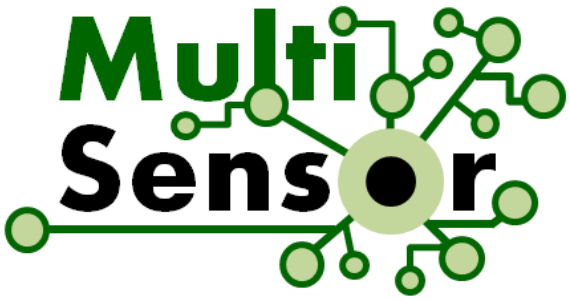

MULTISENSOR

Mining and Understanding of multilinguaL contenT for Intelligent Sentiment Enriched coNtext and Social Oriented inteRpretation

FP7-610411

D6.3

Integrated summarisation system

Dissemination level:	Public
Contractual date of delivery:	Month 34, 31/08/2016
Actual date of delivery:	Month 37, 8/11/2016
Workpackage:	WP6 Summarisation and content delivery
Task:	T6.3 Content selection metrics T6.4 Content delivery procedures T6.5: Concept-based summarisation T6.6: Advanced single and multi-document summary delivery system T6.7: Summarization evaluation
Type:	Prototype
Approval Status:	Final
Version:	2.0
Number of pages:	48
Filename:	D6.3_SummarisationSystem_2016-11-08_v2.0.pdf

Abstract

This deliverable reports the development of advanced extractive and abstractive summarisation methods. This version of the deliverable also reports evaluation results (T6.7) for all summarization functionalities developed in the scope of the project.

and liability.

Co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	22/07/2016	Layout proposal	G. Casamayor (UPF)
0.2	22/07/2016	Description of some modules	M. Ballesteros (UPF)
0.3	25/08/2016	Contributions to section 5	S. Mille (UPF)
0.4	27/08/2016	Contributions to sections 2,3 and 4	G. Casamayor (UPF)
0.5	28/08/2016	Final version of section 5	S. Mille (UPF)
0.6	30/08/2016	Contributions to section 2	J. Codina (UPF)
0.7	30/08/2016	Final versions of sections 2,3 and 4	G. Casamayor (UPF)
0.8	31/08/2016	Added glossary, intro, abstract and proof-read all sections.	G. Casamayor (UPF)
0.9	1/09/2016	Internal review	B. Vaisman (LT)
1.0	2/09/2016	Final version	G. Casamayor (UPF)
1.1	7/11/2016	Extended deliverable	G. Casamayor, J.Codina and S. Mille (UPF)
1.2	8/11/2016	Internal review	B. Vaisman (LT)
2.0	9/11/2016	Amendments following internal review	G. Casamayor (UPF)

Author list

Organisation	Name	Contact Information
UPF	Gerard Casamayor	gerard.casamayor@upf.edu
UPF	Miguel Ballesteros	miguel.ballesteros@upf.edu
UPF	Simon Mille	simon.mille@upf.edu
UPF	Joan Codina	joan.codina@upf.edu

EXECUTIVE SUMMARY

This document presents the advanced methods in extractive and abstractive summarisation developed as part of tasks 6.3, 6.4, 6.5 and 6.6. The extractive summarisation functionality of the MULTISENSOR system presented in D6.1 (T6.1 and T6.2) has been extended to incorporate work in T6.4 and to adapt it to new user requirements, more precisely to the requirements stemming from the exploitation plan followed in Y3 of the project. These requirements include a new query-based summarisation mode. The content selection metrics presented in D6.2 (T6.3) have been applied to two complementary corpora with semantic annotations. We also report the development of the abstractive summariser by focusing on its two modules, the text planner or conceptual summariser (T6.5) and the multilingual linguistic generator (T6.6). The former applies the content selection metrics to select and organise relevant contents from the MULTISENSOR semantic repository, while the latter renders multilingual summaries from the selected contents. The work reported in D6.3 contributes towards the completion of Milestone MS5 (Final System). This version of the deliverable also reports the evaluation results foreseen as part of T6.7, and conclusions drawn from the results.

ABBREVIATIONS AND ACRONYMS

AM	Automatic Summarisation
ANNIE	A Nearly-New IE system
API	Application Programming Interface
ARFF	Attribute-Relation File Format
AS	Automatic Summarisation
CEP	Content Extraction Pipeline
CoNLL	Conference on Natural Language Learning
DoW	Description of Work
DSyntS	Deep Syntactic Structure
DUC	Document Understanding Conference
EL	Entity Linking
GATE	General Architecture for Text Engineering
IDF	Inverse Document Frequency
IE	Information Extraction
JAPE	Java Annotation Patterns Engine
JSON	JavaScript Object Notation
JSON-LD	JSON for Linking Data
LD	Linked Data
LOD	Linked Open Data
LSTM	Long Short-term Memory Network
MorphS	Morphological Structure
MT	Machine Translation
MTT	Meaning Text Theory
NE	Named Entity
NER	Named Entities Recognition
NIF	NLP Interchange Format
NLG	Natural Language Generation
NLP	Natural Language Processing
OWL	Web Ontology Language
RDF	Resource Description Format
REST	Representational State Transfer
ROUGE	Recall-Oriented Understudy of Gisting Evaluation
SEW	Semantically Enriched Wikipedia
SIMMO	Socially Interconnected MultiMedia-enriched Objects
SSyntS	Surface Syntactic Structure
T	Task
TF	Term Frequency

UC	Use Case
WP	Work Package
WPRA	Web Page Ranking Algorithm

Table of Contents

1 INTRODUCTION

WP6 deals with the production of multilingual summaries. The generation of both extractive (T6.1, T6.4) and abstractive (T6.3, T6.5 and T6.6) summaries is foreseen in the DoW. This is the last report on the work in WP6 towards developing summarisation methods. Previous reports (D6.1 and D6.2) described the setup of basic single-document and multiple-document extractive summarisation pipelines (T6.1), the compilation of corpora for developing empirical methods (T6.2), the design of theoretical content selection metrics that can be used to determine the relevance of contents (T6.3), and experiments with using semantic information produced by the CEP (WP2 and WP3) as features for the summarisation methods.

This deliverable reports final work in extractive summarisation (T6.4), which has been geared towards adapting the pipelines to new user requirements stemming from the exploitation plan initiated in Y3 of the MULTISENSOR project. Following discussions between user and technical partners, summarisation was identified as one of the functionalities, which could be commercially exploited by the partners. User feedback and analysis of technical requirements indicated that new modalities of extractive summarisation were needed, namely online version of the pipelines capable to quickly produce summaries of user-provided text, and a new query-based summarisation functionality in which summaries are tailored to include contents relevant to a set of user-specified keywords. Additionally, summarisation parameters have been fine-tuned using corpora-based methods.

The content selection metrics introduced in D6.2 (T6.3) have been applied to two corpora, a dump of the English Wikipedia automatically annotated with high-quality disambiguated mentions to BabelNet senses and the corpus of 36.000+ documents processed by the CEP and stored along with the resulting annotations in the MULTISENSOR semantic repository. We report an experimental test aimed at evaluating the coverage of both corpora on the senses communicated in documents belonging to the project UCs.

In parallel to further improvements of the extractive pipelines, an abstractive summariser has been developed that is capable of generating multilingual summaries from the contents of the semantic repository. The abstractive summariser takes as input an entity and produces a gist of the most relevant contents about that entity in the semantic repository. It includes a concept-based text planner (T6.5) that operates on the contents of the semantic repository and applies a novel method for ranking contents that maximises both relevance and coherence of the resulting summary. The text planner leverages the content selection metrics to empirically estimate the relevance of contents, and uses structures representations of the contents to both distribute the relevance assessments to related contents and to create a sequence of contents that guarantees that contents are always semantically connected to previous contents in the plan. The surface multilingual generation module (T6.6), on the other hand, uses a combination of empirical corpus-based methods and rule-based modules to map the conceptual structures produced by the text planner onto language realisations of the contents, placing emphasis on ensuring both grammaticality and readability. This document concludes with Sections 6 and 7, reporting the results of evaluating the summarization functionalities and the conclusions drawn at the end of the project.

The document is structured as follows: section 2 reports on the extensions and improvements of the extractive functionality of the MULTISENSOR system. Section 3 describes the corpora used to obtain estimations of relevance from the content selection metrics. Section 4 describes the text planning module, while section 5 does the same for the linguistic realisation module. A short summary details the contents missing in this version of the deliverable, will be included in the extended version.

1.1 Architecture

Summarisation functionalities are implemented both in the online and offline modalities of the MULTISENSOR final prototype. Extractive and abstractive functionalities are implemented as part of two separate REST services. The extractive summarisation service is integrated in the offline CEP, as shown in Figure 1.

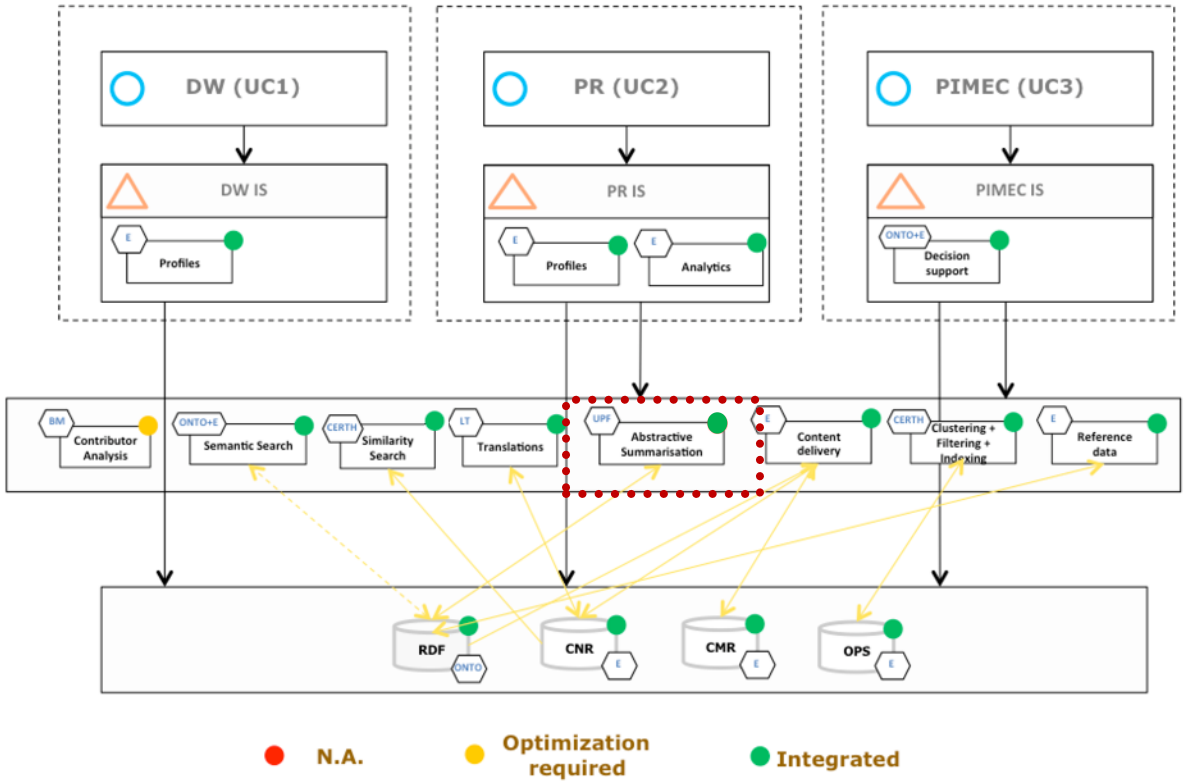

Figure 1: Online modality architecture with Summarisation service highlighted

As the other services in the CEP, it consumes and produces a JSON-encoded SIMMO object, which it updates with a single-document summary. Both extractive and abstractive summarisation services are also integrated in the online architecture of the prototype, as seen in Figure 2. On-line extractive summarisation consumes raw text from one or more documents, and produces the summary as raw text too. On-line extractive summarisation can also be run from a user query and a single text. Abstractive summarisation is always on-line and starts from a user query. It accesses the RDF semantic repository in order to generate multilingual summaries returned as raw text.

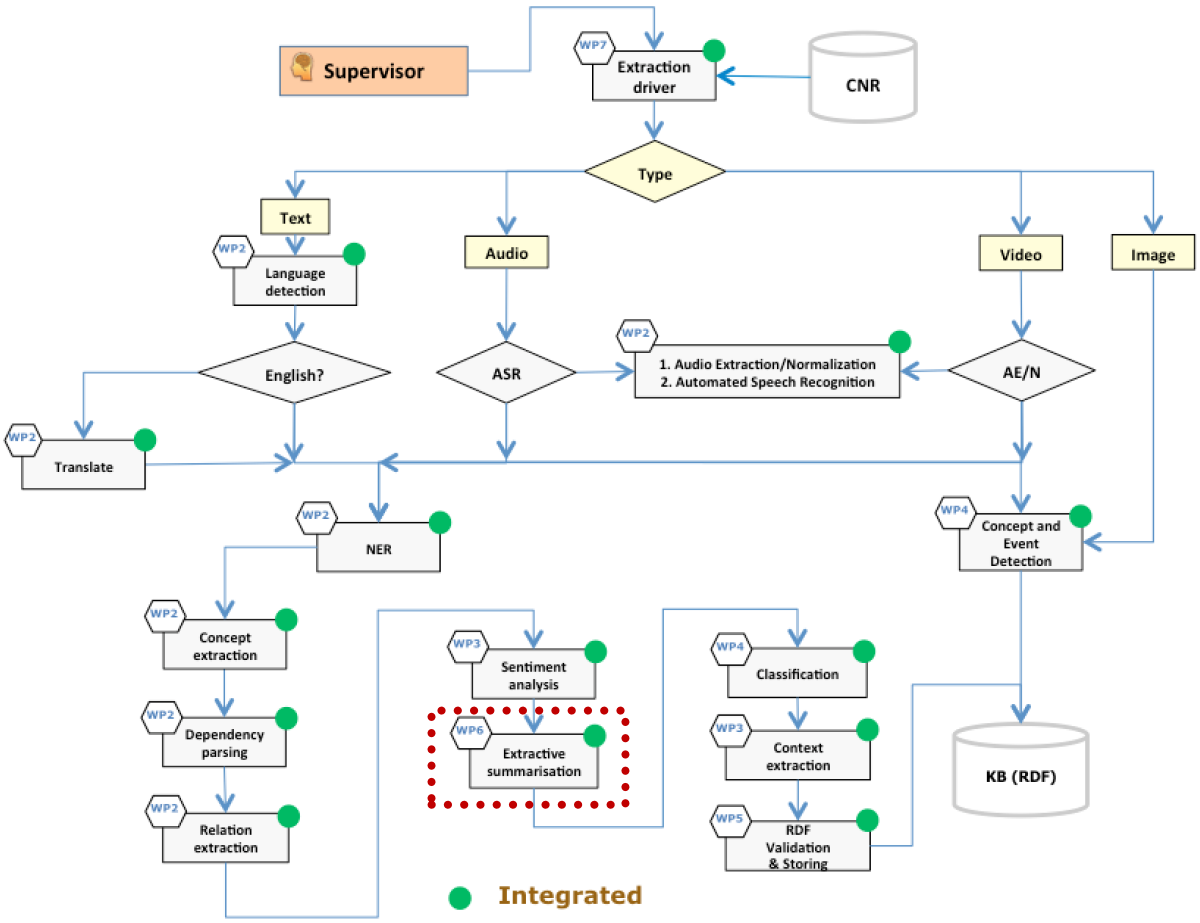

Figure 2: Content Extraction Pipeline (CEP), part of the offline architecture, with the extractive summarisation service highlighted.

2 ADVANCED EXTRACTIVE SUMMARISATION

The development of advanced methods for extractive summarisation as part of task T6.4 was already reported in D6.2. Following exploitation plans and new user requirements, we have carried out new developments that we report in this section. While previous efforts were invested in applying semantic data to extractive (text-to-text) summarisation, the work reported in this deliverable focuses on both new summarisation modalities and incremental updates of the original pipelines reported in D6.1.

2.1 Updates to summarisation pipelines

In order to facilitate exploitation of the extractive summarisation functionality, extensive quality testing has been conducted in order to identify and fix problems in the generated summaries. Additionally, UC-specific IDF tables containing lemmas and respective IDF counts have been obtained from the MULTISENSOR repository of articles. The SUMMA-based pipelines and the resources developed for the MULTISENSOR UCs can be found in **https://github.com/talnsoftware/summa_pipelines**.

2.1.1 Single document

Following user requirements, extractive summarisation for single documents has been made available in both off-line and on-line modalities, where off-line refers to execution of the pipeline as part of the batch production of summaries for articles analysed with the CEP, while on-line execution is triggered by a user request to summarise a text. In off-line execution the input to the pipeline is a JSON-encoded SIMMO (Tsikrika et al., 2015) object that contains, amongst other information produced by the MULTISENSOR services, the RDF-encoded output of tokenisation and sentence splitting steps. In contrast, the on-line execution starts from raw text without any tokenisation or sentence splitting. Adapting to both modes of execution required the creation of separate off-line and on-line pipeline, the former reading the NIF[1] annotations of tokens and sentences (Hellman et al., 2013) and importing them to GATE[2] (Cunningham et al., 2011), while the latter runs GATE-based plugins for tokenizing and splitting into sentences the text received. The architecture of each pipeline are shown in Figure 3 and Figure 4 respectively. An explanation of what the individual modules do can be found in D6.1.

[1] http://persistence.uni-leipzig.org/nlp2rdf/

[2] https://gate.ac.uk/

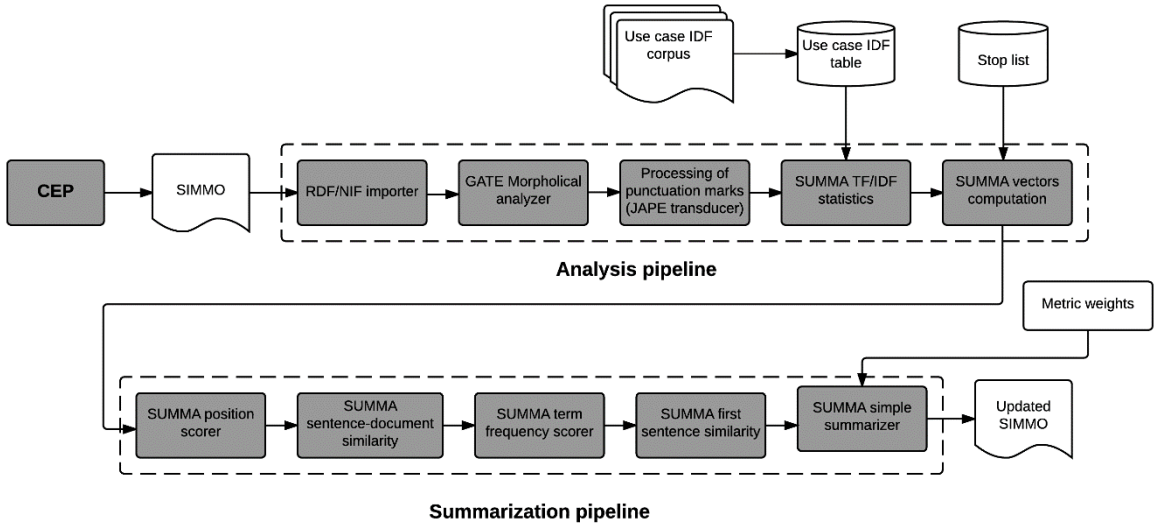

Figure 3: Single document off-line summarisation pipeline

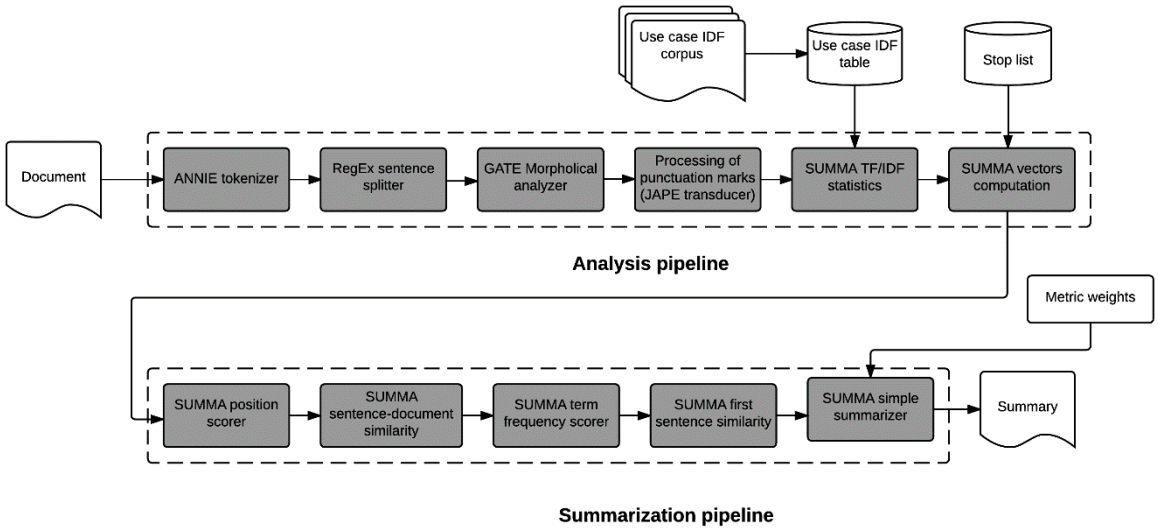

Figure 4: Single document on-line summarisation pipeline

2.1.2 **Query-based**

As part of the exploitation strategy followed during the last year of the MULTISENSOR project, extractive summarisation has been extended to serve user requests and tailored to specific keywords introduced by the user, i.e. generating a summary of a document where sentences making mention to specific words are weighted favourably when considering their inclusion in the summary. The decision of developing a query-based pipeline followed joint analysis of user requirements for possible exploitation and technical requirements of the project functionalities.

A new SUMMA pipeline has been created that performs a linguistic analysis of both the document and the query and creates vector representations for each, and then summarisation proceeds by incorporating a new metric that represents the similarity of each sentence in the original document to the query. The architecture of the resulting pipeline is shown in Figure 5.

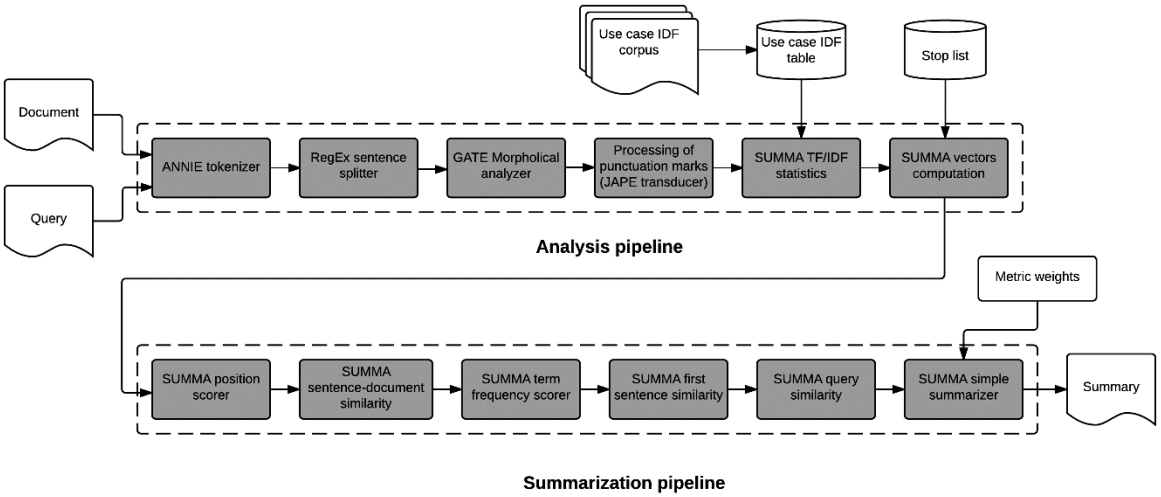

Figure 5: Query-based summarisation pipeline

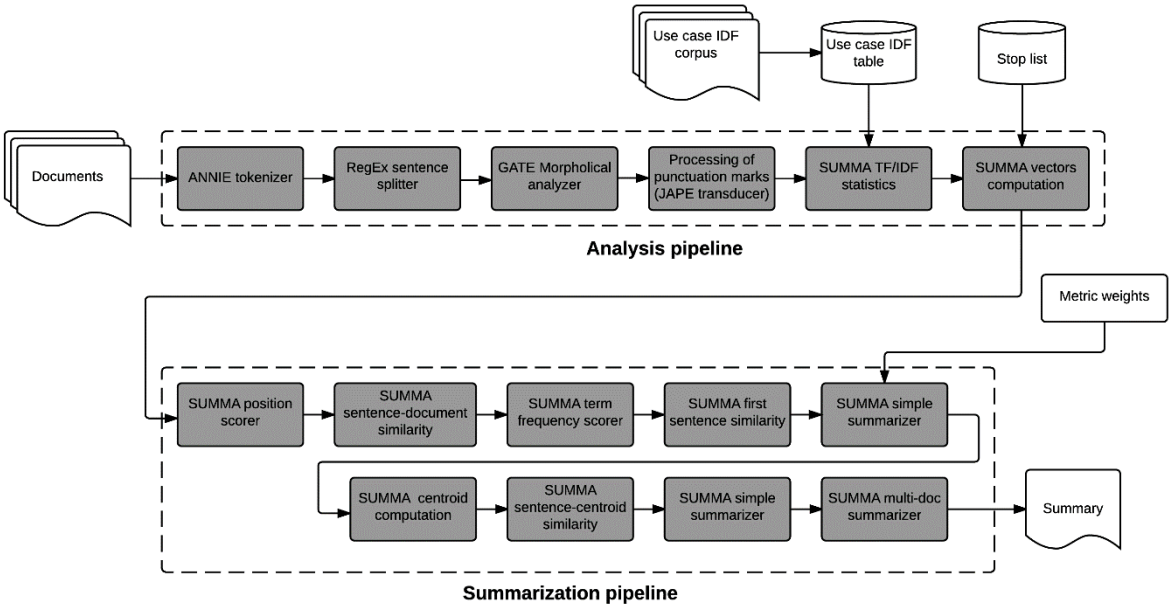

Figure 6: Multiple document summarisation pipelines

2.1.3 **Multiple-document**

According the user requirements and corresponding to technical conditions the multiple document pipeline developed during Y1 and reported in D6.1 has been adapted to be run on-line and consequently extended by the additional modules for tokenization and sentence

splitting of the original documents. The resulting pipeline is shown in **Error! Reference source not found.**.

2.2 Estimating weights for the features

In the SUMMA based extractive summarisation pipelines, all sentences of the documents to summarise are weighted using the following formula:

$$score(U) = \sum_{i=0}^{l} Wi * Fi$$

Where *U* is a sentence, *Fi* is the value of a metric *i* and *Wi* is the weight associated with feature *i.* The metrics used in the SUMMA pipelines developed for MULTISENSOR are described in D6.1 and in section 2.1 of this document. In order to determine the weights for each metric we applied a linear regression procedure on a gold standard of news articles with human-crafted abstractive summaries.

While pressrelations provided a gold standard of journalistic texts paired with human-authored abstractive summaries, we could not use it because the summaries were tailored to specific keywords, and the contents communicated weren't the most relevant of the document in an objective sense. For this reason, we selected the documents and summaries in the gold standard of the Document Understanding Conferences (DUC3)[4]. We used the summaries of the DUC-2001 shared task which composed of near 200 journalistic articles with their respective summaries. We split the set into two subsets, one having 20 documents for testing and the rest for training. The two main reasons to choose this corpus are that it belongs to the news domain and that we can compare our results with other publications based on the same corpus.

We trained the weight estimation procedure in the following way: for each sentence in the document we measure its similarity to each of the sentences in the DUC summary using the Rouge metric (Lin, 2004) and annotate the sentence with this value. We measured the contribution of each metric to the similarity of sentences in the original documents to the sentences in the summary. We processed each document with the MULTISENSOR single document extractive pipeline in order to compute all summarisation metrics for their sentences, and used the ROUGE metric as the target label. We created ARFF files in which each line corresponded to a sentence and contained its metric values followed by the average ROUGE similarity to the sentences in the corresponding summary, as shown in Figure 7. ARFF is a file format used for WEKA 3[5] (a collection of machine learning algorithms for data mining tasks). For each different metric that we want to test we generate two files, one for training and the other for testing. The goal is to select the most appropriate ones for summary generation.

[4] http://www.nist.gov/tac/

[5] http://www.cs.waikato.ac.nz/ml/weka/

```
@RELATION features_summa
@ATTRIBUTE position_score NUMERIC
@ATTRIBUTE first_sim NUMERIC
@ATTRIBUTE sent_doc_sim NUMERIC
@ATTRIBUTE tf_score NUMERIC
@ATTRIBUTE rouge_to_summ NUMERIC

@DATA
1.0,1.0,0.3725176,1.0,0.6969696
0.923,0.0926,0.325334,0.509174,0.3333
0.8463,0.170,0.30093,0.6692824,1.0

……
```

Figure 7: Sample ARFF file heading

We use the standard WEKA Linear Regression algorithm to adjust the weights of the metric given the ARFF file. The output obtained using the whole file for adjusting the weights can be seen on Figure 8. The weights obtained by linear regression have been integrated in the single-document extractive pipelines.

```
=== Run information ===
Scheme:weka.classifiers.functions.LinearRegression -S 0 -C -R 1.0E-8
Relation:     features_summa
Instances:    16819
Attributes:   5
              position_score
              first_sim
              sent_doc_sim
              tf_score
              rouge_to_summ
Test mode:user supplied test set: size unknown (reading incrementally)
=== Classifier model (full training set) ===
Linear Regression Model
rouge_to_summ =
      0.2441 * position_score +
      0.0817 * first_sim +
      0.0928 * sent_doc_sim +
      0.0714 * tf_score +
      0.011
```

Figure 8: Weka output after training, with the weights assigned to each feature

3 CONTENT SELECTION METRICS

D6.2 discussed using the MULTISENSOR RDF repository as a semantic corpus from which content selection heuristics could be obtained, and presented a candidate set of metrics to be used for content selection during the production of abstractive summaries. Due to delays in the deployment of the CEP pipeline and the population of the knowledge base, we have considered to use alternative corpora for the acquisition of content selection metrics. More precisely, we have experimented with the recently published *Semantically Enriched Wikipedia* (SEW)[6] (Raganato et al., 2013).

SEW is a sense-annotated corpus obtained from annotating a late-2014 dump of Wikipedia with disambiguated links to BabelNet synsets. It contains more than 200 million annotations of 4 million different senses. It has been shown to have both high precision (reached a precision of 0.934 in an evaluation against human-labelled corpora) and a large coverage (over 30% of all content words in Wikipedia are annotated in SEW). While the documents in SEW do not belong to the MULTISENSOR use cases, Wikipedia constitutes a body of knowledge large enough to cover most of the concepts and entities relevant to each use case. Additionally, the quality of the annotations in SEW can be hardly replicated in other domains because few collections of documents are structured in the way Wikipedia is (i.e., hyperlinks, categories, etc.).

Nevertheless, we have also experimented with the MULTISENSOR RDF repository as contents became available. At the time of writing this deliverable, the repository contains annotations of BabelNet synsets over 36.000 documents (belonging to UC1 and to UC2). While these annotations are likely to be significantly noisier than the ones in SEW, they are more focused to the domains at hand. In the following subsection we describe how we access the corpora and calculate the metrics, and present an experimental comparison between the two resources.

3.1 Indexing the SEW corpus

In order to obtain the counts needed to calculate the metrics presented in D6.3, we indexed SEW using Solr (Smiley et al., 2014). More specifically, we batch-processed the XML files of the complete version of SEW and populated a Solr index where all the information about the annotations in each document was stored. In order to be able to determine the number of sentences between pairs of annotations, we applied the Stanford CoreNLP (Manning et al., 2013) sentence splitter to the text of each document, and associated to each annotation in the index its sentence index. The Solr schema used to index all this information is shown in Figure 9. The schema can be queried to obtain both frequency counts $N(e_1)$ as the number of documents in which an entity id appears in the `annotationId` list one or more times. Co-occurrence counts $N(e_1, e_2)$ can be similarly obtained by counting the number of documents in which both annotation ids appear in the `annotationId` list. Ordered co-occurrence counts $N\langle e_1, e_2\rangle$ can also be obtained by additionally checking that the offsets in `annotationOffsetStart` and `annotationOffsetEnd` match the desired order. Finally,

[6] http://lcl.uniroma1.it/sew/

the number of sentences between two annotations in a document can be obtained using the `annotationSentenceIndex` value for each annotation.

```
"wikiarticleTitle" : "",
        "text" : "",
        "sentences" : [],
        "annotationId" : [],
        "annotationAnchor" : [],
        "annotationType" : [],
        "annotationOffsetStart" : [],
        "annotationOffsetEnd" : [],
        "annotationSentenceIndex" : []
        }
```

Figure 9: Solr schema for the SEW corpus

3.2 Querying the MULTISENSOR repository

The same counts can be obtained from the MULTISENSOR repository by querying the SPARQL endpoint of the instance of GRAPHDB managing the repository. Figure 10 shows the SPARQL queries used to obtain the frequency and co-occurrence counts. Frequency counts $N(e_1)$ can be obtained by running the first query, while co-occurrence counts $N(e_1, e_2)$ can be obtained by counting the number of document graphs `?g` in the results returned by the

second query. The same query can also be used to obtain ordering counts $N\langle e, e_2\rangle$ by adding an additional condition on the offsets of each pairs of annotations found. The document graphs and sentence indexes retuned by the second query can then be used in the last query to get the number of sentences between each pair of annotations.

3.3 Metrics of relevance and ordering

All three metrics considered for content selection, content co-occurrence $c(a_1, a_2)$, weighted co-occurrence $c_w(a_1, a_2)$ and content ordering $\sigma(a_1, a_2)$, are count-based in that their computation requires counts over the whole corpus. When the abstractive summariser needs to estimate the relevance of an annotation a_2 relative to another annotation a_1, the counts can be quickly obtained from the Solr index. What the formulas of each metric do not specify, however, is what to do when multiple annotations of either a_1 or a_2 are found in the same document.

In the case of content co-occurrence, we chose to ignore multiple annotations in the same document; in other words, we increment the count for each document in which both a_1 and a_2 appear regardless of how many times they are annotated. We adopt the same strategy when calculating weighted co-occurrence, taking the minimum distance between all pairs of annotations as the distance used to weight the count. For content ordering, we view all the annotations of a_1 and a_2 in a document as a sequence, and treat all pairs of consecutive annotations $N\langle a_1, a_2\rangle$ as separate instances, each updating the count once.

We have conducted a test in order to compare the potential contribution of the two corpora under consideration. We randomly selected 5 documents belonging to UC1 and 5 more belonging to UC2, and collected a set of 25 senses for each use case, taking 5 senses from each document. We then queried both SEW and the MULTISENSOR corpus for all pairs of entities in each of the two sets. Table 1 shows, for both SE and MULTISENSOR, the total number of times any two entities in each set of 25 entities were annotated in the same document, shown in the first row as *counts*. It also shows the average $N(e_1)$ as *avg freq*, and the average $N(e_1, e_2)$ as *avg co-occur*. The results indicate that while SEW has in average more mentions to the entities in the documents, the MULTISENSOR corpus has more documents in which pairs of entities are mentioned. In any case, both corpora are complementary and their counts are summed when calculating the content selection metrics during text planning. Indeed, taken together, the two corpora enable our metrics to serve as estimators of the relative relevance of pair of senses. Their contribution to the quality of generated summaries will be evaluated as part of the text planning module in the update to this deliverable.

	SEW		MULTISENSOR	
	UC1.1	UC1.2	UC1.1	UC1.2
Counts	92	64	152	126
Avg freq	5261	4329	627	595
Avg cooccur	15,28	11,22	22,01	19,63

Table 1: Counts for content selection corpora

4 CONCEPT-BASED SUMMARISATION

Task 6.5 involves the development of an abstractive summariser capable of generating summaries from conceptual representations rather than from text. In the context of the MULTISENSOR prototype, this involves generating summaries from the RDF contents of the semantic repository, which has been populated from the results of running multimedia documents through the CEP. While the nature, modelling and storage of the data extracted by the CEP is discussed in depth in the deliverables of WPs 2,3,4 and 5, we provide a short description of the contents used for the production of abstractive summaries in WP6.

An abstractive summariser must first select what contents will be communicated in the summary and organize them into a language-independent text plan, and then translate them into a grammatically correct text in one of the project supported languages. These two summarisation stages are referred to as text planning and surface realisation. T6.5 focuses on text planning. In the remainder of this section we provide a detailed description of the text planning module developed for MULTISENSOR, and then give an account of how the metrics for content selection described in Section 3 are used by the text planner. An evaluation of the text planner will be included in the extension to this deliverable.

4.1 Text planning

As already introduced in D6.3, we have followed a data-driven and bottom-up approach to text planning which involves exploring the contents available to the text planner and evaluating them according to some empirical metric of relevance. The exploration strategy translates into adopting graph view of the contents in the semantic repository and incrementally search and assess new contents from one or more focal points corresponding to user-specified entities which a summary is to be generated. An important concern is to go beyond selecting most relevant contents and structure these contents in a sequence that guarantees coherence in the resulting text, in other words, an ordering of contents that ensures that each sentence in the summary is semantically related to preceding sentences.

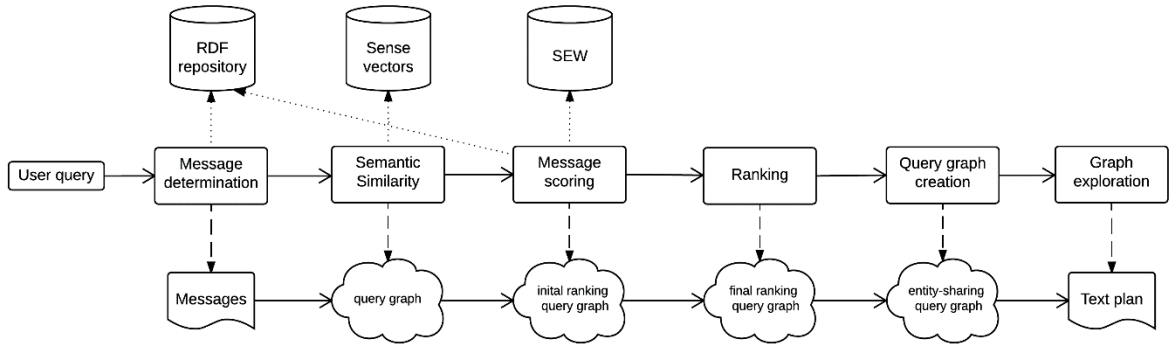

Figure 11: Architecture of the text planning module

The general architecture of the text planner is shown in Figure 11. A text plan is produced in response to a user query in which the user specifies an entity for which a summary must be produced. This summary should contain the most relevant information about in the MULTISENSOR semantic repository. The planner starts by extracting contents related to the

query that will act as atomic units for the remaining planning operations, henceforth referred to as *messages*. Following this message determination step, a query graph representation is built using a semantic similarity metric between messages. This metric is based on distributional sense vectors. The messages are then assigned with scores obtained from the content selection metrics, these scores constituting an initial ranking of the messages in terms of their relevance to the user query. This initial distribution of relevance is iteratively recalculated according to the connections between messages in the graph, until a final distribution or ranking is found. Finally, a new graph representation is created where links between ranked messages are based on entity-sharing relations, and the graph is explored from the highest ranking message in order to find a text plan that maximizes both relevance relative to the query and coherence.

The complete JAVA 8 source code of the text planner module can be found in **https://github.com/talnsoftware/textplanner**.

4.1.1 **The input data**

Abstractive summaries are generated from the output of the relation extraction module, which produces structured analysis of sentences and integrates the output of other modules in the CEP, namely the NER, EL and dependency parsing services. This output contains textual annotations modelled as standoff annotations using the NIF ontology. Figure 12 shows an example of a relational analysis of a sentence. Relational analyses take the form of forests in which each tree has as nodes individual words or phrases recognized as having an atomic meaning (e.g. multiword NEs). The edges of each tree indicate predicate-argument relations between them: all non-terminal nodes correspond to linguistic predicates that have as arguments their children in the tree. Nodes are associated with linguistic and semantic information such as lemmas, POS tags, syntactic dependency relations, BabelNet[7] synsets and FrameNet[8] frames. An example of the NIF-RDF encoding of all these annotations in the semantic repository is shown in Figure 13. The precise nature of this information and its RDF modelling are explained in greater detail in D2.3.

At the time of writing this deliverable, the semantic repository contains the RDF outputs of applying relational analysis to over 36.000 documents.

[7] http://babelnet.org/

[8] https://framenet.icsi.berkeley.edu/fndrupal/

```
:#char=1054,1061
      a                        nif:Word ;
      nif:anchorOf             "needs" ;
      nif:lemma                "need" ;
      nif:beginIndex           "1054"^^xsd:nonNegativeInteger ;
      nif:endIndex             "1061"^^xsd:nonNegativeInteger ;
      its:taClassRef           ms:GenericConcept ;
      its:taIdentRef           bn:s00091028v.
      nif-ann:taIdentConf      "0.789"^^xsd:Double ;
      nif-ann:taIdentProv      <http://babelfy.org/> ;
      nif:dependency           :#char=1062,1071 ;
      upf-deep:deepDependency :#char=1062,1071> ;
      nif:literalAnnotation    "number=SG|tense=PRES|time=applied|
                                manner=applied|modality=EPI/DYN|
                                place=applied" ;
      nif:oliaLink
            penn:VBZ ,
            upf-dep-syn:PRD ,
            upf-deep:II ,
            :#char=1054,1061_ annoSet [
                  a                        fn:AnnotationSet ;
                  fn:annotationSetFrame    fn-upf:Linguistic_situation ,
                                           frame:Needing ;
                  fn:annotationSetLU       lu:need ;
                  fn:hasLayer              :#char=2384,2389_Fes .
            ] ,
            :#char=1054,1061_fe [
                  a           fn:Label ;
                  fn:FE       fe-upf:Argument1.linguistic_situation .
            ] ;
      nif:referenceContext     :#char=0,3796 .
```

Figure 12: NIF-RDF encoding of annotation of the word *needs*

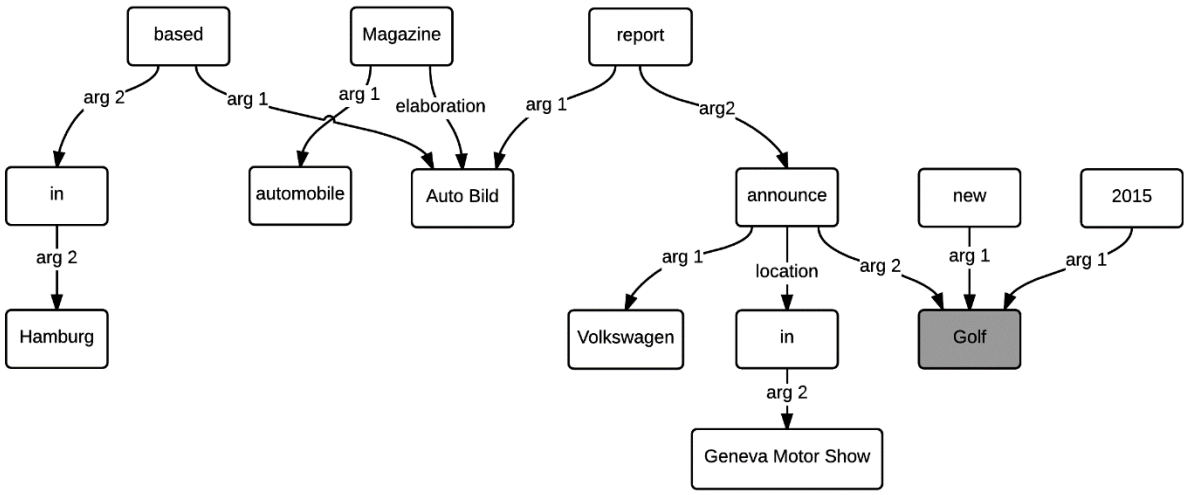

Figure 13: Relational analysis of the sentence “The Autobild automotive magazine, based in Hamburg, reported that Volkswagen will announce the new 2015 Golf in the Geneva Motor Show”

4.1.2 **Message determination**

In NLG literature, text planning is often seen as operating on messages that are understood as atomic units of meaning that combined form text plans. Depending on the data source, a message can be anything from a statement in a knowledge base or a cell in a relational database, to an observed change in a continuous signal. Messages should carry enough meaning to be considered semantic propositions (i.e. statements which can be assigned truth values), which rules out using as messages plain references to entities, or numerical values without further context.

Considering the input to the MULTISENSOR text planner, the n-ary relations obtained from the relation extraction service are a good fit for messages. The meaning of most relations cannot be correctly interpreted in isolation from the text they were extracted from, however. This is because their meaning depends on other parts of the sentence or large document they were extracted from. Given a relation, some of its arguments may be other relations that have no complete meaning without their respective arguments. Additionally, other arguments may be anaphoric or cataphoric expressions the meaning of which can only be resolved by taking into account other text fragments. Even if the meaning of all arguments is well defined, the relation be argument to another relation connecting it to some text fragment that provides the required context to correctly interpret it.

This is an inherently difficult problem that would require further linguistic and semantic analysis beyond the scope of the MULTISENSOR project, i.e. coreference resolution, discourse analysis and reasoning techniques. Nevertheless, we have adopted a simple message determination strategy that splits the analysis forest of a sentence into its constituent trees, as shown in Figure 14.

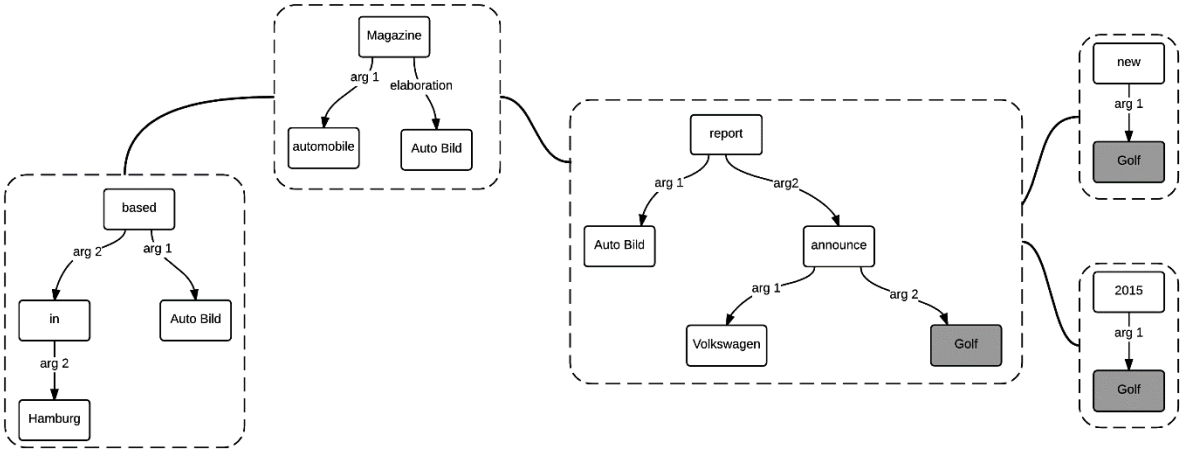

Figure 14: Sentence analysis from Figure 13 split into messages connected through entity-sharing relations

4.1.3 **Creating an initial text plan**

The message determination strategy yields a set of messages from sentence analysis. Sentences may come from different documents in the semantic repository. This set of messages that make direct mention to the user-requested entity constitute an initial text plan of relevant contents. However, there may be many other contents in the semantic

repository that are relevant to the queried entity despite not making direct reference to it. Text planning incrementally builds a query graph of contents potentially relevant to the query by exploring additional contents in the semantic repository from this initial set of messages.

This exploration for contents is driven by entity-sharing relations between messages already included in the plan and other relations. More precisely, given the initial set of messages and the set of BabelNet synsets it makes reference to, additional messages are determined from sentence analyses that also make reference to these synsets, the resulting messages being added to the plan. For instance, if a user requests a summary about the *VW Golf car*, three of the messages in Figure 14 will be initially included in the plan as they make direct reference to the Golf car. After this, the other two messages in which the sentence analysis has been split will also be included in the plan as they share mentions to the entity *Auto Bild*, along with other messages from other sentences also making reference to *Auto Bild*.

This process is iteratively repeated by finding more statements that mention entities referred from the last batch of messages added to the plan, up to a fixed depth. In order to prevent the size of the text plan growing too large with each iteration, the search for new messages is restricted to the sentences of either a single document, or to those documents that make direct reference to the requested entity. By entities we mean the BabelNet sysnets annotated to words and phrases in messages. However, we ignore any BabelNet synsets, which are not associated to nouns or definite non-relative verbs. In the example of Figure 14, this rules out as entities semantic prepositions like *in* and gerunds, participles and verbs in base form like *based*.

4.1.4 **Scoring the set of messages**

The content selection metrics described in section 3 are used to assign scores to the set of messages that constitutes an initial relevance ranking over the contents in the text plan. While the initial metrics produce scores for pairs of entities, we approximate the score for full messages by averaging the relevance score of each of the entities mentioned in a message relative to the queried entity. As in the exploration of the semantic repository, we restrict the entities used to calculate the average score of a message to those, which annotate nouns and definite non-relative verbs only.

4.1.5 **Using sense embeddings as a measure of semantic similarity**

While the corpora used for obtaining content selection metrics has a good coverage of the synsets annotated in documents belonging to the MULTISENSOR use cases, many pairs of a queried entity and an entity mentioned in a message will have score of zero, and by extension many messages are likely to have zero or nearly zero relevance scores. Not all these messages are necessarily irrelevant, as some may communicate similar information to messages with higher scores using different but semantically or pragmatically related entities. In order to capture this semantic relatedness between pairs of messages, we have designed a measure of semantic similarity that uses the edit distance between ordered labelled trees by (Zhang, 1996) to compute the number of changes needed to transform one message tree to another. When calculating the tree edit distance, messages are viewed as trees where leaves are sorted according to their offsets, in other words, according to the order in which they appear in the text.

Our measure assigns variable costs to rename operations based on the distance between pairs of BabelNet synsets. We obtain this distance by comparing the distributional vectors in a modified version of the SenseEmbed vectors[9] (Iacobacci et al., 2015). While the original SenseEmbed vectors contained entries for pairs of word forms linked to their disambiguated BabelNet synsets, we averaged the vectors over all word forms associated to the same synset in order to obtain vectors for synsets only. The original version of SenseEmbed has 1.734.862 vectors of 400 dimensions obtained from annotating with Babelfy[10] (Moro et al., 2014) the same late-2014 dump of Wikipedia used to obtain the SEW corpus. The modified version contains 653855 vectors, each one associated to a BabelNet sense. The computation the distance between sense vectors is done following a similar approach to the method to compare word-sense vectors described in (Iacobacci et al., 2015). We calculate the cosine distance between vectors and adjust the resulting distance according to a graph vicinity factor of 1.6 that rewards senses semantically connected in BabelNet, while penalizing others.

4.1.6 **Ranking the messages**

The set of messages in a text plan can be interpreted as a *query graph* where nodes correspond to individual messages and are connected in the graph through edges weighted with their semantic similarity, calculated as described in the previous subsection. We use this weighted undirected graph to transform the initial distribution of scores over messages into a stationary distribution of score that accounts for the semantic similarity between nodes. In other words, we redistribute relevance mass from those nodes assigned with a high score by the content selection metrics to other nodes connected to them in the query graph and according to the similarity weights associated to them in the graph. This process is akin to focused WPRAs, e.g. the *Directed Surfer Model* of (Richardson and Domingos, 2002), the *Focused PageRank* and *Focused HITS* algorithms described in (Diligenti et al., 2004), and the ontology summarisation algorithm *Focus Weighted PageRank* proposed in (Zhang et al., 2007).

The pseudocode for our ranking algorithm is shown in Figure 15. The algorithm starts from the transition matrix of a query graph and an initial distribution over the nodes of the graph. The algorithm performs a power iteration by multiplying the distribution to the transition matrix until the observed changes in the distribution fall below a fixed threshold, moment in which the algorithm is considered to have reached convergence to a quasi-stationary distribution. This distribution is the final ranking of messages produced in this step of the planning task.

4.1.7 **Exploration of the query graph**

While the ranking of messages could be applied to select the list of most relevance contents to be communicated in a summary, we also aim at maximising the coherence in the summary, finding a sequence of messages that maximises relevance while satisfying coherence constraints. More specifically, the coherence constraint that we want to satisfy is that no message can be added to the summary unless at least one of the entities it makes

[9] http://lcl.uniroma1.it/sensembed/

[10] http://babelfy.org/

reference to, has already been introduced by a previous message in the summary. In order to compute a sequence of contents that satisfies this constrain while maximising coherence, we build a new weighted query graph where nodes correspond to messages, weights are assigned to them according to the final ranking, and edges are added between each pair of nodes that shares at least one mention to the same entity. We then perform a greedy exploration of this query graph starting from the node with the highest rank. The pseudocode for this exploration algorithm is shown in Figure 16. The sequence of nodes visited during the exploration constitutes the output of the text planner module and the input to the linguistic realisation module. The length of the summary can be adjusted by removing messages from the end of the sequence.

```
algorithm power_rank is
    input:  transition matrix P
            initial distribution W_0
            convergence threshold k
    output: stationary distribution W

W_i ← W_0
δ ← 1.0
while (δ>k) do
    W_{i+1} ← W_i P
    δ ← max(abs(W_{i+1} − W_i))
W ← W_{i+1}
return W_{i+1}
```

Figure 15: Pseudocode for the ranking algorithm

```
algorithm greedy_exploration is
    input:  undirected weighted graph G = (V, E, W)
            node v ∈ V
    output: sequence of nodes in V

S ← ∅
V_v ← G.adjacentSet(v)
S_v ← sort(V_v)
foreach v_adj in V_v
    S ← S ∪ greedy_exploration(G, v_adj)

return S
```

Figure 16: Pseudocode for the query graph exploration algorithm

5 ADVANCED SUMMARY DELIVERY SYSTEM

In this section, we describe the surface generation pipeline that takes as input the text plans as described in the previous section, in order to produce an abstractive summary.

5.1 Surface generation pipeline

As described in D6.1, the generation is performed step by step, by successively mapping one level of representation onto the adjacent one: *Semantic Structure -> Deep-Syntactic Structure -> Surface-Syntactic Structure -> Morphologic Structure -> Linearized Structure -> Sentence.* The underlying linguistic framework is the Meaning-Text Theory (Mel'čuk, 1988), which foresees a very similar stratification in language description. In this subsection, we describe the role of each transition.

5.1.1 From Semantic Structure to Deep-Syntactic Structure (DSyntS): lexicalization and sentence structuring

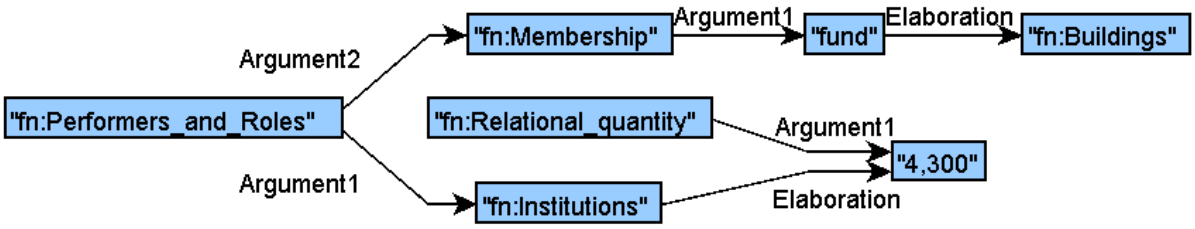

Figure 17: A sample Semantic Structure for the sentence *4,300 institutions are part of the pension fund*.

During this transition, the semantic graph that is outputted by the Text Planning module (see Figure 17) is assigned a communicative structure and mapped onto a tree. The part of the graph that constitutes the topic of the sentence to be generated is identified –and called "theme"-, as well as what is said about that topic, which is called "rheme". The main node of the rheme will be the head of the sentence, that is, the main verb, while the theme generally corresponds to the syntactic subject and the rest of the rheme to the objects and adverbs. From this root, the whole tree can be built node after node.

Only meaningful units (**lexemes**) are part of the DSyntS; in other words, there are no grammatical units at this point (governed prepositions, auxiliaries, etc.).

Figure 18, shows that "be" is the main syntactic node of the sentence, and that all other elements are organized around it. Instead of a pure predicate-argument structure, the edge labels reflect the syntactic structure of the sentence, in particular the opposition between arguments (*I*, *II*, *IV*) and the modifiers (*ATTR*).

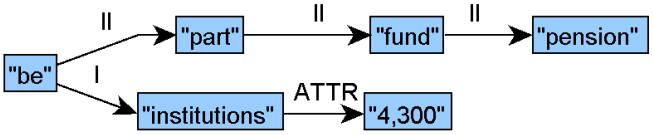

Figure 18: A sample deep-syntactic structure corresponding to Figure 17

5.1.2 From Deep-Syntactic Structure to Surface-Syntactic Structure (SSyntS): introduction of idiosyncratic information

Once the structure of the sentence has been defined, non-meaningful, or *grammatical* units have to be introduced. In our language-specific lexicons, an entry of a word indicates which preposition, case, finiteness, number, etc. has to be inserted on its dependent. For instance, if there is the DSynt configuration *part-II-> X*, *of* is added in the SSynt tree because in the lexicon, the entry of *part* indicates that the second argument must be introduced by the preposition *of*, which is then a governed preposition. Other non-lexical nodes are introduced, such as governed conjunctions, auxiliaries, determiners, expletive subjects, etc.

Finally, the generic syntactic relations found in DSynt are refined into more idiosyncratic relations which convey very accurate syntactic information, instead of semantic, as it is the case with the argument numbers. For instance, the DSynt relation 'I' can be mapped to *SBJ* (subject) if the verb is active, *OBJ* (object) if the verb is passive, *NMOD* if the head is a noun, etc. A *SBJ* has the syntactic property to trigger an agreement on the verb, to undergo demotion in some conditions, and to be realized before the verb in a neutral sentence. An *OBJ*, on the contrary, appears by default after the verb, can undergo promotion, and is cliticizable with an accusative pronoun (the last two properties are restricted to direct transitive verbs). A *NMOD* cannot be promoted or demoted, does not trigger any agreement and always has to be realized to the right of its governor.

Figure 19 shows the surface-syntactic structure with functional words and language-specific syntactic relations.

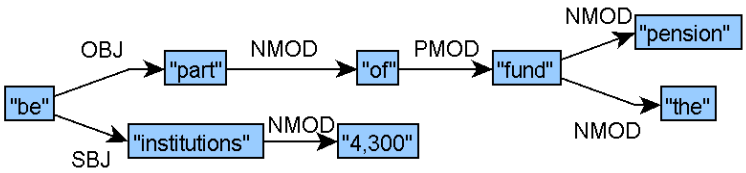

Figure 19: A surface-syntactic structure that corresponds to the deep-syntactic structure in Figure 18

5.1.3 From Surface-Syntactic Structure to Morphologic Structure (MorphS)

Thanks to the idiosyncratic set of surface-syntactic relations, all agreements and orders between the components of the sentence can be resolved. Every word of the sentence contains all the indications to make the production of the final form possible. This can be done either by creating a full-fledged dictionary under the form "be<VB><IND><PRES><3><PL> = are", or by using some automata such as Two Level Morphology, which automatically inflects forms, based on inflection schemas. Upper case can be introduced when necessary.

Thanks to the idiosyncratic set of surface-syntactic relations, all orderings between the components of the sentence can be resolved; for instance, a subject goes before its governing verb, a determiner before its governing noun, etc.

Figure 20 shows that at this level, the words carry all the necessary information for inflection (e.g. the attribute editor for the word "be"): part-of-speech, mood, tense, person, and number. The precedence relations are shown in red.

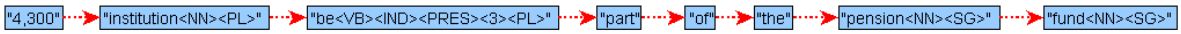

Figure 20: A (linearized) morphological structure that corresponds to the surface-syntactic structure in Figure 19

5.1.4 From Morphologic Structure to Sentence

Once all the words are ordered, punctuation marks are introduced, the final form of the words is retrieved and the sentence is ready to be delivered to the next module. In the case of the running example shown throughout this section, the output would be "*4,300 institutions are part of the pension fund.*"

5.2 Tools and resources

As stated in D6.1, for surface generation, two different approaches are foreseen, both based on a pipeline of graph transducers that convert the output of the text planning stage into a well-formed text. The first approach consists in manually crafting graph-transduction grammars for each transition between two consecutive layers, while the second approach consists of performing each transition with statistical modules trained on annotated data

In this subsection, we describe the new graph transduction environment developed in the framework of the project, the lexical resources used for rule-based generation and the annotated resources for the statistical modules.

5.2.1 New graph transduction environment

We have been re-implementing the graph transducer MATE (Bohnet and Wanner, 2010) in order for the transduction rules to be more expressive and compact, as well as for the tool to perform the transductions faster; we will refer to this new tool as "MATE-2" in this section.

MATE is a graph transducer programmed in Java. It contains different editors for graph construction, rule and lexical resource writing, a debugger, as well as a tool for regression tests. The rules (and their corresponding conditions) match a part of an input graph, and create a part of the output graph. The main problem with MATE was that the conditions stated in the rules were limited and forced the developers to end up with complex rule systems where one single rule would suffice with a different implementation, which is why we needed to develop MATE-2. We took advantage of this reimplementation for removing known bugs and improving the speed and usability of the tool.

MATE-2 is currently in a very advanced state but no publication describes it yet. Figure 21 shows a project open in MATE-2.

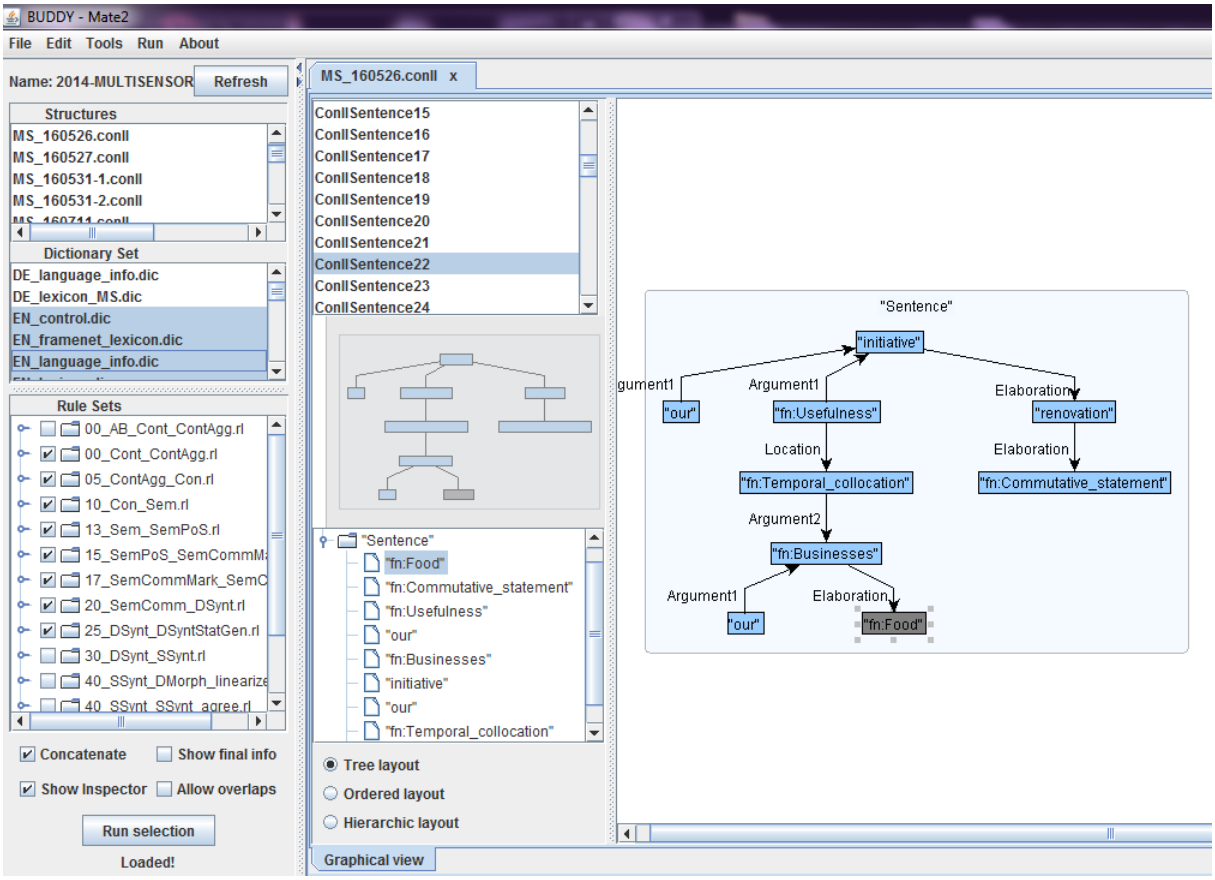

Figure 21: A screenshot of MATE-2 (Graph Editor view)

MATE-2 contains:

- A Project Browser

  It is used to open and navigate the different resources of a project. It's the part on the left, which contains 3 list fields, some options and the Run Selection button. Each list field corresponds to a resource used for the transduction:

  - Structures: a list of input structures on which the rules can be applied in order to create new structures.
  - Dictionary Set: a list of lexical resources used by the rules.
  - Rule Sets: a list of rule sets (= grammars); each rule set performs one transduction.

- A Resource Editor

  Each of the three resources can be opened in an editor tab, by double clicking on it. The Graph editor contains 5 fields (see Figure 21):

  - Graph List: the list of graphs contained in one file (top left)
  - Graph Global View: the global view of the selected graph (middle top left)
  - Graph Node List: the list on nodes of the selected graph (middle bottom left)
  - Graph layout options (bottom left).

- Graph View: the complete graph view of the selected graph (right)

The Rule editor contains 2 main fields and some parameters (see Figure 22):

- Rule List: the rule tree (left)
- Rule View: the complete rule view of the rule selected in the rule tree (right).

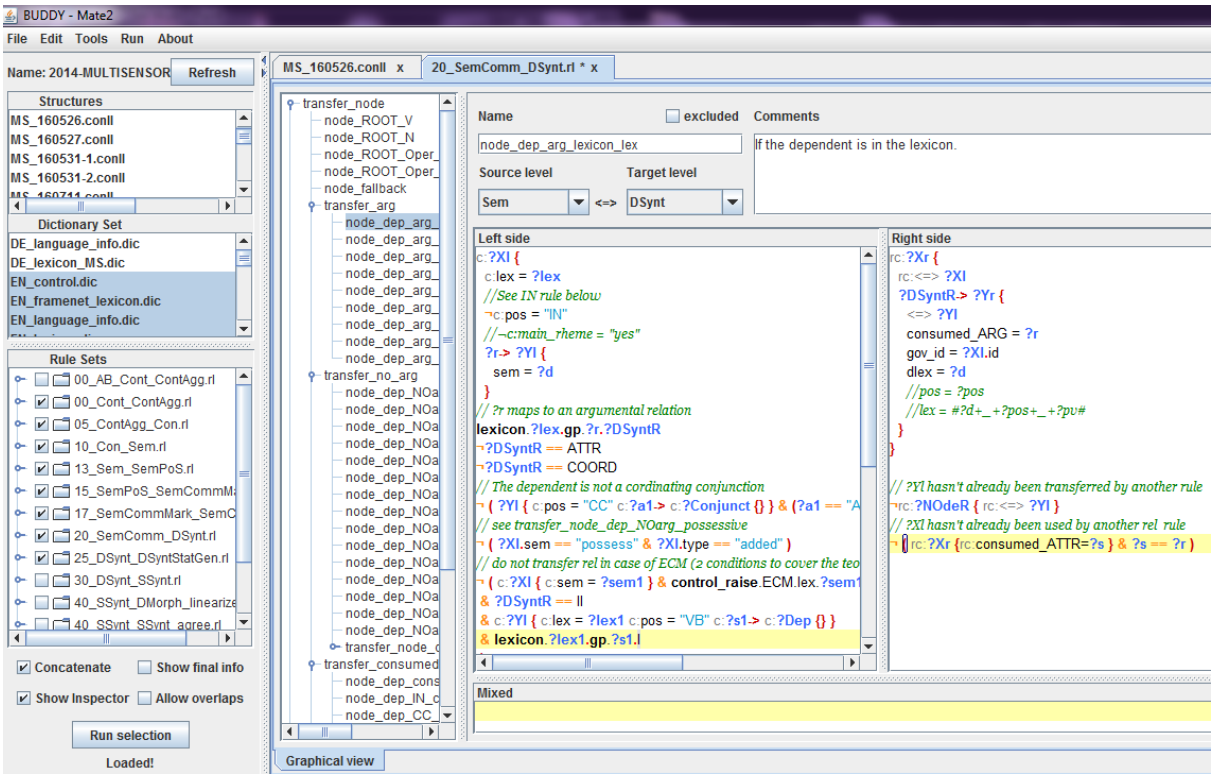

Figure 22: A screenshot of MATE-2 (Rule Editor view)

The Lexicon editor does not have an advanced view at this stage and only is available in textual mode.

5.2.2 **Lexical resources**

As explained in D2.4, for the Content Extraction Pipeline, we created lexical resources for each language covered by the system. We designed these resources in a way that they can be used for analysis as well as for generation. For the sake of clarity, in this subsection, we repeat (and slightly adapt) the description of the lexicons made in D2.4.

The lexical resources must not only be monolingual, but also be somehow linked to each other, in order to allow for the mapping from English to each of the target languages. Given that BabelNet senses annotated during the analysis stage are language-independent, we use them as the cross-linguistic link needed for the multilingual generation pipeline. Below, we detail the creation procedure and structure of the language-specific lexicons used to go from predicate-argument structures and BabelNet synsets to each language, tested using as basis two texts in each language -Spanish, French and German- of the energy policies domain (around 3,500 words).

The compilation of the language-specific lexicons was done in different stages. Given that word sense ambiguity is a problem inherent to any language, it was necessary first to

disambiguate and recognize the right sense of a lexical unit (LU) before assigning any specific BabelNet (Navigli et al., 2012) id to it. The WSD tool BabelFy (Moro et al., 2014), which is connected to BabelNet, was used to deal with this problem: using the API offered to remotely access the service, the selected texts were passed through BabelFy. As output of this step, a list of non-duplicated BabelNet ids (1,013 items in total) was obtained, which served as the basis for testing the lexicons. This list was locally enriched with the word form linked to each id in each language, in order to facilitate and accelerate the manual compilation of units.

For each LU, we store in the corresponding lexicon entry its part of speech (which refers to a most general entry), lemma, BabelNet synset and government pattern –the elements required by the unit, i.e., the subcategorization frame of the LU- are stored. Within the government pattern, the information collected for each argument includes its part of speech, the preposition introducing it –if it is required by the described LU- and the corresponding case. As an example, we show below the entry for the same specific BabelNet id in German (a language with case) and in Spanish:

SPANISH

```
"contar_VV_01":_verb_ {
lemma = "contar"
bn = bn:00091011v
gp = {
   I = {dpos = "N"}
   II = {}
   III = {dpos = "N" prep = "a"}}}
```

GERMAN

```
"sagen_VV_01":_verb_ {
lemma = "sagen"
bn = bn:00091011v
gp = {
    I = {dpos = "N" case = "nom"}
    II = {dpos = "N" case = "acc"}
    III = {dpos = "N" case = "dat"}}}
```

So, from the structure with English node labels, the system turns to the lexicons to obtain information about the specific characteristics of the sentences to be generated in each language. If no specific information is added (as in the second argument for Spanish), the system interprets that are not restrictions with respect to such argument (e.g. the second argument in Spanish could be a noun, but also a subordinated verb). Thus, the four parallel language-specific lexicons compiled serve in a direct way to the multilingual generation pipeline, allowing the mapping from English to any of the other languages included. Potentially, even, the mapping could be done not only from English to other language, but from any other language included in the system to each other.

5.2.3 **Multilingual Dataset**

We worked on the parallel annotation of morphologic and surface- and deep-syntactic structures; see Figure 20, Figure 19 and Figure 18 respectively.

All the layers (and the original text) need to be aligned sentence by sentence and node by node, which can be done thanks to unique identifiers associated to each sentence and node. The morphologic annotation consists of features associated to each word of the sentence, including coarse-grained and fine-grained part-of-speech, gender, number, tense, aspect, finiteness, mood, person, etc. On this layer only, the order of the words is kept. The surface-syntactic annotation consists of dependency trees with all the words of a sentence linked by idiosyncratic relations. At the deep-syntactic layer, all functional (i.e. non-meaningful) units are removed: definite and indefinite determiners and auxiliaries are replaced by

attribute/value pairs on the concerned nodes, while punctuations and governed prepositions and conjunctions are simply removed. In addition to attributive and coordinative relations, the dependency relations also encode predicate-argument information, through the assignment of an argument slot in the valency (*subcategorization framework*) of its governor predicate.

We have available multi-layered corpora for Spanish, English, French and German. In English, we use the Penn Treebank 3 (Marcus et al., 1994) converted to dependency trees (Johansson and Nugues, 2007), which we use as such as surface-syntactic and morphologic annotations. We automatically derived a first version of the deep annotation from the surface annotation using graph transduction grammars implemented in the MATE environment (developed by TALN-UPF). During the mapping, we removed all determiners, auxiliaries, *that* complementizers, infinitive markers *to*, punctuations and functional prepositions. The treebank contains 41.678 sentences (1.015.843 tokens at the surface-syntactic and morphologic layers, 768.865 tokens at the deep-syntactic layer).

In German and French, we follow the same method as for English; we start with the existing surface-syntactic corpora (respectively the TIGER corpus (Brants et al., 2002) and the French Dependency Treebank (Abeillé et al., 2003)) from which we automatically derive the deep-syntactic annotation thanks to graph transduction grammars. Since the surface-syntactic annotation is quite different from the English one–the dependency tag set and the annotation scheme are not the same – it is not possible to re-use the English mapping. We did not review manually the conversions in these languages.

For Spanish, we use the AnCora-UPF dependency Treebank (Mille et al., 2013)11. In the AnCora-UPF, each layer is already independently annotated, and the annotation has been manually validated, with a high inter-annotator agreement. For the surface-syntactic annotation, there are several dependency tag sets available, with many or few tags. The treebank currently contains 3.513 sentences (100.892 tokens at the surface-syntactic and morphologic layers, 65.889 tokens at the deep-syntactic layer.

For linearisation and punctuation, we use the surface-syntactic annotations in the CoNLL'09 format (Hajič et al., 2009), since they contain ordering and punctuation information.

5.3 Implementation

As mentioned in the previous section, we foresaw two different implementations of the generation pipeline, one rule-based, and one based on machine learning. However, in the final version of the system, not all transitions have a statistical counterpart. This is due either to the fact that no training data is available (e.g., for the definition of the communicative structure, no large corpus is annotated with such information), or that machine learning does not make sense (as, e.g., for the resolution of morphological agreements, for which rules are sufficient in order to obtain good results).

As a consequence, the final generation pipeline in MULTISENSOR is hybrid, with a combination of rule-based modules and statistical components.

11 The corpus underlying this treebank is the 2008 version of AnCora (Taulé et al., 2008).

5.3.1 Semantic to deep syntactic mapping

The mapping from the output of Text Planning and deep-syntactic structures comprises 4 steps, each of which implemented as a graph transduction grammar in the MATE environment.

A first grammar ensures that all words have a connection to the lexicon of the given language, and if not predicts a part of speech and suggests a disambiguation identifier.

The second grammar looks for possible roots for the syntactic tree to be built: verbs with arguments, if any, and if not, adjectives adverbs or nouns. Only nodes that are not too embedded in the semantic graph can be candidates for being syntactic roots.

The third grammar compares all possible candidates and chooses the one that "looks better", that is, the one that has more impact on the other nodes in the graph. The notion of impact for a node N is based on a mixture of number of nodes below N and type of nodes surrounding N in the graph. If the distance between N and the most distant leaf node on N's branch is important, N is said to have more impact. If N does not have any governor in the semantic graph, or if N's governor(s) in the tree are of a type that can never be a root, N is said to have more impact. For this reason, the node *fn:Performers_and_roles* in Figure 17 is the best root for the sentence: it has no governor in the graph, and the distance until the most distant leaf (*fn:Buildings*) is larger than the distance between any two other nodes in the graph.

Finally, the fourth grammar unfolds the syntactic tree starting from the root found in the previous step. The main challenge is to map the semantic relations onto deep-syntactic relations. A first semantic argument of a verb or a noun is generally mapped to a first deep-syntactic argument (this kind of mapping is available in the language-specific lexicon); the first argument of a number in the semantic graph becomes its deep-syntactic governor, and the relation is labelled *ATTR* (e.g. *4,300* and *fn:Institutions* in Figure 17); etc. We identified around 30 different types of mappings between semantic and deep-syntactic configurations. The grammar is able to handle structural mismatches between the two; for instance, control or raising verbs are embedded differently in the semantic and in the syntactic structure. If the grammar detects that the root cannot be a verb in a language in which a verb must be the root of the syntactic tree, the grammar is able to introduce support verbs; for instance, in the case of a semantic graph *successful-Argument1-> company*, a verb *be* would be introduced in order to get to *the company is successful*.

These four grammars are largely language independent; only about 10% of the fourth grammar must be specific to a particular language, in order to adapt to particular semantic configurations.

5.3.2 Deep to surface syntactic mapping

We used, and implemented in the context of the project, the statistical deep sentence generator (Ballesteros et al., 2015) that successfully handles the non-isomorphism between deep-syntactic trees and surface-syntactic structures in terms of a machine learning approach. The model is based on a pipeline of support vector machine classifiers that solve first local decisions while it produces a global output.

In order to project a DSyntS onto its corresponding SSyntS in the course of generation (where both DSyntSs and their corresponding SSyntSs are stored in the 14-column CoNLL'09 format (Hajič et al., 2009)), the following types of actions need to be performed by the statistical generator:

1. Project each node in the DSyntS onto its SSynS-correspondence. This correspondence can be a single node, as, e.g., *job*->[*NN*] (where *NN* is a noun), or a subtree (hypernode, known as syntagm in linguistics), as, e.g., *time*->[*DT NN*] (where *DT* is a determiner and *NN* a noun, as in *the time*) or *create*->[*VAUX VAUXVB IN*] (where *VAUX* is an auxiliary, *VB* a full verb and *IN* a preposition, as in *to have been created*). In formal terms, we assume any SSyntS-correspondence to be a hypernode with a cardinality ≥1.
2. Generate the correct lemma for the nodes in SSyntS that do not have a 1:1 correspondence with an origin DSyntS node (as *DT* and *VAUX* above).
3. Establish the dependencies within the individual SSyntS-hypernodes.
4. Establish the dependencies between the SSyntS-hypernodes (more precisely, between the nodes of different SSyntS-hypernodes) to obtain a connected SSyntS-tree. For Spanish, we apply after the DSyntS–SyntS transition rules for the generation of relative pronouns that are implied by the the SSyntS. Since we cannot count on the annotation of coreference in the training data, we do not treat other types of referring expressions. The lemmas of nodes with 1:1 correspondence are the same in both structures.

In order to foresee the absence of training data in some languages, we also developed a rule-based grammar that performs the same transduction. In this case, generic (language-independent) rules check the language-specific lexicon and retrieve the nodes to be added in the surface-syntactic structure, and the labels of the surface-syntactic edges.

The models and JAVA source code for the statistical deep sentence generator can be found in **https://github.com/talnsoftware/deepgenerator**.

5.3.3 **Morpho-syntactic agreement**

The morpho-syntactic agreements are resolved with a graph transduction grammar that contains rules that check the agreements according to the different languages. Since the agreements can be quite different according to the language, these rules are mainly language specific. For instance, in English, adjectives are invariant, while in French and Spanish, they take their number and gender from the governing noun, and in German for instance, they also take a case from it. All the morphological attributes necessary for inflecting the words are percolated to the node of said word, for further resolution. In the case of *be* in Figure 20, its morphologic attributes are *be<VB><IND><PRES><3><PL>*: the part of speech (verb), the mood (indicative) and the tense (present) come from the sentence itself, while the person (3) and the number (plural) come from the agreement with the subject (institutions).

5.3.4 Morphological realisation

The morphological generation or realisation is done by using a dictionary of entries with their corresponding morphological features. A large amount of raw text has been tagged with morphological tags, using Bohnet et al.'s (2013) tagger. We retrieved all the unique combinations of lemmas with their morphological tags and their final form, and stored them in a large dictionary. The algorithm is just a mapping in which the key is the lemma with its corresponding tag, and the value is the desired inflected form: *be<VB><IND><PRES><3><PL> = are*. When the lemma with its tag is queried, the form is returned.

5.3.5 Linearisation

The linearization is performed with Bohnet et al.'s (2012) lineariser. It is based on a bottom-up algorithm based on the syntactic tree. The lineariser is a machine learning approach that uses beam-search to find the best tree possible.

5.3.6 Generation of punctuation

We used and defined in the context of the project a sequential algorithm that introduces punctuation marks into sentences that do not contain any punctuation. In the context of MULTISENSOR, the input sentence would be the result of the surface realisation task or linearization. The algorithm uses two data structures: the *input buffer* and the *output buffer*. The algorithm starts with an input buffer full of words and an empty output buffer. The two basic actions of the algorithm are *shift*, which moves the first word from the input to the output, and *generate*, which introduces a punctuation mark after the first word in the output.

The representation of the input and the output buffers is similar to the one used by Ballesteros et al. (2016). In particular, we use a long short-term memory network (LSTM) for the input and the same for the buffer. A LSTM is a recurrent neural network that is better remembering long and short-term information. The source code for the algorithm can be found in **https://github.com/miguelballesteros/LSTM-punctuation**.

6 EVALUATION

6.1 Evaluation of extractive summarization

The final version of the extractive summarization modules has been evaluated separately for each of its modalities, single document, query-based and multiple documents. The evaluation metric reported, as in previous versions of this deliverable and according to the self-assessment plan in D1.1 and D2.1, is N-gram ROUGE (Lin 2004).

6.1.1 Evaluation of single document summaries

For the evaluation of single document summaries, we use one of the datasets in the corpus from the 2001 edition of the Document Understanding Conference (DUC)[12]. This dataset is composed of 200 documents taken from news articles and divided in two parts, one consisting of 180 documents for training and the remaining 20 for evaluation. The training part has manually authored abstractive summaries which we use as the reference set for our evaluation. The choice of this corpus is motivated by the fact that the summaries are not crafted with specific keywords or topics in mind, but rather provide a generic gist of the original articles.

We have evaluated several versions of the single document pipeline using the semantic features for summarization already reported in D6.2. Each version of the pipeline uses its own IDF table obtained from the analysis with the CEP of the source documents in the DUC corpus. Results can be seen in Table 2, where column *feature* indicates the different features extracted from sentences in order to rate them.

Feature	ROUGE-1	ROUGE-2
Lemma	**0.54**	**0.25**
HeadBnIdRel	0.55	0.27
HeadFrameRel	0.55	0.26
BnId	0.55	0.26
Mix	**0.57**	**0.27**
Frame	0.48	0.21

Table 2: ROUGE-1 and ROUGE-2 n-gram recall obtained using different linguistic features to compute sentence similarity

The features used are:

- Lemma: base form of a word
- BnId: BabelNet senses assigned to words or phrases
- Frame: FrameNet frames assigned to predicative words
- Mix: Frame or BabelNet sense of words and phrases, with frame annotations taking precedence when both are available for a given word.
- HeadBnIdRel: BabelNet sense of the argument and lemma of the governor

[12] http://www-nlpir.nist.gov/projects/duc/data.html

- HedadFrameRel: BabelNet sense of the argument and frame of the governor

Some other combinations including governors, dependents and relations were also tested but the results obtained were lower than listed above, so they are not reported here. The results indicate that bag-of-word representations work better than features reflecting predicate-argument structure. The best performing feature is to use both FrameNet frames and BabelNet senses (*mix*) obtain the best results outperforming the results obtained by using only lemmas. However, the difference with using lemmas is small. Our guess is that this is caused by errors in the disambiguation of words against BabelNet and FrameNet introduced by EL (Babelfy) and the relation extraction modules of the CEP.

6.1.2 **Evaluation of Query Based summaries**

For the evaluation of the query-based pipeline, we have compiled a corpus of 88 pairs of documents and abstractive summaries belonging to UC 1.1 and a corpus of 100 pairs corresponding to UC 1.2. The summaries in the first set come from the pressrelations News Radar dataset and were tailored for the company Electrolux, while the summaries in the second dataset were created by Deutsche Welle in the scope of the MULTISENSOR corpus and we tailored for the energy policies topic. System summaries have been generated using keywords *Electrolux* and *energy policies* respectively.

Use case	ROUGE-1	ROUGE-2
UC 1.1- Household appliances Query: Electrolux	0.73	0.65
UC12-Energy Policies Query: Green Energy	0.41	0.20

Table 3: ROUGE-1 and ROUGE-2 results obtained on query based summaries for the two different use cases

Table 3 shows the results of the evaluation using N-ROUGE metric. The differences between the two use cases are likely to be originated in the nature of the gold standard summaries. While summaries for UC 1.1 are regular in length and contain many fragments copied verbatim from the original texts, summaries in UC1.2 can vary greatly in length depending on whether the author saw many or few facts relevant to Electrolux, and in most cases were highly rephrased versions of the originals. In any case, even UC 1.2 results are in line with the state of the art on query-based summarization, e.g. ShafieiBavani et al. (2016) report maximum Rouge-1 levels of 0.37 on the 2005 DUC dataset and 0.44 on the 2007 DUC dataset.

6.1.3 **Evaluation of multi-document summarization**

For the evaluation of multi-document summarization, we have used the training part of the multi-document dataset of the 2001 DUC corpus, which is composed of 27 sets of 10 documents and abstractive summaries of different lengths (50, 100, 200 and 400 words) for each set. We have evaluated our system against the 200-word summaries. For this reason,

we set the parameters of the summarization pipeline to produce a fixed length summary instead of doing a percentage reduction.

Use case	ROUGE-1	ROUGE-2
UC 1.1- Household appliances Query: Electrolux	0.38	0.06

Table 4: ROUGE-1 and ROUGE-2 results obtained on multiple-document summaries.

Table 4 shows the ROUGE-and ROUGE-2 values obtained. While the results are pretty law, multiple document summarization is a notoriously challenging task, as testified by results reported in state-of-the-art systems, e.g. Cao et al. (2015) reports ROUGE-1 and ROUGE-2 for different systems that range from 0.33 to 0.37 ROUGE-1 to 0.06 to 0.08 for ROUGE-2.

6.2 Evaluation of concept-based summarization

The abstractive summarization functionality in the MULTISENSOR prototype is evaluated using multiple metrics, as planned in D1.1 and reported in D1.2. More precisely, the content selection metrics and text planning algorithm are evaluated using precision and recall-based metrics by comparing against a gold standard and a baseline. For multilingual generation, on the other hand, UAS and LAS metrics were foreseen. For the evaluation of the overall summaries, D1.1 foresaw using BLEU as a quantitative metric. In this document we report the closely-related ROUGE metric instead, as this makes it easier to compare the performance of the abstractive and extractive summarizers.

A qualitative evaluation of the summarization functionality using questionnaires is reported in D8.5, as part of the evaluation of the final system (see results on sections covering summarization functionalities).

6.2.1 Evaluation of content selection metrics and text planning

As explained in Section 4, the text planner operates on the RDF-encoded output resulting from processing documents with the CEP. The text plans it produces are mostly language-independent and serve as input to the linguistic generation module. Plans are composed of messages, which are tree-like structures obtained from the CEP analysis of textual contents in documents. Nodes in these trees correspond to either language-independent BabelNet synsets expressing word senses, or language-specific word forms where no sense was found for the word in question. Edges, on the hand, indicate deep syntactic or semantic roles that one node (the dependent) adopts in the situation indicated by anther node (the governor). The text planner uses the content selection metrics of Section 3 to evaluate the relevance of individual entities with respect to one or more reference entities selected by the user. The estimations produced by these metrics are extended to whole messages using a semantic similarity metric based on distributional vectors and a tree edit similarity algorithm.

Our evaluation procedure compares the entities and overall messages in system-produced messages with those in reference messages obtained from a gold standard of summaries. Entities in the messages of a pair of text plans are compared using precision, recall and f-

score, while whole messages are compared using the same similarity metric used by the text planner. Additionally, the summaries produced by the linguistic generator are evaluated using n-gram ROUGE, a metric which has been showed useful to evaluate not only extractive but also abstractive summaries (Steinberger and Ježek 2009).

We use the manually crafted extractive summaries of the CAST corpus[13] (Hasler et al. 2003) as a gold standard. Single-document summaries in CAST consist of sentence fragments from a source document, which have been annotated as either essential or important. The fact that the summaries are extractive contributes to make the entity and message-based evaluations less dependent of discourse structuring and linguistic generation issues that are better evaluated using qualitative methods, and more focused on the selection of relevance contents. On the other hand, having fragments annotated rather than whole sentences helps us to evaluate the success of our message determination strategy in identifying the most important parts of sentences.

The evaluation procedure consists in using the MULTISENSOR system to generate pairs of text plans and abstractive summaries in English for the 141 documents of the CAST corpus that belong to the news domain. Extractive summaries are also generated using the single-document, SUMMA-based, extractive summarization pipeline, which acts as a baseline. Both the baseline and gold summaries are processed with the CEP in order to obtain relational structures that can be compared to the text plans.

	Entity-based evaluation			Sem sim	N-gram ROUGE					
					1-gram			2-gram		
	P	R	F		P	R	F	P	R	F
extractive	0.49	0.38	0.42	0.41	0.55	0.73	0.62	0.55	0.60	0.52
abstractive	0.54	0.59	0.56	0.42	0.49	0.61	0.54	0.26	0.33	0.29

Table 5: results of the evaluations of the content selection metrics and text planning module.

The results of running the three types of evaluation described above for the baseline extractive and abstractive summarization modules against the gold standard are shown in Table 5. The columns to the left report results of comparing the entities in the system summaries compared to those in the gold summaries, in terms of precision (*P*), recall (R) and f-score (*F*). The central column reports the average similarity (normalized to the range [0,1.0]) between messages of the system and gold plans. The columns on the right provide results for the unigram and bigram ROUGE evaluation of the texts of the system and gold summaries. It is important to note that none of these metrics reflect coherence or linguistic-related quality, e.g. order of the contents, coherence, grammaticality, fluidity, etc.

From the results it can be seen that, in terms of N-gram overlap as estimated by ROUGE, the extractive baseline outperforms the abstractive baseline, especially when going from the bag-of-words view of unigrams to bigrams. In contrast to these results, the semantically

[13] http://rgcl.wlv.ac.uk/projects/CAST/corpus/

oriented evaluation using entities in word senses (entities) and the semantic similarity metric of the text planner indicate better performance of the abstractive approach. This divergence in the results is likely to be caused by the paraphrasing of contents with respect to the original contents that happens in the abstractive approach, as opposed to the extraction of whole sentences in the extractive approach.

6.2.2 **Evaluation of linguistic generation**

Due to the low coverage of the lexical resources and grammars and the poor performance of the disambiguation tools, the output summaries in other languages that English are mostly constituted of ungrammatical sentences; thus, we did not carry out an evaluation on all the languages, but only on English and Spanish material. With respect to the generation component, we carried out a two separate evaluations: (i) rule-based and machine learning systems from the Deep-Syntatic structures used as input to the abstractive summarization pipeline, and (ii) rule-based system from abstract predicate-argument graphs.

For the advanced version of linguistic generation, we performed experiments in Spanish and English, evaluating steps 1 to 4 depicted in Section 5.3.2, together with the linearization step (5.3.5). We used as baseline Bohnet et al.'s (2010) generator. The Spanish treebank, AnCora-UPF (Mille et al., 2013) has been divided into: (i) a development set of 219 sentences, with 3,437 tokens in the DSyntS treebank and 4,799 tokens in the SSyntS treebank (with an average of 21.91 words by sentence in SSynt); (ii) a training set of 3,036 sentences, with 57,665 tokens in the DSyntS treebank and 84,668 tokens in the SSyntS treebank (with an average of 27.89 words by sentence in SSynt); and a (iii) a held-out test for evaluation of 258 sentences, with 5,878 tokens in the DSyntS treebank and 8,731 tokens in the SSyntS treebank (with an average of 33.84 words by sentence in SSynt).
For the English treebank (Johansson and Nugues, 2007), and a UPF annotation of the deep-syntax), we used a classical split of (i) a training set of 39,279 sentences, with 724,828 tokens in the DSynt treebank and 958,167 tokens in the SSynt treebank (with an average of 24.39 words by sentence in SSynt); and (ii) a test set of 2,399 sentences, with 43,245 tokens in the DSynt treebank and 57,676 tokens in the SSynt treebank (with an average of 24.04 words by sentence in SSynt). In Table 6 and Table 8, we show the system performance on both treebanks.

	Semantics-Syntax (ULA/LAS)	
	English	Spanish
MULTISENSOR advanced	94.42/89.13	99.06/93.13
Bohnet et al. 2010	94.77/89.76	98.39/93.00

Table 6: Overview of the ULA and LAS results on the semantics-syntax interface

The results in Table 6 do not show the expected improvement compared to Bohnet et al.'s (2010) system (+3% UAS, +5% LAS); however, for Bohnet's experiment, only a small subset of functional elements were removed, while in MULTISENSOR, we removed all functional elements as found in the PropBank and NomBank lexicons, resulting in a representation in which a lot more words have to be introduced.

	BLEU score	
	English	Spanish
MULTISENSOR advanced	0.77	0.54
MULTISENSOR baseline	0.69	0.52
Bohnet et al. 2010	0.85	0.78

Table 7: Overview of the results on the English and Spanish test set excluding punctuation marks after the linearization

On the linearized output, the MULTISENSOR approach did not achieve any improvement on the Spanish or English dataset with respect to the BLEU score compared to Bohnet et al.'s (2010) system on English, but again, this largely due to the fact that the input structures contain more superficial features in the baseline, making MULTISENSOR more capable of generating from genuine semantic structures. Since the comparison between the two systems is difficult to interpret due to the difference of input structures they accept, we developed another baseline, under the form of a rule-based module that, as MULTISENSOR advanced system, does not have access to any external information during generation. MULTISENSOR advanced system achieved a reduction or error of 25.81% for English and 4.17% for Spanish (to compare with the 5% BLEU improvement defined in D1.1).

The evaluation (ii), from abstract predicate-argument graphs, covers the steps described in 5.3.1 to 5.3.5; for our system, we used a classical Penn Treebank split of a development set of 39,279 sentences and a test set of 2,399 sentences, which we processed with the WP2 relation extraction component of MULTISENSOR to serve as an input to the generator. The input for this experiment is at a considerably more abstract level of abstraction than in evaluation (i). The results are contrasted to a state-of-the-art system, shown in Table 8.

System	BLEU score
MULTISENSOR	0.26
Flanigan et al. 2016	0.22

Table 8: Evaluation of the English generation component

Note that even though the input structures in this case are very similar to those used by Flanigan et al. (2016), the results are not directly comparable since the evaluated sentences are not the same; however, if we compare the two systems, the error reduction achieved by the MULTISENSOR generator would reach 5.13%.

7 CONCLUSIONS

The summarization functionalities developed in the scope of the MULTISENSOR project and integrated in the final version of the system includes both the result of research into state-of-the-art extractive methods and a fully functional abstractive system capable of producing multilingual summaries. The development of these methods has faced difficulties not too uncommon to research in this area; scarcity of multilingual corpora and tools required to produce multilingual summaries, difficulty of comparing evaluation results across main extractive and abstractive summarization paradigms, and the challenging nature of the abstractive approach.

What the evaluation results show first and foremost is that simple extractive methods based on shallow linguistic features are hard to beat. Compared to the expectations set in D1.1, in most cases we haven't reached the significant improvements over this baseline that we set for ourselves. One reason for this is that the advantages of incorporating semantic analysis to the summarization pipeline are often offset by the limitation on performance of NLP tools. This is especially true of pipeline architectures like the one adopted in MULTISENSOR, which suffer from error propagation across modules. Attaining deep levels of linguistic and semantic analysis is notoriously difficult for joint approaches to NLP, however.

While user evaluation showed high levels of user satisfaction with extractive summaries, obtaining the same results with abstractive summaries is notoriously difficult. An ideal system should be able to analyse source documents at the linguistic and semantic level, reason about the contents extracted from the documents, and then generate coherent, fluent and grammatical texts. Unfortunately, no reasoning geared towards producing summaries was deployed in MULTISENSOR (e.g. event detection, temporal reasoning, etc.), meaning that the level of abstraction attainable by the system is limited to rephrasing sentences in the source texts. This is one of the reasons why abstractive summarization is restricted to single documents. Nevertheless, we have tried to produce abstractive summaries that are grammatically correct and possess a certain degree of coherence by virtue of the ordering of their sentences.

Adapting the system to new domains, document genres and languages require additional resources relevant to the new intended usage. The type of resources depends largely on the nature of the approach: extractive summarization based on shallow linguistic analysis will require mostly corpora in the target language and belonging to the target domain in order to tune the summarizer. Semantic analysis, on the other hand, demands analysis tools that support the desired languages and capable of effectively working with the target styles of documents. Syntactic parsers, for instance, require language-specific corpora, while EL tools require dictionaries that cover both the language and the concept communicated in the domain. Tools capable of semantic analysis such as semantic role labelling and full semantic parsing may not even be available in most languages. This serves to illustrate the importance of having strategies that can work using only shallow linguistic analysis.

The text planning module is domain-independent and is prepared to work with any inputs data which can be mapped to tree-like representations (i.e. messages). This includes the tree and graph-based output of most state-of-the art deep syntactic and semantic parsers, as well as rich ontological representations were facts are encoded as binary or n-ary relations.

Content selection metrics can be obtained by indexing semantically annotated corpora. In general domains like press articles, the main obstacle towards obtaining corpora is resolving the ambiguity between word senses. In more specialized domains, ambiguity is likely to be a less crucial issue as terms are often used with unequivocal senses. However lexical knowledge bases covering the mapping between linguistic expressions and their associated meanings (e.g. BabelNet) may not be available. Other resources such as word and word sense distributional vectors can be obtained fairly easily provided a large enough collections of documents are available in the desired language and belonging to the desired domain.

Regarding surface realization, the main bottleneck has been the compilation of cross-lingual lexicons, needed for the generation of well-formed sentences. Multilingual summarization also heavily relies on high quality disambiguation in the source language: one single word that is not correctly disambiguated is sufficient to produce an incoherent output in the target language. As a result, the multilingual generation did not reach acceptable quality in order to be evaluated; instead, we focussed on the implementation of a methodology to achieve multilingual summarization, and will show the results on a restricted set of articles during the final demonstrator. For English generation, the comparison between the different systems has not been facilitated by the wide variety of inputs that each of them accepts (and the complexity to adapt a particular system to a given input). However, the technologies that have been developed in the framework of the project compete with state-of-the-art systems while making a step further towards text generation from genuinely abstract structures that contain very little idiosyncratic information.

8 SUMMARY

In this deliverable we have reported the implementation of WP6 modules integrated in the MULTISENSOR final prototype. We have described the last version of the SUMMA-based extractive summarisation pipelines, the corpora used to obtain empirical metrics for estimating relevance of contents during summarisation, and the multilingual abstractive summariser. Due to the complexity of the abstractive summariser, its two main modules (text planning and surface generation) have been described separately.

Following a request for an extension of this deliverable, a new version has been submitted that reports the evaluation of all modules and includes a discussion of the results against the metrics defined in D1.1 and D1.2. The extended version of the deliverable also includes a discussion on how to adapt the modules to other languages, application domains and document genres.

9 REFERENCES

Abeillé, A., Clément, L. and Toussenel, F., (2003). Building a treebank for French. In *Treebanks*, pages 165-187. Springer Netherlands.

Ballesteros, M., Bohnet, B., Mille, S. and Wanner, L. (2015). Data driven sentence generation with non-isomorphic trees. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO. 2015.

Ballesteros, M., C.Dyer, Y. Goldberg and N.Smith. (2016). Greedy Transition-based Dependency Parsing with Stack-LSTMs . In *Computational Linguistics*. MIT Press.

Bohnet, B. and Wanner, L., (2010). Open Soucre Graph Transducer Interpreter and Grammar Development Environment. In *Proceedings of LREC 2010*.

Bohnet, B., L. Wanner, S. Mille and A. Burga. 2010. Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-Level Realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics* (COLING 2010), p.98-106.

Bohnet, B., Björkelund, A., Kuhn, J., Seeker, W. and Zarrieß, S., (2012). Generating non-projective word order in statistical linearization. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928-939. Association for Computational Linguistics.

Bohnet, B., Nivre, J., Boguslavsky, I., Farkas, R., Ginter, G., and Hajic, J. (2013). Joint Morphological and Syntactic Analysis for Richly Inflected Languages, Transactions of the Association for Computational Linguistics.

Brants, S., Dipper, S., Hansen, S., Lezius, W. and Smith, G., (2002). The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, Volume 168.

Cao, Z. et al., (2015). Ranking with Recursive Neural Networks and Its Application to Multi-Document Summarization. In AAAI. pp. 2153–2159.

Cunningham, H., Maynard, D., Bontcheva, et al. (2011). Text Processing with GATE (Version 6).

Diligenti, M., Gori, M., and Maggini, M. (2004). A unified probabilistic framework for web page scoring systems. In *IEEE Transactions on Knowledge and Data Engineering*, Volume 16, pages 4–16.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., and Straňák, P. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1-18. Association for Computational Linguistics.

Hasler, L., Orasan, C., and Mitkov, R., (2003). Building better corpora for summarisation. In *Proceedings of Corpus Linguistics 2003*, Lancaster, UK, March, pp. 309 -- 319

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using linked data. In *Lecture Notes in Computer Science*, pages 98–113.

Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2015). SENSEMBED: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of the American Computational Linguistics* (ACL), pages 95–105.

Flanigan J, Dyer C, Smith NA, Carbonell J. (2016). Generation from Abstract Meaning Representation using Tree Transducers. In Proceedings of the North-American Chapter of the Association for Computational Linguistics: HLT, p.731-739.

Johansson, R., and Nugues, P. (2007). Extended constituent-to-dependency conversion for English, *Proceedings of 16th Nordic Conference of Computational Linguistics*, p. 105-112.

Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. *In Text summarisation branches out: Proceedings of the ACL-04 workshop*, Volume 8.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

Moro, A., Raganato, A., and Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. In *Transactions of the Association for Computational Linguistics*, Volume 2, pages 231-244.

Marcus, M.P., Marcinkiewicz, M.A. and Santorini, B., (1993). Building a large annotated corpus of English: The Penn Treebank. Computational linguistics, 19(2), pp.313-330.

Mel'čuk, I. 1988. *Dependency syntax: Theory and practice*, State University of New York Press.

Mille, S., A.Burga, and L. Wanner. (2013). AnCora-UPF: A Multi-Level Annotation of Spanish. In *Proceedings of the 2nd International Conference on Dependency Linguistics* (DepLing).

Moro, A., Cecconi, F. and Navigli, R., (2014). Multilingual word sense disambiguation and entity linking for everybody. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*, pages 25-28. CEUR-WS.

Navigli, R. and Ponzetto, S.P., (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. In *Artificial Intelligence*, Volume *193*, pp.217-250.

Raganato, A., Delli Bovi, C., & Navigli, R. (2013). Automatic Construction and Evaluation of a Large Semantically Enriched Wikipedia. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (ACL-IJCNLP 2015), Volume 1: Long Papers.

Richardson, M., and Domingos, P., (2002). The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems*, Volume 14, pages 1441–1448. MIT Press.

ShafieiBavani, E. et al., (2016). A Query-Based Summarization Service from Multiple News Sources. In Proceedings of the *2016 IEEE International Conference on Services Computing*. 2016 IEEE International Conference on Services Computing (SCC). pp. 42–49.

Smiley, D., Pugh, E., Parisa, K., and Mitchell, M. (2014). Apache Solr 4 Enterprise Search Server (1st ed.). Packt Publishing.

Steinberger, J., & Ježek, K. (2012). Evaluation measures for text summarization. In *Computing and Informatics*, 28(2), 251-275.

Tsikrika, T., Andreadou, K., Moumtzidou, A., Schinas, E., Papadopoulos, S., Vrochidis, S., and Kompatsiaris, I. (2015). A unified model for socially interconnected multimedia-enriched objects. In *International Conference on Multimedia Modelling,* pages 372-384. Springer International Publishing.

Zhang, K. (1996). A Constrained Edit Distance Between Unordered Labelled Trees. In *Algorithmica*, Volume 15, pages 205–222.

Zhang, X., Cheng, G., and Qu, Y. (2007). Ontology summarisation based on RDF sentence graph. In *Proceedings of the 16th international conference on World Wide Web WWW 07*, volume 2, page 707. ACM.