and its **definition**. The definition is the expansion of the acronyms or the normalized word for the infrequent slang.

Syntactic Simplification

Syntactic Processor

Currently two types of structural obstacles are tackled by the service: a) detecting and rephrasing structurally complex sentences, and b) detecting appositions and noun post-modifiers. These cover several user requirements related to syntactic complexity listed in D2.2.

Detecting structurally complex constituents is performed by a statistical tagging approach which is described in D3.i. The method classifies certain words or tokens (signs) which typically bound or link constituents with one of over 30 labels. This annotated information is used by a rule processor. Each rule identifies specific constituents, then rephrases the sentence into a sequence of at least two simpler sentences. This method is described in D3.i and D3.ii.

Original: *Imelda Marcos, former first lady of the Philippines, is currently becoming the subject of musicals, song cycles and shows on a worldwide arena.*

Transformed: *Imelda Marcos is former first lady of the Philippines. Imelda Marcos is currently becoming the subject of musicals and shows on a worldwide arena.*

Detecting appositions and noun post-modifiers is performed by two complementary methods. The first method uses labelled signs and applies another set of rules to detect the extent of appositions and is similar to detection of complex constituents. The second method employs a statistical CRF tagger trained on a specially annotated dataset. This second approach uses generic linguistic features and can be employed for all three languages. Currently, data annotation is complete for English, and on-going for Spanish and Bulgarian.

Example: *There is a danger that these new art forms airbrush out atrocities**, which accompanied the ostentation and glamour**.*

Several parameters can be used to specify which obstacles need to be processing, the format of the input/output documents as well as thresholds determined by the personalisation settings of the user.

Long words

As proposed in D2.2, a word is considered long when it has more than seven characters. In addition, within this obstacle, we also include the detection and resolution of adverbs ending by –ly for English, -mente for Spanish and –o for Bulgarian.

For the detection and resolution on this obstacle a module within the disambiguation web service is developed. Therefore the Disambiguation Web Service described in the previous section also tackles this obstacle.

Example below shows the information produced for a long noun "**mathematics**". The **typeToken** feature is **LONGWORDS**. For adverbs, the output has similar information (the only difference is the part of speech tag: **N** for Nounds, **Adv** for adverbs).

Example: output format for long word: mathematics

```
<Annotation EndNode="1855" Id="4352" StartNode="1844" Type="Definition">
        <Feature>
                <Name className="java.lang.String">definition</Name>
                <Value className="java.lang.String">a science (or group of related
        sciences) dealing with the logic of quantity and shape and arrangement|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">synonyms</Name>
                <Value className="java.lang.String">mathematics, math, maths|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">token</Name>
                <Value className="java.lang.String">mathematics</Value>
        </Feature>
        <Feature>
                        <Name className="java.lang.String">synonym</Name>
                <Value className="java.lang.String">mathematics|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">idWN</Name>
                <Value className="java.lang.String">06000644n|</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">typeToken</Name>
                <Value className="java.lang.String">LONGWORDS</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">lemma</Name>
                <Value className="java.lang.String">mathematics</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">POS</Name>
                <Value className="java.lang.String">N</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">complexity</Name>
                <Value className="java.lang.String">0.9</Value>
        </Feature>
        <Feature>
                <Name className="java.lang.String">confidence</Name>
                <Value className="java.lang.String">1.0|</Value>
        </Feature>
</Annotation>
```

Wikipedia Disambiguation

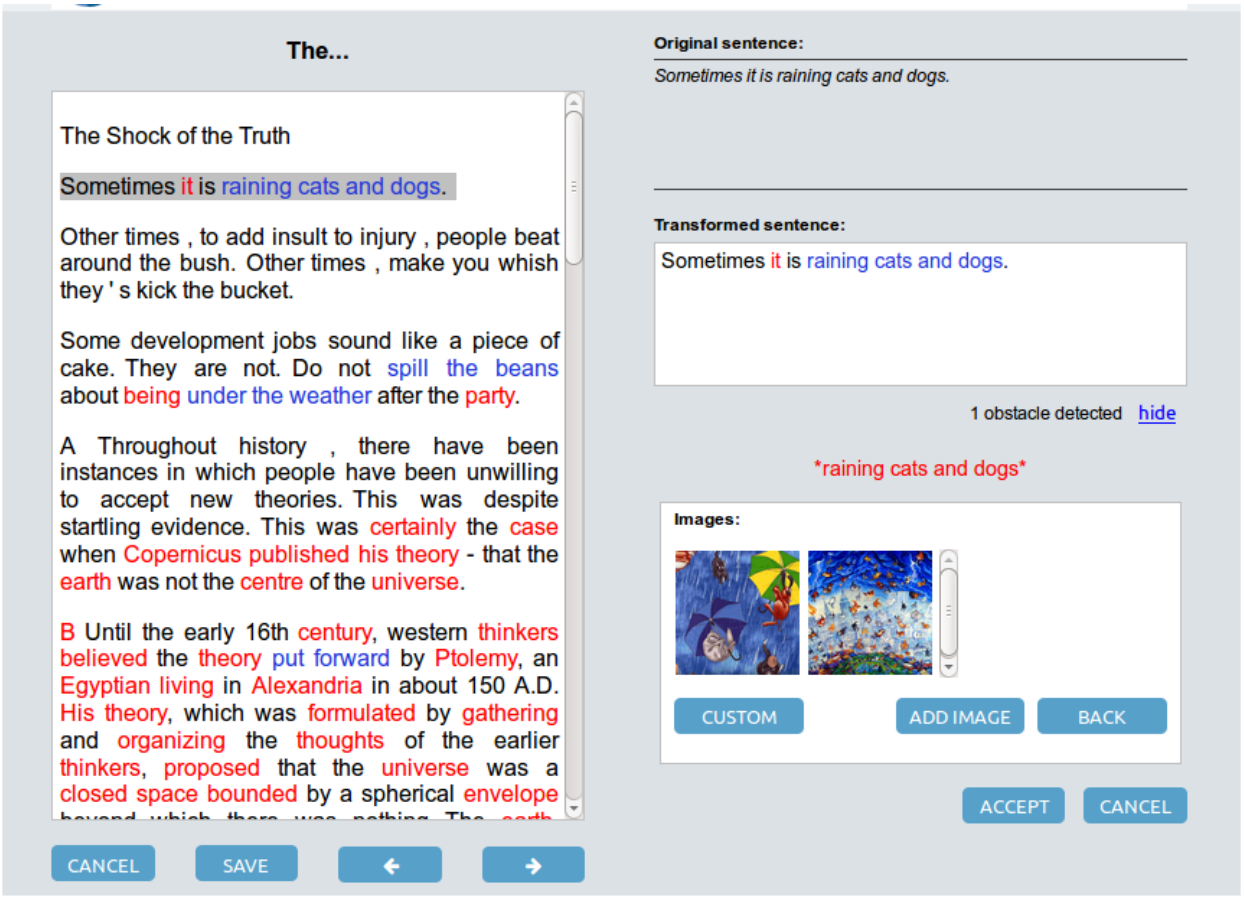
Further changes have been made to the Disambiguator service which now uses JWPL framework to retrieve information from Wikipedia. Currently Wikipedia categories are retrieved and the intention in the future is to also retrieve other useful information. This takes the specialised terms detected by the Disambiguation service (UA) and then links them to a relevant Wikipedia article, if possible.

Offline Image Retrieval

This service retrieves images from Wikipedia for those concepts which were successfully disambiguated and linked to Wikipedia articles. It also retrieves images from ImageNet for disambiguated WordNet concepts. Links to these images are added as GATE annotations and processed by the IE. Users and Carers using Open Book prototype v2 will see the actual images displayed in the interface.

Idiom Detection

This web service makes use of in-house developed lists of figurative language expressions for English, Spanish and Bulgarian to identify idioms in the user documents. It can be configured to perform the matching at the lemma level or using Java Annotation Patterns Engine (JAPE) expressions. Currently, a definition/explanation is generated. A carer can chose to inserted it in the document (to enable the user to learn new idioms) or to remove the idiom and rephrase the text accordingly.



Conclusion & Next Steps

Open Book software version prototype 2 can detect and process a range of obstacles in three languages of the consortium. It extends the functionality of the first prototype and provides an interactive editor for carers. The next version to be released will focus mainly on enhancing the current functionality with more complex obstacle resolution tools, developing and including support for of personalised document processing and addressing feedback collected from clinical partners and users of the platform.

The main change planned for the next phase of the project is adding personalised obstacle processing capabilities to the LT services. Although the current version supports personalisation aspects such as visual preferences, it was determined that, depending on which obstacles are relevant to a particular user, more complex obstacle processing is necessary including possible conflict resolution. The agreed approach for accommodating this functionality is to include support for personalisation parameters in each LT service.

These parameters will be serialised in an extensible format (JSON) to allow flexibility in adding or updating parameter values. The parameters control which (sub) types of obstacles to be analysed, personalisation information regarding complexity thresholds. This will allow the service to filter out certain obstacles which are deemed irrelevant to a particular user. This change will be operated in the third year of the project.

Future enhancements planned for the next 6 months are to split the functionality of the disambiguation web service: one web service will tackle obstacles needing a word sense disambiguation stage and the other web service will tackle obstacles not needing this stage. Also, the Disambiguation Web Service will be optimized with respect to system run time and the amount of information provided (e.g., unifying the annotations for words appearing several times in the text with the same sense).

From a usability perspective, another activity relevant to the Open Book Tool is to determine which obstacles are robust enough to be removed automatically, and which obstacles can only be used as an assistive element (reading aide) and otherwise requires carer validation. Depending on evaluation results collected in WP3, WP4, WP5, a priority for the third year is to determine a set of ‘default’, recommended profiles.

At the level of the user interface, a task for the year ahead is to identify a user friendly way to present personalisation settings affecting linguistic parameters. This has to take into consideration the fact that most users do not have knowledge of linguistic concepts.