EUROPEAN LANGUAGE GRID

D4.1

Grid Content: Services, Tools and Components (First Release)

Authors:	Ian Roberts (USFD), Andres Garcia Silva (EXPSYS), Miroslav Janosik (SAIL), Andis Lagzdiņš (Tilde), Nils Feldhus (DFKI), Georg Rehm (DFKI), Dimitris Galanis (ILSP), Dusan Varis (CUNI), Ulrich Germann (UEDIN)

Dissemination Level:	Public
Date:	29-02-2020

About this document

Project	ELG – European Language Grid
Grant agreement no.	825627 – Horizon 2020, ICT 2018-2020 – Innovation Action
Coordinator	Dr. Georg Rehm (DFKI)
Start date, duration	01-01-2019, 36 months
Deliverable number	D4.1
Deliverable title	Services, Tools and Components (First Release)
Type	Other (report plus deployed software)
Number of pages	75
Status and version	Final – Version 1.0
Dissemination level	Public
Date of delivery	Contractual: 29-02-2020 – Actual: 29-02-2020
WP number and title	WP4: Grid Content – Services, Tools, Components
Task number and title	Tasks 4.1: Identify and collect existing services, tools, and components (Section 2) and Tasks 4.2–4.5: Integration of existing ASR, IE, MT and other types of tools, services, and components (Sections 4–7)
Authors	Ian Roberts (USFD), Andres Garcia Silva (EXPSYS), Miroslav Janosik (SAIL), Andis Lagzdiņš (Tilde), Nils Feldhus (DFKI), Georg Rehm (DFKI), Dimitris Galanis (ILSP), Dusan Varis (CUNI), Ulrich Germann (UEDIN)
Reviewers	Gerhard Backfried (SAIL), Florian Kintzel (DFKI)
Consortium	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
	Institute for Language and Speech Processing (ILSP), Greece
	University of Sheffield (USFD), United Kingdom
	Charles University (CUNI), Czech Republic
	Evaluations and Language Resources Distribution Agency (ELDA), France
	Tilde SIA (TILDE), Latvia
	Sail Labs Technology GmbH (SAIL), Austria
	Expert System Iberia SL (EXPSYS), Spain
	University of Edinburgh (UEDIN), United Kingdom
EC project officers	Philippe Gelin, Alexandru Ceausu
For copies of reports and other ELG-related information, please contact:	DFKI GmbH European Language Grid (ELG) Alt-Moabit 91c D-10559 Berlin Germany Dr. Georg Rehm, DFKI GmbH georg.rehm@dfki.de Phone: +49 (0)30 23895-1833 Fax: +49 (0)30 23895-1810 http://european-language-grid.eu © 2020 ELG Consortium

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface – a specification of the rules by which two or more software components can interact
ASR	Automatic Speech Recognition
h2c	HTTP/2 in clear-text (without the use of TLS encryption)
IE	Information Extraction
JSON	JavaScript Object Notation – used as a simple, compact data exchange format for many web APIs, with near-universal support across programming languages
LT	Language Technology
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
SSE	Server-sent events (W3C recommendation https://www.w3.org/TR/eventsource/)

Abstract

This document details the process carried out within the ELG project to identify the tools, services and components available among the ELG project partners, and prioritise which tools to integrate in the ELG platform at which of the three principal release stages (M14, M22, M34). For those tools identified as priorities for integration in the first release, this document also gives details of the approach taken and the work done to integrate each tool. We also include a summary of the API with which tools must comply to be runnable within the ELG platform.

1 Introduction

This document describes the integration framework, tools, services and components and details those tools, services and components that have been developed and made available for the first release (M14/16) of the ELG platform.

Section 2 discusses the work carried out under Task 4.1 to identify existing tools, services and components from among the ELG project partners and prioritise which tools to integrate at which stages of the ELG platform development cycle. The approach we have taken is to initially prioritise those tools that target the languages of the countries where ELG project partners are based, with tools for other EU and related languages to be integrated in the second phase (M22) and other languages in the third phase (M34).

Section 3 gives a brief overview of the ELG platform architecture and APIs, and describes the various options for different ways partners have approached the integration of their tools, to introduce the terminology that is used repeatedly in later sections.

The remainder of the document details the specific tools that have been integrated, grouped as per the WP4 task breakdown – Automatic Speech Recognition (Task 4.2), Information Extraction and Text Analysis (Task 4.3), Machine Translation (Task 4.4) and other types of tools (Task 4.5).

2 Tools, Services and Components: Survey and Prioritisation

As part of the work on Task 4.1 (Identify and collect existing services, tools and components to make them available through the ELG), the ELG partners provided a list of tools that they wanted to integrate in the platform. The information was gathered, the tools grouped by their main function (ASR/IE/MT/other) and standardized so that the different service name variations were unified under common names.

A given tool may provide several different “services” to ELG, for example, a single tool may do tokenisation, sentence splitting, POS tagging and NER, or one MT engine may offer the distinct services of English-to-German and German-to-English translation. We report in Table 1 an overview of the number of services of each type that each ELG partner plans to integrate in the platform. The totals have been broken down further based on the languages covered, divided into four groups: (A) official EU languages; (B) other EU languages without offi-

cial status, plus languages, from candidate countries and free trade partners; (C) languages spoken by immigrants or important trade and political partners; (D) languages that do not fit (A), (B), (C). In addition, Table 2 shows the overall language coverage of each category of services across all partners.

	A	B	C	D	Total
ASR	**13**	**3**	**11**	**1**	**28**
SAIL LABS	**10**	**3**	**11**	**1**	**25**
Speech Recognition	10	3	11	1	25
Tilde	**2**				**2**
Speech Recognition	2				2
UEDIN	**1**				**1**
Speech Recognition	1				1
IE & Text Analysis	**377**	**58**	**157**	**26**	**618**
CUNI	**122**	**35**	**64**		**221**
Dependency Parsing	24	7	13		44
Lemmatisation	24	7	12		43
Morphological analyser	24	7	13		44
Named Entity Recognition	2				2
Part of Speech tagging	24	7	13		44
Tokenization	24	7	13		44
DFKI	**54**	**6**	**13**	**13**	**86**
Categorization	1				1
Date detection	2				2
Language identification	22	6	13	13	54
Morphological analyser	6				6
Named Entity Recognition	4				4
Negation Detection	1				1
Parsing	1				1
Relation Extraction	1				1
Sentence splitting	3				3
Summarization	1				1
Textual Entailment	3				3
Time annotation	2				2
Tokenization	3				3
Misc.	4				4
Expert System	**69**	**4**	**43**	**13**	**129**
Categorization	2				2
Key phrase Extraction	7		5		12
Language identification	22	4	13	13	52
Lemmatisation	7		5		12
Named Entity Recognition	7		5		12
Part of Speech tagging	7		5		12
Sentiment Analysis	3				3
Summarization	7		5		12
Word sense disambiguation	7		5		12
ILSP	**5**				**5**
Information Extraction	2				2
Named Entity Recognition	2				2
Sentiment Analysis	1				1
SAIL LABS	**54**	**10**	**34**		**98**
Keyword extraction	9	3	9		21
Language identification	15	3	8		26

	A	B	C	D	Total
Named Entity Recognition	16	4	9		29
Polarity detection	7		4		11
Sentiment Analysis	7		4		11
USFD	**73**	**3**	**3**		**79**
Categorization	4				4
Entity linking	2				2
Language identification	5				5
Measurement annotation	1				1
Measurement normalisation	1				1
Morphological analyser	1				1
Named Entity Recognition	15	1	1		17
NER Disambiguation	7				7
Noun phrase extraction	1				1
Number annotation	1				1
Number normalisation	1				1
Opinion Mining	2				2
Part of Speech tagging	22	2	2		26
Sentence splitting	3				3
Summarization	2				2
Text extraction	1				1
Tokenization	4				4
Other	**7**	**1**	**1**	**2**	**11**
DFKI	**5**	**1**	**1**	**2**	**9**
Text to Speech	5	1	1	2	9
Tilde	**2**				**2**
Text to Speech	2				2
MT (↓ From – To →)					
CUNI	**5**		**1**	**1**	**7**
A	4		1	1	6
C	1				1
DFKI	**3**				**3**
A	3				3
ILSP	**2**				**2**
A	2				2
Tilde	**34**		**4**		**38**
A	32		4		36
C	2				2
UEDIN	**17**		**1**		**18**
A	16		1		17
C	1				1
Total general	**457**	**68**	**168**	**28**	**721**

Table 1. Overview of languages covered by tools of each category from each partner

	A	B	C	D
ASR				
Speech Recognition	12	3	9	1
IE & Text Analysis				
Categorization	2			
Date detection	2			

	A	B	C	D
Dependency Parsing	24	7	13	
Entity linking	2			
Information Extraction	1			
Key phrase Extraction	7		5	
Keyword extraction	9	3	9	
Language identification	22	6	14	13
Lemmatisation	24	7	13	
Measurement annotation	1			
Measurement normalisation	1			
Morphological analyser	24	7	13	
Named Entity Recognition	16	5	11	
Negation Detection	1			
NER Disambiguation	4			
Noun phrase extraction	1			
Number annotation	1			
Number normalisation	1			
Opinion Mining	1			
Parsing	1			
Part of Speech tagging	24	7	13	
Polarity detection	7		4	
Relation Extraction	1			
Sentence splitting	4			
Sentiment Analysis	8		4	
Summarization	7		5	
Text extraction	1			
Textual Entailment	3			
Time annotation	2			
Tokenization	24	7	13	
Word sense disambiguation	7		5	
MT (↓ From – To →)				
A	30		2	1
C	1			
Other				
Text to Speech	7	1	1	2

Table 2. Language coverage per category of the services to integrate in ELG

2.1 Approach

The long list of tools, services and languages supported, and the different development stages of the ELG platform (i.e., the three different releases) make it necessary to prioritize the integration of the tools according to support required for the ELG pilot projects where external SMEs will use the ELG to develop innovative projects. Thus, integration of the tools has been divided according to the three platform releases on M14, M22 and M34 and synchronized with the call for pilots (M15 and M21).

For the first integration stage on M14 partners are required to integrate their tools covering the languages of countries where the ELG partners are located, that is German, English, Spanish, French, Greek, Latvian, and

Czech. Given the large number of services in IE and Text Analysis the integration is focused on the most frequent services offered in this category. Next, in M22 partners will integrate the services covering all the languages in categories A and B. Finally, in M34 the goal is to integrate the rest of services supporting languages in categories C and D. In the following, for each tool type we present the tools to integrate in each of the aforementioned stages.

2.2 ASR: Automatic Speech Recognition

Automatic Speech Recognition (ASR) are services that take audio as input and produce text (e.g., a transcription) as output, possibly with metadata associated with each segment. The integration plan of the ASR tools according to the three releases is presented in Table 3, Table 4 and Table 5.

		M14		
Provider	**Tool**	**Service**	**Language**	**Category**
SAIL LABS	SAIL ASR	ASR	French	A
SAIL LABS	SAIL ASR	ASR	English	A
SAIL LABS	SAIL ASR	ASR	German	A
SAIL LABS	SAIL ASR	ASR	Spanish	A
SAIL LABS	SAIL ASR	ASR	Greek	A
Tilde	Tilde ASR	ASR	Latvian	A
UEDIN	Cloud ASR1	ASR	Czech	A

Table 3. ASR tools and services to integrate in the first release

		M22		
Provider	**Tool**	**Service**	**Language**	**Category**
SAIL LABS	SAIL ASR	ASR	Norwegian (Bokmal)	B
SAIL LABS	SAIL ASR	ASR	Romanian	A
SAIL LABS	SAIL ASR	ASR	Albanian	B
SAIL LABS	SAIL ASR	ASR	Italian	A
SAIL LABS	SAIL ASR	ASR	Turkish	B
SAIL LABS	SAIL ASR	ASR	Polish	A
SAIL LABS	SAIL ASR	ASR	Dutch	A
Tilde	Tilde ASR	ASR	Lithuanian	A

Table 4. ASR tools and services to integrate in the second release

		M34		
Provider	**Tool**	**Service**	**Language**	**Category**
SAIL LABS	SAIL ASR	ASR	Arabic	C
SAIL LABS	SAIL ASR	ASR	Egyptian Arabic	C
SAIL LABS	SAIL ASR	ASR	Levantine Arabic	C
SAIL LABS	SAIL ASR	ASR	Indonesian (Bahasa)	C
SAIL LABS	SAIL ASR	ASR	Urdu	C
SAIL LABS	SAIL ASR	ASR	Malay	C
SAIL LABS	SAIL ASR	ASR	Persian (Farsi)	C
SAIL LABS	SAIL ASR	ASR	Hebrew	C
SAIL LABS	SAIL gender detection	Gender-detection	Lang. Ind.	E
SAIL LABS	SAIL age detection	Age-bracket-detection	Lang. Ind.	E

[1] The models developed by UEDIN will be incorporated into the SAIL LABS ASR component.

		M34		
Provider	**Tool**	**Service**	**Language**	**Category**
SAIL LABS	SAIL ASR	ASR	Russian	C
SAIL LABS	SAIL ASR	ASR	Pashto	C
SAIL LABS	SAIL ASR	ASR	Mexican Spanish	D
SAIL LABS	SAIL ASR	ASR	Chinese (Mandarin)	C
SAIL LABS	SAIL ASR	ASR	Portuguese	A

Table 5. ASR tools and services to integrate in the third release

2.3 IE: Information Extraction and Text Analysis

Information Extraction (IE) services are services that take text and annotate it with metadata on specific segments. This class can cover a wide variety of services from basic NER through to complex sentiment analysis and domain-specific tools. As mentioned before, the list of services in this category is long and hence a first filter on the services to integrate in the first release is based on the popularity. In Table 6 we present the list of services ranked according to number of tools that support them. Based on this ranking the top 15 most popular services covering the languages of the countries in which the ELG partners are located will be integrated in the first release. The second and third integration releases include the rest of services for all the languages in categories A and B for the former, and the rest of languages for the latter.

Service	**Freq**	**Rank**	**Service**	**Freq**	**Rank**
Named Entity Recognition	22	1	Cross-lingual Candidate Search	1	19
Part of Speech tagging	8	2	Date detection	1	20
Information Extraction	8	3	GATE app on the cloud	1	21
Categorization	6	4	Key phrase Extraction	1	22
Tokenization	5	5	Keyword extraction	1	23
NER Disambiguation	4	6	Measurement annotation	1	24
Sentiment Analysis	4	7	Measurement normalisation	1	25
Summarization	3	8	Negation Detection	1	26
Language identification	3	9	Bots	1	27
Sentence splitting	3	10	Number normalisation	1	28
Morphological analyser	3	11	Parsing	1	29
Dependency Parsing	2	12	Polarity detection	1	30
Number annotation	2	13	Relation Extraction	1	31
Opinion Mining	2	14	Text analysis	1	32
Lemmatisation	2	15	Textual Entailment	1	33
Text extraction	2	16	Time annotation	1	34
Entity linking	2	17	Word sense disambiguation	1	35
Noun phrase extraction	1	18			

Table 6. IE and text analysis tools ranked according to the frequency of tool support

An overview of the services that will be integrated in each release is presented in Table 7 and Table 8 (release 1), Table 9 and Table 10 (release 2), and Table 11 (release 3). In each table the cell number is the number of tools that will be integrated providing support for the service type. We provide the full list of tools and their corresponding services to integrate in each release in Tables 30, 31 and 32 that are part of the appendix.

	Czech	English	French	German	Greek	Latvian	Spanish	Total
Categorization		6					1	7
Dependency Parsing	1	1	1	1	1	1	1	7
Information Extraction					2			2
Language identification	3	4	4	4	3	2	4	24
Lemmatisation	1	2	2	2	1	1	2	11
Morphological analyser	1	3	2	2	1	1	2	12
Named Entity Recognition	2	13	4	8	2		2	31
NER Disambiguation		4	1	1			1	7
Number annotation		1						1
Opinion Mining		2						2
Part of Speech tagging	2	5	3	3	2	2	3	20
Sentence splitting		2		2				4
Sentiment Analysis		2	2	1	1		1	7
Summarization		3	1	1			2	7
Tokenization	1	4	1	3	1	1	1	12
Total general	**11**	**52**	**21**	**28**	**14**	**8**	**20**	**154**

Table 7. Overview of IE and Text Analysis services to integrate in the first release

	Czech	English	French	German	Greek	Latvian	Spanish	Total
CUNI	**6**	**6**	**5**	**5**	**5**	**5**	**5**	**37**
Dependency Parsing	1	1	1	1	1	1	1	7
Lemmatisation	1	1	1	1	1	1	1	7
Morphological analyser	1	1	1	1	1	1	1	7
Named Entity Recognition	1	1						2
Part of Speech tagging	1	1	1	1	1	1	1	7
Tokenization	1	1	1	1	1	1	1	7
DFKI	**1**	**8**	**2**	**6**	**1**	**1**	**2**	**21**
Categorization		1						1
Language identification	1	1	1	1	1	1	1	7
Morphological analyser		1	1	1			1	4
Named Entity Recognition		2		2				4
Sentence splitting		1		1				2
Summarization		1						1
Tokenization		1		1				2
Expert System	**1**	**7**	**6**	**5**	**1**	**1**	**6**	**27**
Categorization		1					1	2
Language identification	1	1	1	1	1	1	1	7
Lemmatisation		1	1	1			1	4
Named Entity Recognition		1	1	1			1	4
Part of Speech tagging		1	1	1			1	4
Sentiment Analysis		1	1					2
Summarization		1	1	1			1	4
ILSP		**1**			**4**			**5**
Information Extraction					2			2
Named Entity Recognition		1			1			2
Sentiment Analysis					1			1
SAIL LABS	**2**	**3**	**3**	**3**	**2**		**3**	**16**
Language identification	1	1	1	1	1		1	6
Named Entity Recognition	1	1	1	1	1		1	6

	Czech	English	French	German	Greek	Latvian	Spanish	Total
Sentiment Analysis		1	1	1			1	4
USFD	**1**	**27**	**5**	**9**	**1**	**1**	**4**	**48**
Categorization		4						4
Language identification		1	1	1			1	4
Morphological analyser		1						1
Named Entity Recognition		7	2	4				13
NER Disambiguation		4	1	1			1	7
Number annotation		1						1
Opinion Mining		2						2
Part of Speech tagging	1	3	1	1	1	1	1	9
Sentence splitting		1		1				2
Summarization		1					1	2
Tokenization		2		1				3
Total	**11**	**52**	**21**	**28**	**14**	**8**	**20**	**154**

Table 8. IE and Text Analysis services provided by ELG partners to be integrated in the first release

	Bulgarian	Croatian	Danish	Dutch	English	Estonian	Finnish	French	German	Greek	Hungarian	Irish	Italian	Lithuanian	Maltese	Polish	Portuguese	Romanian	Slovak	Slovenian	Spanish	Swedish	Total
Date detection					1				1														2
Dependency Parsing	1	1	1	1		1	1				1	1	1	1	1	1	1	1	1	1		1	17
Entity linking					1				1														2
Key phrase Extraction				1	1			1	1				1				1				1		7
Keyword extraction				1	1			1	1	1			1			1		1			1		9
Language identification	3	2	2	4		2	2				3		3	2		3	3	3	3	2		3	40
Lemmatisation	1	1	1	2		1	1				1	1	2	1	1	1	2	1	1	1		1	20
Measurement annotation					1																		1
Measurement normalisation					1																		1
Morphological analyser	1	1	1	2		1	1				1	1	2	1	1	1	1	1	1	1		1	19
Named Entity Recognition	1	1		3							1		2			1	2	2	1			1	15
Negation Detection									1														1
Noun phrase extraction					1																		1
Number normalisation					1																		1
Parsing									1														1
Part of Speech tagging	2	2	2	4		2	2				1	1	2	1	1	2	3	2	2	2		2	33
Polarity detection					1			1	1				1			1	1				1		7
Relation Extraction									1														1
Sentence splitting				1									1										2
Sentiment Analysis													2			1	1						4
Summarization				1									1				1						3
Text extraction					1																		1
Textual Entailment					1				1				1										3
Time annotation					1				1														2

	Bulgarian	Croatian	Danish	Dutch	English	Estonian	Finnish	French	German	Greek	Hungarian	Irish	Italian	Lithuanian	Maltese	Polish	Portuguese	Romanian	Slovak	Slovenian	Spanish	Swedish	Total
Tokenization	1	1	1	2		1	1				1	1	2	1	1	1	1	1	1	1		1	19
Word sense disambiguation				1	1			1	1				1				1				1		7
Total	**10**	**9**	**8**	**23**	**13**	**8**	**8**	**4**	**11**	**1**	**9**	**5**	**23**	**7**	**5**	**13**	**18**	**12**	**10**	**8**	**4**	**10**	**219**

Table 9. Overview of IE and Text Analysis services to integrate in the second release (category A)

	Albanian	Basque	Catalan	Galician	Norwe-gian	Serbian	Turkish	Ukrainian	Welsh	Total
Dependency Parsing		1	1	1	1	1	1	1		7
Keyword extraction	1				1		1			3
Language identification	3		1		3		3	2	1	13
Lemmatisation		1	1	1	1	1	1	1		7
Morphological analyser		1	1	1	1	1	1	1		7
Named Entity Recognition	1		1		1		1		1	5
Part of Speech tagging		2	2	1	1	1	1	1		9
Tokenization		1	1	1	1	1	1	1		7
Total	**5**	**6**	**8**	**5**	**10**	**5**	**10**	**7**	**2**	**58**

Table 10. Overview of IE and Text Analysis services to integrate in the second release (category B)

	Afrikaans	Arabic	Bengali	Chinese	Gujarati	Hebrew	Hindi/Urdu	Indonesian	Japanese	Kannada	Korean	Language independent	Latin	Macedonian	Malay	Marahati	Nepali	Panjabi	Pashto	Persian	Russian	Somali	Swahili	Tagalog	Tamil	Telugu	Thai	Urdu	Vietnamese	Total
Dependency Parsing	1	1		1		1	1	1	1		1	1	1							1	1				1				1	14
Entity linking												1																		1
GATE app on the cloud												1																		1
Key phrase Extraction		1		1					1		1										1									5
Keyword extraction		1		1		1	1	1							1				1	1	1									9
Language identification	2	3	2	2	2	3	3	3	2	2	2	1		2	3	2	2	2	1	3	3	2	2	2	2	2	2	2	2	61
Lemmatisation	1	2		2		1	1	1	2		1		1							1	2				1				1	17
Morphological analyser	1	1		1		1	1	1	1		1		1							1	1				1				1	13
Named Entity Recognition		2		2		1	1	1	1		1				1				1	1	3									15
Part of Speech tagging	1	2		2		1	1	2	2		2	1	1							1	3				1				1	21
Polarity detection		1						1							1						1									4
Sentence splitting												1																		1
Sentiment Analysis		1						1							1						1									4
Summarization		1		1					1		1	1									1									6
Text extraction												1																		1
Tokenization	1	1		1		1	1	1	1		1	1	1							1	1				1				1	14
Word sense disambiguation		1		1					1		1										1									5
Total	**7**	**18**	**2**	**15**	**2**	**10**	**10**	**13**	**13**	**2**	**12**	**9**	**5**	**2**	**7**	**2**	**2**	**2**	**3**	**10**	**20**	**2**	**2**	**2**	**7**	**2**	**2**	**2**	**7**	**192**

Table 11. Overview of IE and Text Analysis services to integrate in the third release (category C and D)

2.4 MT: Machine Translation

Machine Translation (MT) services take text in one language and translate it into text in another language, possibly with additional metadata associated with each segment (sentence, phrase, etc.). These services mostly follow the general approach to integration in ELG based on the language coverage. However, some services have been delayed to a different stage for example some MT services that are being developed in the context of other projects (e.g., Bergamot). The list of MT tools and services to integrate in each stage is presented in Table 12, Table 13 and Table 14.

		M14			
Provider	**Service**	**From**	**Category**	**To**	**Category**
CUNI	Machine Translation	Czech	A	English	A
CUNI	Machine Translation	English	A	Czech	A
CUNI	Machine Translation	English	A	French	A
CUNI	Machine Translation	French	A	English	A
DFKI	TQ-AutoTest	German	A	English	A
DFKI	Quality Estimation	German	A	English	A
DFKI	Quality Estimation	English	A	German	A
DFKI	Quality Estimation	Spanish	A	English	A
DFKI	Quality Estimation	English	A	Spanish	A
DFKI	Quality Estimation	French	A	English	A
DFKI	Quality Estimation	English	A	French	A
DFKI	Machine Translation	German	A	English	A
Tilde	Machine Translation	English	A	Bulgarian	A
Tilde	Machine Translation	English	A	Latvian	A
Tilde	Machine Translation	English	A	Polish	A
Tilde	Machine Translation	Latvian	A	English	A
Tilde	Machine Translation	Polish	A	English	A
UEDIN	Machine Translation	Czech	A	English	A
UEDIN	Machine Translation	English	A	German	A
UEDIN	Machine Translation	German	A	English	A

Table 12. MT tools and services to integrate in the first release

		M22			
Provider	**Service**	**From**	**Category**	**To**	**Category**
Tilde	Machine Translation	Bulgarian	A	English	A
Tilde	Machine Translation	Danish	A	English	A
Tilde	Machine Translation	English	A	Danish	A
Tilde	Machine Translation	English	A	Estonian	A
Tilde	Machine Translation	English	A	Finnish	A
Tilde	Machine Translation	English	A	Lithuanian	A
Tilde	Machine Translation	English	A	Swedish	A
Tilde	Machine Translation	Estonian	A	English	A
Tilde	Machine Translation	Finnish	A	English	A
Tilde	Machine Translation	Lithuanian	A	English	A
Tilde	Machine Translation	Swedish	A	English	A
UEDIN	Machine Translation	English	A	Estonian	A
UEDIN	Machine Translation	English	A	Latvian	A
UEDIN	Machine Translation	English	A	Portuguese	A
UEDIN	Machine Translation	English	A	Romanian	A
UEDIN	Machine Translation	English	A	Spanish	A
UEDIN	Machine Translation	Estonian	A	English	A
UEDIN	Machine Translation	Latvian	A	English	A

		M22			
Provider	**Service**	**From**	**Category**	**To**	**Category**
UEDIN	Machine Translation	Portuguese	A	English	A
UEDIN	Machine Translation	Romanian	A	English	A
UEDIN	Machine Translation	Spanish	A	English	A

Table 13. MT tools and services to integrate in the second release

		M34			
Provider	**Service**	**From**	**Category**	**To**	**Category**
CUNI	Machine Translation	English	A	Hindi	D
Tilde	Machine Translation	English	A	Arabic	C
Tilde	Machine Translation	Russian	C	English	A
Tilde	Machine Translation	English	A	Russian	C
UEDIN	Machine Translation	English	A	Polish	A
UEDIN	Machine Translation	English	A	Russian	C
UEDIN	Machine Translation	Russian	C	English	A
UEDIN/Bergamot	Machine Translation	Polish	A	English	A
UEDIN/Gourmet	Machine Translation	English	A	Bulgarian	A

Table 14. MT tools and services to integrate in the third release

2.5 Other Services

Here, we list the tools provided by the ELG partners that do not fit in any of the previous categories (ASR, MT, IE and Text Analysis). Since these tools are not common and less relevant for the pilot projects, their integration has been postponed mostly for the second integration stage. The list of the tools in this category according to each integration stage is presented in Table 15 and Table 16, and currently comprise only TTS (text-to-speech) tools. Other tools may be added to this list as they emerge later in the project.

			M14	
Provider	**Tool**	**Service**	**Language**	**Category**
Tilde	Tilde TTS	Text to Speech	Lithuanian	A
Tilde	Tilde TTS	Text to Speech	Latvian	A

Table 15. Other tools and services to integrate in the first release

		M22		
Provider	**Tool**	**Service**	**Language**	**Category**
DFKI	Mary	Text to Speech	Luxembourgish	D
DFKI	Mary	Text to Speech	Telugu	D
DFKI	Mary	Text to Speech	Turkish	B
DFKI	Mary	Text to Speech	Russian	C
DFKI	Mary	Text to Speech	English	A
DFKI	Mary	Text to Speech	Italian	A
DFKI	Mary	Text to Speech	French	A
DFKI	Mary	Text to Speech	Swedish	A
DFKI	Mary	Text to Speech	German	A

Table 16. Other tools and services to integrate in the second release

3 ELG Tool Integration Framework

The approach taken in ELG to facilitate the integration of a large number of disparate tools rests on a number of basic principles:

- Define common APIs for each type of tool, designed to be powerful enough to support the necessary use cases but lightweight and flexible enough to allow tools to expose their own specific parameters where this makes sense.
- Use containerization to isolate tools from one another and allow each tool to manage its own software dependencies. The ELG platform uses the well-established Kubernetes[2] software to manage the execution of containers and knative[3] to handle auto-scaling of the containers up and down, on demand.

3.1 API and Protocol Specification

For this release of the ELG platform, Task 2.5 has defined an HTTP-based protocol for LT tools to implement, with a standardized set of JSON request and response messages. Each LT tool (or group of related tools) runs as a single pod in Kubernetes terminology – a group of one or more Docker images that are run together as a unit by Kubernetes and can connect to one another via the localhost network interface. The pod must offer one or more HTTP (1.1 or h2c) endpoints that can accept requests in the relevant ELG message format, perform the required processing, and return a properly formatted response, and the ELG platform can react to changing demand for particular services by scaling up and down, creating and destroying replicas based on the same pod definition. The current API specification is explained below, but the most up to date version will always be available on https://gitlab.com/european-language-grid.

Request format

There are three types of request messages available for different types of services. **Services that process *text*** should accept a POST request with "Content-Type: application/json" in the following format:

```
{
  "type":"text",
  "params":{...},   /* optional */
  "content":"The content, as a string inline",
  // mimeType optional - this is the default if omitted
  "mimeType":"text/plain",
  "features":{ /* arbitrary JSON metadata about this content, optional */ },
  "annotations":{ /* optional */
    "<annotation type>":[
      {
        "start":number,
        "end":number,
        "features":{ /* arbitrary JSON */ }
      }
    ]
  }
}
```

[2] https://kubernetes.io
[3] https://knative.dev

Most parts of the request are optional, the only items that are guaranteed to be included in all requests are "type" and "content". The "features" and "annotations" sections may be useful for services that can build on partially-annotated output from other services, for example a named entity recognizer that expects the input text to have been tokenized and split into sentences by a previous component. The "start" and "end" for each annotation are numbers that refer to the position of the annotation counting in characters[4] from the start of the content – zero is before the first character, 1 is between the first and second, etc.

As an alternative to taking plain text with separate annotations, some services may prefer instead (or in addition) to accept a "structured text" request which has been segmented explicitly. Again, this is a POST request with "Content-Type: application/json", with the format as follows:

```
{
  "type":"structuredText",
  "params":{...},   /* optional */
  "texts":[
    {
      "content":"The content, as a string inline", // either
      "texts":[/* same structure, recursive */],   // or
      // mimeType optional - this is the default if omitted
      "mimeType":"text/plain",
      "features":{ /* arbitrary JSON metadata about this content, optional */ },
      "annotations":{ /* optional */
        "<annotation type>":[
          {
            "start":number,
            "end":number,
            "features":{ /* arbitrary JSON */ }
          }
        ]
      }
    }
  ]
}
```

This is a recursive structure – each of the objects in the top-level "texts" array may contain either a single string of "content" or another array of "texts" in the same format. Again, most parts of this structure are optional. If "annotations" are supplied along with "content" then the start and end refer to character offsets as before, but if they are supplied along with "texts" then they refer to offsets into the texts array – if for example the "texts" array represents a list of sentences, then annotations could refer to the second and third sentences using "start":1 and "end":3.

Services that accept *audio* input are slightly different, they will receive a POST request of "Content-Type: multipart/form-data", where the first MIME part named "request" contains JSON in the following format:

```
{
  "type":"audio",
  "params":{...}, // optional
  "format":"string", // LINEAR16 for WAV or MP3 for MP3
  "sampleRate":number,
  "features":{ /* arbitrary JSON metadata about this content, optional */ },
  "annotations":{ /* optional */
    "<annotation type>":[
      {
        "start":number,
```

[4] Following RFC4627, the character encoding used within JSON is UTF-8.

```
        "end":number,
        "features":{ /* arbitrary JSON */ }
      }
    ]
  }
}
```

And the second part named “content” will contain the actual audio data as “audio/x-wav” or “audio/mpeg”. In the future, an alternative approach of passing in references (URIs) to stored audio objects may be required.

Response format

Whatever the input format, the endpoint should respond with JSON. There are a number of defined response formats suitable for different types of services.

Services that perform annotation of the input text or audio can return an “annotations” response:

```
{
  "response":{
    "type":"annotations",
    "warnings":[...], /* optional */
    "features":{...}, /* optional */
    "annotations":{
      "<annotation type>":[
        {
          "start":number,
          "end":number,
          "features":{ /* arbitrary JSON */ }
        }
      ]
    }
  }
}
```

The interpretation of annotations is as for the text request type above – for services that process text the start and end are character offsets, for services that process audio they are floating point time offsets in seconds. The “features” section is a placeholder for metadata pertaining to the response as a whole, the “annotations” are for metadata pertaining to specific sections.

Services that *classify* the input as a whole may use the following response format:

```
{
  "response":{
    "type":"classification",
    "warnings":[...], /* optional */
    "classes":[
      {
        "class":"string",
        "score":number /* optional */
      }
    ]
  }
}
```

We allow for zero or more classifications, each with an optional score. Services should return multiple classes in whatever order they feel is most useful (e.g., “most probable class” first), this order need not correspond to a monotonic ordering by score.

Services that generate *new text* as output (e.g., translation services) should use the following response format, which is somewhat more complex than the others:

```
{
  "response":{
    "type":"texts",
    "warnings":[...], /* optional */
    "texts":[
      {
        "role":"string", /* optional */
        "content":"string of translated/transcribed text", // either
        "texts":[/* same structure, recursive */],         // or
        "score":number, /* optional */
        "features":{ /* arbitrary JSON, optional */ },
        "annotations":{ /* optional */
          "<annotation type>":[
            {
              "start":number,
              "end":number,
              "sourceStart":number, // optional
              "sourceEnd":number,   // optional
              "features":{ /* arbitrary JSON */ }
            }
          ]
        }
      }
    ]
  }
}
```

As with the structured text request, this structure is recursive – each entry in the top-level "texts" array can have either a single string of "content" or another array of "texts". The texts response type can be used in two different ways, either

- the top-level list of texts is interpreted as a set of *alternatives* for the whole result – in this case we would expect the "content" property to be populated but not the "texts" one, and a "role" value of "alternative"
    - in this case services should return the alternatives in whatever order they feel is most useful, typically descending order of likelihood (though as for classification results we don't assume scores are mutually comparable and the order of alternatives in the array need not correspond to a monotonic ordering by score).
- the top-level list of texts is interpreted as a set of *segments* of the result, where each segment can have N-best alternatives (e.g., a list of sentences, with N possible alternative translations or transcriptions for each sentence). In this case we would expect "texts" to be populated but not "content", and a "role" value indicating the nature of the segmentation such as "sentence", "paragraph", "turn" (for speaker detection), etc.
    - in this case the order of the texts should correspond to the order of the segments in the result.

This pattern may be continued recursively if appropriate. Each alternative or segment may optionally contain annotations over sub-ranges of characters within that content or segments within the next level "texts". Annotations may optionally be anchored also to locations within the *source* data (which may be text for machine translation or audio for ASR) via the "sourceStart" and "sourceEnd" properties.

Finally, **services that return *audio*** may use the following format:

```
{
  "response":{
    "type":"audio",
    "warnings":[...], /* optional */
    // either embedded content or contentLocation
    "content":"base64 encoded audio for shorter snippets",
    "format":"string",
    "features":{/* arbitrary JSON, optional */},
    "annotations":{
      "<annotation type>":[
        {
          "start":number,
          "end":number,
          "sourceStart":number, // optional
          "sourceEnd":number,   // optional
          "features":{ /* arbitrary JSON */ }
        }
      ]
    }
  }
}
```

As with the "texts" response, this response may contain annotations that are anchored to points in both the output audio ("start" and "end") and the source data ("sourceStart" and "sourceEnd").

Error reporting

If processing fails for any reason the service should respond with the following structure:

```
{
  "failure":{
    "errors":[
      {
        "code":"elg.service.internalError",
        "text":"Internal error during processing: {0}",
        "params":["execution timeout"]
      }
    ]
  }
}
```

The object inside the "errors" array is referred to as a "status message", and the structure is designed to be amenable to internationalisation. The "code" identifies the message for lookup in a list for each language, the "text" gives fallback text to use if no translation can be found. Message strings can include numbered place-holders which are filled in using the values in the "params" array. The ELG platform will define a standard set of error codes whose messages will be fully translated, and a REST service to render a status message into any supported language, but individual service providers can use their own codes instead (though they will then be responsible for supplying translations).

The same status message structure can be used in the "warnings" section of any of the successful response types above, to communicate problems that occurred but did not prevent processing from completing successfully.

Progress reporting

Some LT services can take a long time to process each request, and in these cases it may be useful to be able to send intermediate progress reports back to the caller. This serves both to reassure the caller that processing

has not silently failed, and also to ensure the HTTP connection is kept alive. The mechanism for this in ELG leverages the W3C standard Server-Sent Events (SSE) protocol format[5] – *if* the client sends an "Accept" header that announces that it is able to understand the `text/event-stream` response type, then the service may choose to immediately return a 200 "OK" response with "Content-Type: text/event-stream" and hold the connection open (using chunked transfer encoding in HTTP/1.1 or simply not sending a Content-Length in HTTP2). It may then dispatch zero or more SSE "events" with JSON data in the following structure:

```
{
  "progress":{
    "percent"://number between 1.0 and 100.0,
    "message":{
      // optional status message, with code, text and params as above
    }
  }
}
```

followed by *exactly one* successful or failed response in the usual format.

3.2 Containerisation

LT tools and services may be implemented in a wide variety of programming languages and frameworks, and different services may use different and incompatible versions of the same libraries. To isolate tools from one another and avoid dependency conflicts, the platform has adopted Docker as the containerization technology and Kubernetes and knative as the orchestration framework. To run tools within the ELG Kubernetes cluster, LT service implementors must provide their tools as one or more Docker images, working together as a single Kubernetes pod. There are three principal approaches that have been taken so far by ELG project partners when integrating their services, which are summarized in Figure 1:

[5] https://www.w3.org/TR/eventsource/ – ELG usage ignores the event type, ID and retry fields and simply uses the format as a standard way to divide the response into a stream of discrete chunks with the semantics that chunks should be delivered back to the caller in as close as possible to real time when they are generated, rather than being buffered and returned in a single group at the end of processing.

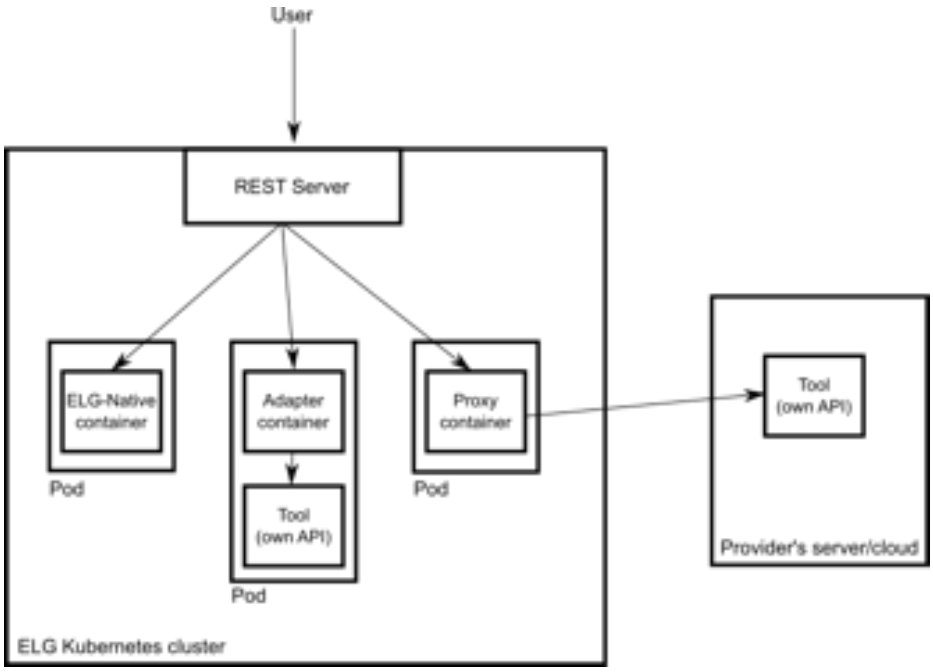

Figure 1. The three basic integration options – ELG-native, adapter, or proxy

1. "ELG-native" approach – provide a single Docker image that includes the provider's own software with a native implementation of the ELG API specification. This approach is suited to tools that provide a software library API (or are built within a standard framework that provides such an API) but do not have their own HTTP interface by default.
2. "Adapter pattern" approach – for tools that already present their own non-ELG HTTP API interface, it is often simpler to use an adapter pattern, with one image containing the tool itself exposed via its native HTTP API, and one or more separate "adapter" images that translate ELG requests into native requests and the native responses back to the ELG format. Kubernetes allows a pod to be made up of multiple containers, and the containers within a pod share a common network and can communicate with one another as "localhost".
3. "Proxy" approach – this is very similar to the adapter pattern above, except that the main service is hosted elsewhere on the internet, outside the ELG cluster, and the pod running in Kubernetes contains only the adapter image, receiving ELG requests and translating them to native calls out to the remote service.

Individual Docker images may be hosted in any Docker registry that is accessible from the ELG cluster, both public (anyone can pull the image without authentication) and private (authentication required) registries are supported. In the case of private repositories, it is necessary to supply the registry credentials using the Kubernetes "secrets" mechanism. For this, we have set up a dedicated Kubernetes namespace for each LT service provider with access restricted to the cluster administrators plus the nominated staff of that particular provider. The provider must then manually provision secrets in their namespace containing the credentials for their registry, and then reference the relevant secret from each pod definition. Pods running within the provider's namespace are likewise only accessible to that provider's staff – other cluster users cannot read their logs or inspect the container filesystem, secrets, etc.

Each pod can implement any number of logical LT services, each exposed on a different URL path, for example one pod may be able to provide translation services from German to English at /translate/de-en and English to German at /translate/en-de – the metadata records in the ELG catalogue provide the link between each separate logical service and the specific back end URL endpoint.

It is important to note that the APIs described in Section 3.1 are not directly exposed to the public. There is a separate component within the ELG platform, referred to within the project as the RESTServer, which is the public entry point into all ELG services. The RESTServer offers public-facing API endpoints for each logical LT service, deals with cross-cutting concerns such as authentication, access control and rate limiting, and then dispatches processing requests to the individual LT service containers following the Section 3.1 API. At present the public-facing APIs mirror very closely the internal ones described in this document, but this architecture choice allows the ELG platform to later implement other APIs (including other interaction styles such as batch processing) and make them available in a consistent way for all compatible LT services without the individual services needing to implement every different API themselves.

4 ASR Tools, Services and Components (Task 4.2)

4.1 ASR Tools Integrated by SAIL

SAIL LABS brings a set of ASR components to the ELG platform which are based on SAIL's Media Mining Indexer (MMI, https://www.sail-labs.com/media-mining-system/#MMIndexer).

The MMI is a component combining a set of audio- and text-based processing technologies. These are implemented as sub-components which are connected internally in a pipelined manner (these connections can likewise be configured to support different setups). MMIs can operate as instances and be coupled to form groups of processing units. The actual functionality of an MMI is controlled by a set of license-flags and a set of parameter files and models. All processing steps can be activated or de-activated on demand; all models can be updated in a transparent manner even during processing. Several modes of operation are offered, depending on parameter settings (e.g., a real-time transcription mode for ASR via self-aware gauging of processing). For R1, only a single model per container and no dynamic updating of models is supported. For the dockerized version of this component, dynamic updating and refreshing may easily be implemented by the creation of new versions of the container.

SAIL provides the MMI for its ASR services within the ELG. The components include a set of processing steps, such as segmentation and classification of segments and the actual ASR component, which can be configured as a single or multiplass recognizer. Results may be obtained as 1-best, n-best or lattice (currently only 1-best is delivered within ELG).

ASR is supported for six languages: Czech, English, French, German, Greek, and Spanish.

All models have been developed by SAIL LABS with the exception of Czech, which has been developed by the University of Edinburgh. As the same underlying technology (KALDI) as well as the same recipe for model-building are used, it was possible to integrate the ASR models for Czech in a seamless manner into the SAIL LABS MMI.

ASR can be configured based on a set of parameters and optimized for speed or accuracy, the components models can be set to a standard model (for the transcription of broadcast-news) or to specific, domain-dependent models (such as a model for the transcription of crisis/disaster-related content). The current implementation uses the base settings. In the future it is planned to provide a series of domain-dependent model (for improved accuracy) as well as to allow settings for real-time (or faster if run from file) transcription.

License: SAIL ASR is a commercial component that requires a commercial license to use it. While the ELG platform is under construction and there is no billing and payment option in place, the services are available free for testing and development purposes, request limits can be introduced at any time.

Integration approach

For each of the above-mentioned components, the MMI and the respective model have been packaged into a single container. Communication takes place via the ELG-API, parameters being passed according to the specifications of Task 2.5 (see Section 3.1 above). Output is produced according to the same specifications.

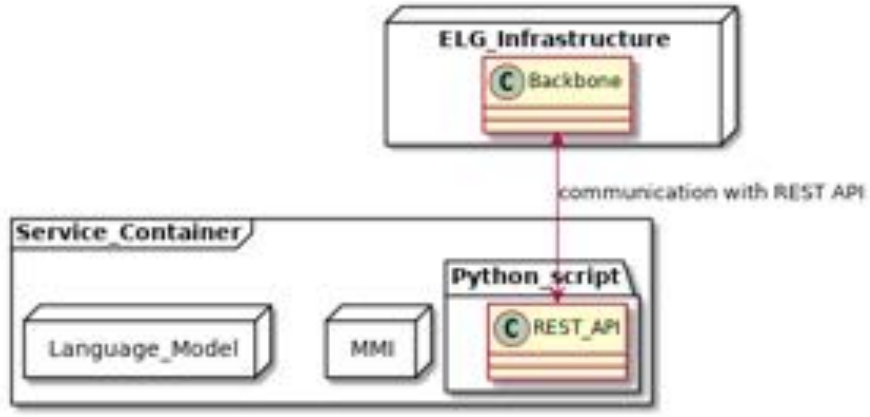

Figure 2. Integration of SAIL MMI

The audio file is passed in, decoding is triggered and upon completion a sequence of words, time-stamps (in cs) and confidence-scores is delivered.

Deployed components

Table 17 shows the ASR software components that are deployed in the ELG platform. The code is available in the ELG project repository in gitlab and the images in the container registry in the same platform. The code and images are private projects as they are commercial software.
Container registry: sail/<image name> (full path: registry.gitlab.com/european-language-grid/sail/<image name>)
Code repository: sail/<image name> (full path: https://gitlab.com/european-language-grid/sail/<image name>)

Type	Tool Type	Image name	Service	Languages/variants supported	Licence
tool	ASR	sail-asr-en	Automatic Speech Recognition	English	SAIL LABS
tool	ASR	sail-asr-fr	Automatic Speech Recognition	French	SAIL LABS
tool	ASR	sail-asr-de	Automatic Speech Recognition	German	SAIL LABS

tool	ASR	sail-asr-el	Automatic Speech Recognition	Greek	SAIL LABS
tool	ASR	sail-asr-es	Automatic Speech Recognition	Spanish	SAIL LABS
tool	ASR	sail-asr-cs	Automatic Speech Recognition	Czech	TBD

Table 17. SAIL ASR tools integrated in the first release

4.2 ASR Tools Integrated by TILDE

Tilde ASR is part of Tilde's Speech platform which provides speech solutions[6] for the Baltic languages.

Tilde Speech platform is a scalable software platform for automatic speech recognition and speech synthesis that can be easily integrated using Web APIs. It can be deployed on premise and in cloud environments.

Tilde Speech Platform Web API provides methods to perform speech recognition in Latvian, Lithuanian and Estonian. For each language two types of recognition APIs are provided:

- Online or real-time. Typically used by mobile applications. Speech is recognized simultaneously as it is recorded and transmitted.
- Offline. Processing begins only when a full recording is submitted. This is intended for long audio recording recognition and allows to achieve better recognition quality.

Tilde ASR can work with any popular audio format and can even automatically extract the audio track from a video file. Tilde ASR can automatically perform:

- Speaker diarisation (only for offline API);
- Inverse text normalization by converting numbers, dates, acronyms, and abbreviations;
- Capitalization of proper names;
- Punctuation restoration.

License: Tilde ASR solution is a commercial cloud platform that requires a commercial license to access it. While the ELG platform is under construction and there are no billing and payment options in place, the services are available free for testing and development purposes, request limits can be introduced at any time.

Integration approach

As the Tilde ASR is a cloud-based platform and cannot currently be deployed into ELG platform directly, it was integrated is via the "proxy" approach. A small proxy image is deployed in the ELG platform as a Docker container containing system-specific parameters and access credentials. As the Tilde ASR platform provides an original API endpoint, methods and request lifecycle, this proxy transforms the requests and results according to the ELG specification to be ready for smooth integration with other ELG components.

[6] https://www.tilde.com/products-and-services/solutions-for-baltic-languages

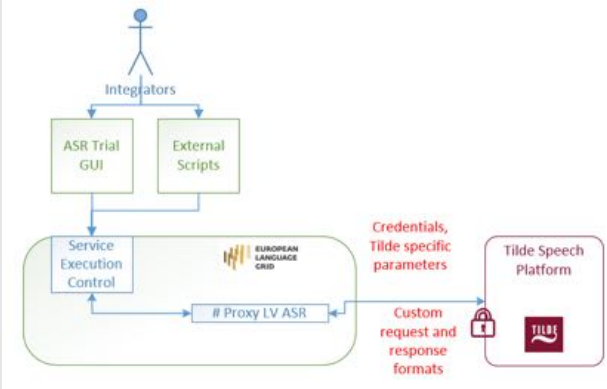

Figure 3. Tilde ASR service execution flow

Each Docker image contains access credentials to the production version of the Tilde ASR platform, therefore all Docker images are kept in the private image repository.

Deployed components

For each ASR system a Docker image has been created. The source code is pushed to Azure DevOps[7] code repository, and automatically an Azure Pipeline[8] is triggered, that pulls the source code for the proxy, adds configuration values, builds, tests, and tags Docker images, and finally pushes newly created images to the private Azure Container Registry[9]. Each Docker image version is tagged automatically with auto-increasing version number and with a static tag to manually control release versions.

The ELG platform deployment pulls that Docker image containing the proxy to the Tilde ASR platform, by providing image pull secrets.

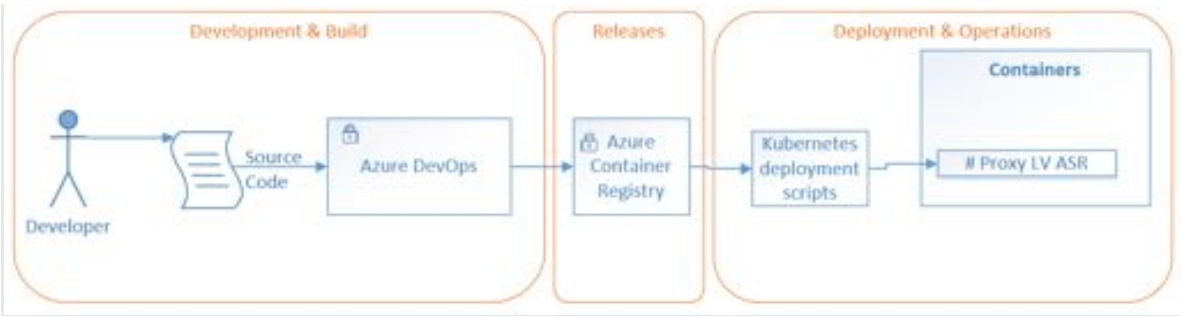

Figure 4. Development and deployment pipeline for Tilde ASR integration

Screenshots

The Tilde ASR service is registered in the ELG catalogue, thus a metadata view is available (see next image for details):

[7] https://azure.microsoft.com/en-us/services/devops/
[8] https://azure.microsoft.com/en-us/services/devops/pipelines/
[9] https://azure.microsoft.com/en-us/services/container-registry/

Figure 5. Tilde ASR service metadata view

To give the user the possibility to evaluate the results of an ASR service, a simple GUI for ASR services has been provided using the ELG portal. The GUI allows uploading an audio file or directly recorded sound within the browser; later the result is submitted to the Tilde ASR service. After the ASR process has finished, the end-user can see the output using the browser.

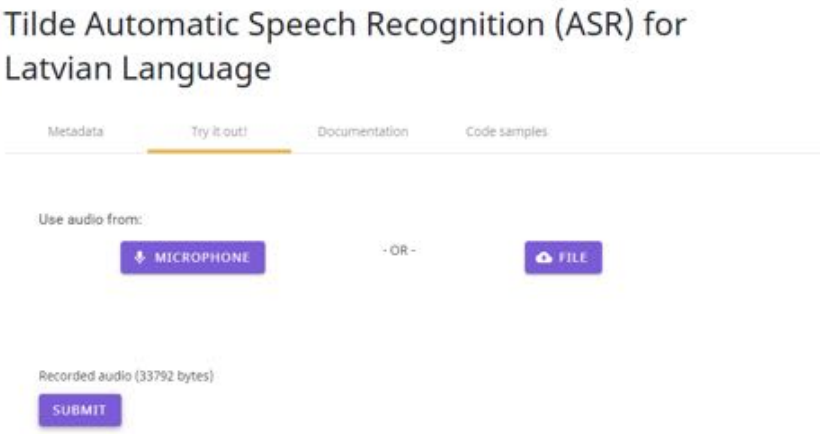

Figure 6. Tilde ASR service execution using a simple trial GUI

4.3 ASR Tools Integrated by UEDIN

The University of Edinburgh provides broadcast ASR models that were developed during the EU project SUMMA and the IARPA Material programme. The models were trained with the Kaldi toolkit using the semi-supervised approach as described in Carmantini, A., Bell, P., Renals, S. (2019) “Untranscribed Web Audio for Low Resource Speech Recognition”. Proceedings of Interspeech 2019.

License: UEDIN ASR models are released under a Creative Commons CC BY-SA 4.0 license.

Integration approach

The provided models are integrated into the ELG platform by SAIL LABS as described above. Furthermore, these models can be used with CloudASR and AlexASR toolkits by interested parties.

Deployed components

The provided models are deployed within the SAIL LABS platform (for details see Section 4.1). There are currently no screenshots for the UEdin ASR component as only ASR models are provided within the ELG project.

5 IE Tools, Services and Components (Task 4.3)

5.1 IE Tools Integrated by EXPSYS

Expert System brings to ELG its product Cogito Discover, a text analytics, extraction and categorization software to make information fully available for analytics, automation, robotics and more. The main challenge addressed in the integration of Cogito Discover in the ELG platform is the security assurance of a proprietary software as Cogito Discover while deploying and running it in ELG.

5.1.1 Cogito Discover

Cogito Discover is a scalable software platform for automatic semantic metadata generation and auto-classification that can be easily integrated in the production environment of document-processing applications or workflows. It can be deployed on premise and in cloud environments and it is available for both Linux and Windows systems.

Cogito Discover services that are included in the first release are:

- Part of Speech annotation: Annotations at different levels (token, word/compound word, group, clause, sentence) with grammatical types.
- Summarization: Annotation of the most relevant information: main syncons, main lemmas, main multiword expressions, main sentences and main domains.
- Named Entity Recognition: Annotation of entities, i.e., People, Organizations, Places, Known concepts, Unknown concepts. And also tags, i.e., URLs, mail addresses, phone numbers, addresses, dates, time, measures, money, percentage, file folder.
- Categorization: Classify documents using the IPTC taxonomy.
- Lemmatisation: This service returns the lemma of each concept spotted in the text that is modelled in the Cogito Discover knowledge graph.
- Sentiment analysis: Provides a sentiment score (positive or negative) for the entities recognized in the text, and an overall score for the whole set of entities in the document.
- Language detection: Identify the main language used in the text.

License: Cogito Discover is a proprietary software that requires a commercial license. Until the ELG platform billing system is available the services are available for free and limited by number characters processed. The limit will be defined according to the pilot needs.

Integration approach

For its deployment in ELG, Expert System has generated a docker image containing a Cogito Discover installation, and a set of adapters, one for each service to integrate, that manage the communication between the ELG platform and Cogito Discover (Figure 7). Each adapter provides support for different languages.

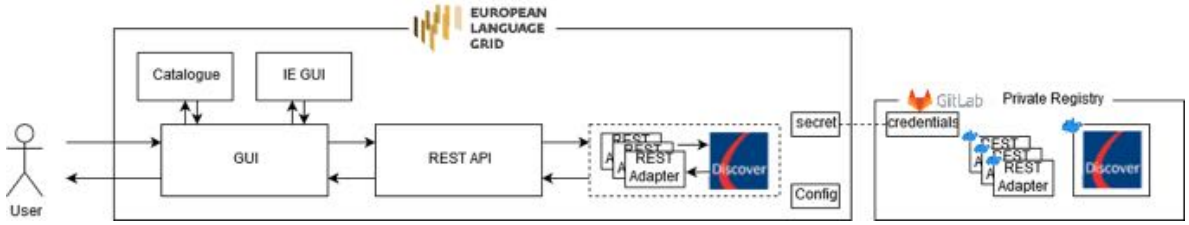

Figure 7. High level overview of the integration of Cogito Discover in the ELG

The ELG platform is informed about the Cogito Discover services by adding a configuration file that tells the platform where the tool and adapter images are so that they can be downloaded and deployed as containers. Since the registry is private, we have provisioned a dedicated namespace for EXPSYS in the ELG cluster as described in Section 3.2 where a secret can be created containing the registry credentials. Access to the namespace is limited to the owners and ELG administrators. Therefore, the private registry, the secret and the namespace are the components assuring the security of the Cogito Discover software for deployment and execution in the ELG.

When the user searches a service in the catalogue and selects a service, he/she has access to the service GUI, which is going to make a request to the public-facing REST API, which in turn calls the relevant Cogito Discover adapter. When the GUI receives the response from the REST API, it processes the response and shows it in a graphical way to the user. Note that we are not using only one adapter, since now we have more services. We decided to have one adapter per service, i.e., five adapters in total. Each adapter is going to be in a separate container, but since they are in the same pod, they are going to able to communicate with the Cogito Discover container locally.

Deployed components

Table 18 shows the software components, including images and code to generate them, that are deployed in ELG to support Cogito Discover integration in the platform. The code is available in the ELG project repository in GitLab and the images in the container registry in the same platform:

- Image registry base path: registry.gitlab.com/european-language-grid/
- Code repository base path: https://gitlab.com/european-language-grid/

The script that deploys the images in the ELG and enable the use of Cogito Discover services is available in the code repository: platform/infra/tree/master/elg-srv/srv/expsys

Type	Image name	Service	Languages	Container registry	Code repository
Tool	cogito-discover	Provides different services via adapters	see adapters	expertsystem/cogito-discover	Non-available (Commercial software)
Adapter	cogito-discover-ner-adapter	Named Entity recognition	English, French, German, Spanish	expertsystem/cogito-discover-ner-adapter	expertsystem/tree/master/cogito-discover-ner-adapter

Adapter	cogito-discover-pos-adapter	Part of Speech tagging	English, French, German, Spanish	expertsystem/cogito-discover-pos-adapter	expertsystem/tree/master/cogito-discover-pos-adapter
Adapter	cogito-discover-classification-adapter	Categorization	English	expertsystem/cogito-discover-classification-adapter	expertsystem/tree/master/cogito-discover-classification-adapter
Adapter	cogito-discover-semantic-adapter	Lemmatisation	English, French, German, Spanish	expertsystem/cogito-discover-semantic-adapter	expertsystem/tree/master/cogito-discover-semantic-adapter
Adapter	cogito-discover-summarization-adapter	Summarisation	English, French, German, Spanish	expertsystem/cogito-discover-summarization-adapter	expertsystem/tree/master/cogito-discover-summarization-adapter
Adapter	cogito-discover-sentiment-adapter	Sentiment analysis	English, French	expertsystem/cogito-discover-sentiment-adapter	expertsystem/tree/master/cogito-discover-sentiment-adapter
Adapter	cogito-discover-langdetect-adapter	Language detection	Czech, English, French, German, Spanish, Latvian, Greek	expertsystem/cogito-discover-langdetect-adapter	expertsystem/tree/master/cogito-discover-langdetect-adapter

Table 18. Cogito Discover software components for deployment in ELG

Screenshots

Figure 8 shows a list of some of the Cogito Discover services that are already integrated and available in the ELG catalogue. Figure 9 depicts an example of running the Cogito Discover Semantic Annotation service using the “Try it out!” option available in the ELG portal to test the different services.

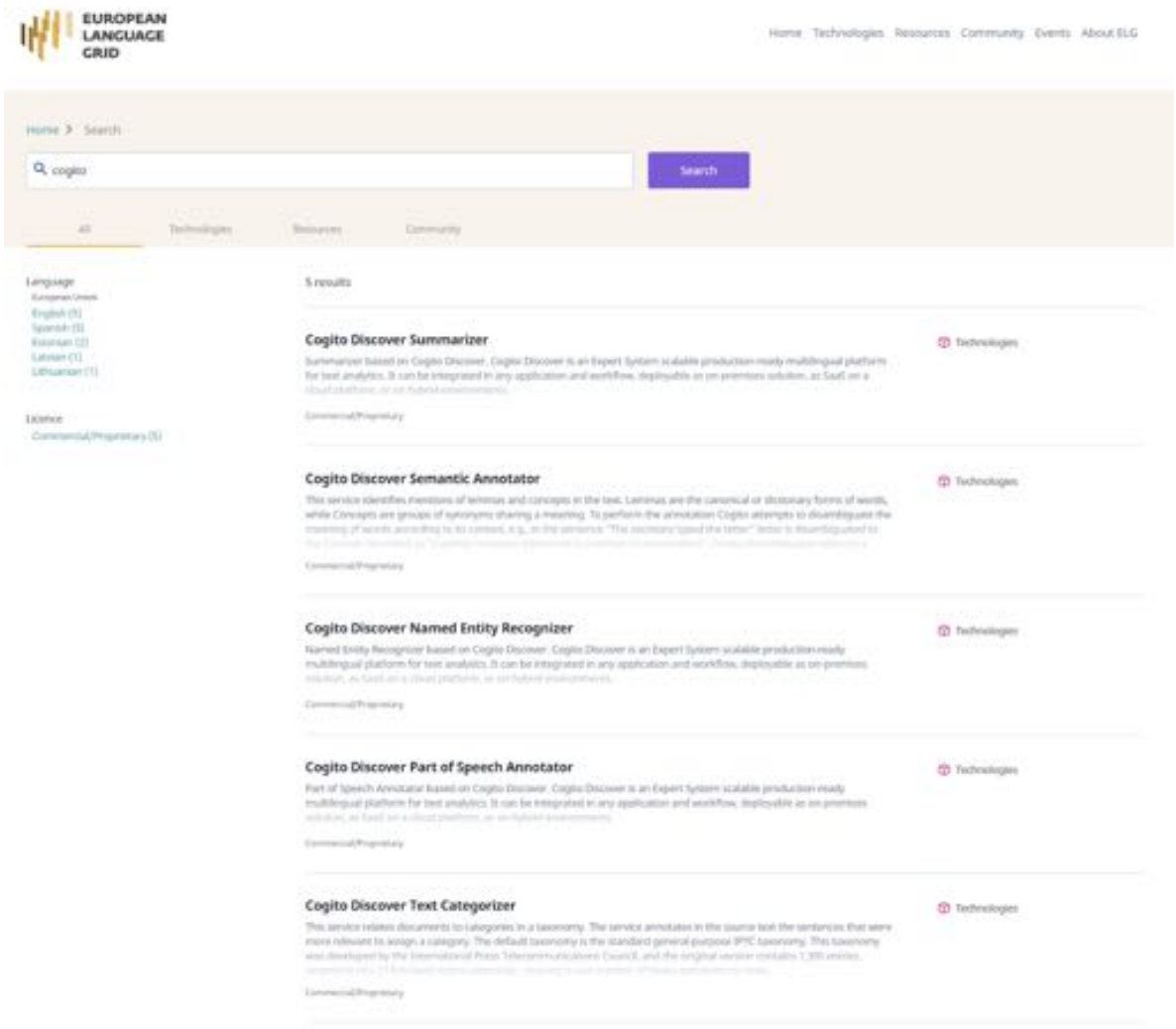

Figure 8. Cogito Discover services available in ELG

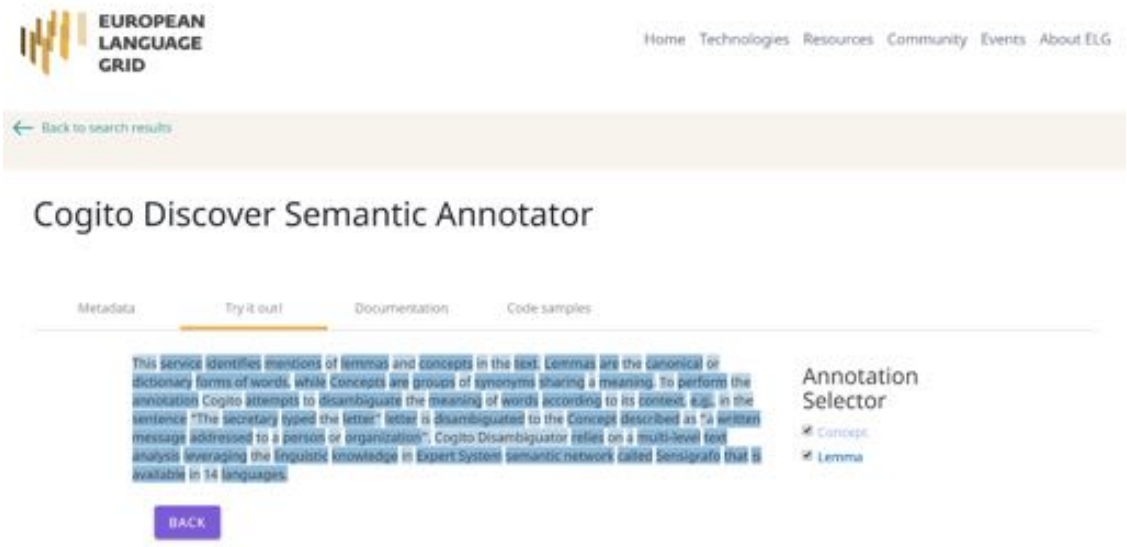

Figure 9. Cogito Discover semantic annotator service integrated in ELG

5.2 IE Tools Integrated by USFD

The University of Sheffield brings a variety of text analysis tools to the ELG platform, all based on the open-source GATE text processing framework. USFD already operates our own public platform for GATE-based annotation services called GATE Cloud, and we have implemented a "bridge" component to allow any existing GATE Cloud service to be exposed via the ELG platform. In addition, we have implemented a tighter integration for certain "flagship" services where the GATE application is run directly within the ELG cluster. It is a simple matter to convert other individual services from the bridged GATE Cloud style to the full integration style if they prove to be particularly popular.

5.2.1 GATE Cloud

GATE Cloud is a public text analysis platform operated by the GATE team at the University of Sheffield. The GATE framework is an open-source toolkit for developing text analysis applications, which has been in active development and use for over 20 years, and the GATE Cloud platform provides a way to take GATE-based text analysis pipelines and make them available as services on the web. GATE Cloud currently (February 2020) provides 63 separate GATE pipelines which can be called via a simple REST API to process individual documents, or can be run over larger batches of data using on-demand cloud computing capacity on Amazon Web Services.

Integration approach

For ELG, we created a proxy component that accepts requests in the ELG API format, converts and dispatches the request to the appropriate endpoint on GATE Cloud, and translates the response back to match ELG requirements. The code for this is publicly available on GitLab at https://gitlab.com/european-language-grid/usfd/elg-gate-cloud-bridge and is packaged as a docker image that runs as a container within the ELG cluster. In addition much of the code dealing specifically with exposing endpoints that conform to the ELG API specification has been factored out into a Spring Boot "starter" at https://gitlab.com/european-language-grid/usfd/elg-spring-boot-starter which can be used by any Java-based software for easy integration with the ELG specification – the deeper GATE integration described below also uses the starter, as do some of the services contributed by ILSP.

The code for both the starter and the proxy itself is public (under the Apache Licence v2), but in order to communicate with the GATE Cloud platform it requires a set of API credentials. GATE Cloud services are available without credentials but at very low daily quotas and with relatively slow rate limits enforced. Therefore, to run in the ELG we have set up a dedicated Kubernetes namespace that is private to USFD and contains a "secret" defining a set of GATE Cloud credentials with an elevated daily quota and rate limit compared to regular users of GATE Cloud.

GATE Cloud itself offers an API endpoint providing a listing in JSON of all the individual pipeline endpoints that the authenticating user can access, and the adapter uses this information to configure itself with an ELG-compatible endpoint for each available GATE Cloud service. Individual metadata records can then be registered with the ELG catalogue for each of the GATE Cloud services we wish to expose to the ELG.

Deployed components

The GATE Cloud services exposed via the bridge container for the first release of the ELG platform are listed in Table 19 (general purpose pipelines), Table 20 (Linked Data disambiguation) and Table 21 (domain-specific).

Name	Description	Languages/variants supported
Non-English Named Entity Recognition	Named entity recognition services for documents in some other languages. Based on ANNIE (see below), each one identifies names of persons, locations, and organizations but some also support other entity types.	German, French – the English ANNIE pipeline is described in the next section
Twitter language ID	Attempts to identify the language of tweets	Has classifiers for English, French, German and Spanish, among others
English Tweet Tokeniser	Identifies words and punctuation in tweets, including splitting up #hashtags into their component words	Tokeniser is tuned for English, but this pipeline includes the above language identifier as well
Part-of-Speech Tagger for English Tweets	Runs the above tokeniser pipeline plus a part-of-speech tagger tuned for Twitter data to assign parts of speech (verb, noun, etc.)	Models tuned for English, but the pipeline includes a language identifier
Named Entity Recogniser for Tweets	NER pipeline tuned for Twitter data. Identifies person, location, organization etc. and also performs normalization of abbreviations and common shorthands.	French, German – the English version has been integrated to run directly within the cluster and is described in a later section
Part of Speech and Morphological Analysis	PoS tagger and morphological analyser for English, which attempts to determine the root form for inflected nouns and verbs (e.g., “was” -> past tense of “be”)	English
Measurement expression annotator	Annotates numbers and measurement expressions, with their normalised values in SI units.	English, but many such expressions are language-independent
OpenNLP pipelines	Text analysis pipelines from Apache OpenNLP. All languages offer the tokeniser, sentence splitter and PoS tagger, English also includes NER	English, German
Universal Dependencies PoS Tagger	PoS tagger based on a simple window-based maximum entropy model and trained on the Universal Dependencies corpus.	Czech, English, French, Greek, Latvian, Spanish

Table 19. GATE Cloud services for general linguistic pre-processing and named entity recognition

Name	Description	Languages/variants supported
YODIE	Identifies named entities of various types and disambiguates them against DBpedia.	English, French, German, Spanish
BioYODIE	Identifies bio-medical named entities of various types including drugs, diseases, body parts, medical investigations, etc. and disambiguates them against various subsets of the UMLS	All UMLS, MeSH (medical subject headings) only, SNOMED only

Table 20. GATE Cloud services for Linked Data disambiguation

Name	Description	Languages/variants supported
“Brexit” analyser	A service for analysing tweets about the UK's referendum on EU membership. It identifies topics, hashtags, user mentions, and voting intention (where possible).	English

Name	Description	Languages/variants supported
DecarboNet environmental annotator	Identifies named entities, environmental terms, linguistic features and sentiment in social media texts. This pipeline was developed by the DecarboNet FP7 research project.	English, German
Political Futures Tracker	A service for analysing tweets about UK politics. It identifies topics, politicians, sentiment, abuse terms and more.	English
Rumour Veracity Classifier	Classifies tweets based on whether the claims they make are true, false or unverifiable.	English
SUMMA Text Summarisation	Extractive summarisation pipeline that selects key sentences from a text to form a summary	English, Spanish
Generic opinion mining	Recognises opinionated sentences in English text and classifies them as positive or negative. It also indicates emotion type, author and target of the opinion, average sentiment, nd some sentence types.	English
Twitter opinion mining	Variant of the generic opinion mining app that is tuned to work better on short texts such as tweets or other social media.	English
Twitter user classification	Attempts to classify the *author* of a tweet as a natural person or an organization based on clues within the user profile metadata. Note that this is an unusual application as it does not operate over the *text* but only over the user profile metadata, therefore it only works correctly on very specific types of input.	English

Table 21. Special-purpose GATE Cloud pipelines

5.2.2 GATE Pipelines running in the Cluster

Given the inherent latency and rate-limitation of delegating calls from the ELG platform over the internet to GATE Cloud, we have also implemented a tighter integration mechanism whereby a GATE pipeline can be run directly within a Docker container in the ELG infrastructure, accessed natively via the ELG-specified API. So as not to take a disproportionate share of the ELG platform processing capacity (which is still relatively limited at this stage of the project), only a small number of GATE services have been integrated via this mechanism for the first platform release, however the approach taken means that it is very simple to switch a given pipeline from integration via the GATE Cloud bridge to direct integration within the cluster, and this is transparent from the end-user perspective – the only difference they will notice is potentially faster response time and higher throughput.

Integration approach

The GATE framework is designed around reusable components supplied via a plugin mechanism – to build a GATE pipeline an LT developer loads the relevant plugins, then creates and configures the components they need for their application and connects them in the correct order. Documents then pass through the pipeline, with each component adding new annotations (metadata about specific segments of the document), or modifying or deleting annotations that were created by earlier steps. The result of processing is the final annotated document that emerges from the end of the pipeline.

The configuration of a GATE pipeline – the set of required plugins plus the list of components and their parameters – can be persisted in an XML format called XGAPP, which can then be transferred to any other software based on a compatible version of GATE and used to re-create the same pipeline automatically. To run GATE

pipelines within the ELG platform we can therefore create a single piece of software that knows how to load any XGAPP, then accept ELG API requests, convert the request into a GATE document, pass it through the pipeline, and extract the relevant final annotations to convert into the ELG API response. Different instances of this software can then be created for specific pipelines by parameterizing it with the relevant XGAPP and specification of the annotations of interest.

The generic software is https://gitlab.com/european-language-grid/usfd/gate-ie-worker on GitLab, it is open-source under the Apache licence v2 and is based on the same Spring Boot starter as the GATE Cloud proxy described above. It is built into a public Docker image registry.gitlab.com/european-language-grid/usfd/gate-ie-worker/base which contains the Java runtime, the GATE worker software, but *no* XGAPP. On top of this base image we then build a number of child images that add specific XGAPPs and their associated configuration files, at https://gitlab.com/european-language-grid/usfd/gate-ie-tools (all currently open-source, but the precise licence varies from pipeline to pipeline). Integrating a new GATE application is thus simply a matter of creating a new child image that embeds the relevant XGAPP at a known filesystem location, no new ELG-specific code needs to be written.

Since the base image and the currently-integrated pipelines are all open-source, as well as permitting direct integration within the ELG platform the images can be freely downloaded and run by anyone in their own Docker or Kubernetes instance. To facilitate this, as well as the ELG-specified API endpoint accepting JSON requests the GATE-IE-Worker can accept the direct post of plain text to be annotated, but still returning its result in the standard ELG response format.

Deployed components

Two GATE pipelines are integrated via this direct approach at this stage, but as mentioned above it is trivial to migrate any of the GATE Cloud bridged services to direct integration at a later stage if the levels of usage warrant it.

Name	Description	Languages/variants supported	Licence
ANNIE	English named entity recognizer, identifying mentions of persons, locations, organisations and dates, as well as other optional types Money (money amounts) and Percent (percentage expressions) if requested via a parameter.	English	LGPLv3 (the standard licence of GATE)
TwitIE	Named entity recognition service for Twitter data. Identifies person, location, organization etc. and also performs normalization of abbreviations and common shorthands (“brb”, “gr8”, “2day”,etc.) and attempts to split #hashtags into their component words.	English (other languages are integrated via the GATE Cloud bridge, described above)	GPLv3 (as TwitIE includes Stanford CoreNLP components, which are GPL licensed)

Table 22. GATE pipelines integrated directly in the ELG cluster

5.3 IE Tools Integrated by DFKI

DFKI provides a selection of tools from the two relevant groups located in Berlin (Speech and Language Technology) and Saarbrücken (Multilingual Language Technologies).

These services were mostly developed for English and German language data. Some of them focus upon Spanish, Dutch, French or Italian. The services cover the following tasks:

- Morphological analysis
- Named Entity Recognition
- Summarisation
- Tokenization
- Sentence splitting
- Language identification
- Social media geolocation

For most tools, we reference at least one corresponding paper publication as additional documentation.

Integration approach

The DFKI ELG team coordinates in consultation with the respective developers if and how the services can be used in the ELG. The integration of services into the ELG is done by the developers themselves, in some cases with the help of the DFKI ELG team. This way, DFKI can integrate a lot of services in a short amount of time. Additional services will follow at a later stage.

The individually developed services are exposed through the ELG platform independently of each other, but they share the containerization (Docker) and REST API endpoints.

Name	Description	Languages/variants supported	Licence	Code repository
geolocator	Social media geolocation prediction for a given Tweet, or a short text. Documentation: Philippe Thomas, Leonhard Hennig. **Twitter Geolocation Prediction Using Neural Networks.** In Bioinformatics. 2016 Sep 15. https://link.springer.com/chapter/10.1007/978-3-319-73706-5_21	English	GNU General Public License v3.0	https://github.com/Erechtheus/geolocation Docker: https://hub.docker.com/r/erechtheus79/geolocation
SETH	SETH finds gene mutation mentions in textual input and annotates them with several information: mutation type, mutation residue, wildtype residue, mutation position and Human Mutation Nomenclature (HGVS) grounding. Documentation: Philippe Thomas, Tim Rocktäschel, Jörg Hakenberg, Yvonne Lichtblau, Ulf Leser. **SETH detects and normalizes genetic variants in text.** *Bioinformatics*, Volume 32, Issue 18, 15 September 2016, Pages 2883–2885.	English	GNU General Public License v3.0	https://github.com/rockt/SETH Docker: https://hub.docker.com/r/erechtheus79/geolocation

Name	Description	Languages/variants supported	Licence	Code repository
	https://www.ncbi.nlm.nih.gov/pubmed/?term=27256315			
DKT NER	In the NER, we use two different approaches to recognise entities: (1) based on models and (2) based on dictionaries. This module has been implemented using the Apache OpenNLP tool. The model-based approach is using the NameFinderME module that is provided with a specific model (language-dependent). We have trained separate models for each entity type we distinguish. This means that we look for persons, organisations and locations serially, with three corresponding models applied consecutively. The list of tokens is analysed given the NER model provided. The dictionary-based approach uses the DictionaryNameFinder module. The list of tokens is analysed given the dictionary. Any entities that appear in the dictionary and also in the text are added to the list of entities. The dictionary consists of a mapping from entities to their URIs, so that when an entity is found in a text, the corresponding URI will also be added to the annotation. Documentation: • Peter Bourgonje, Julián Moreno Schneider, and Georg Rehm. Domain-specific Entity Spotting: Curation Technologies for Digital Humanities and Text Analytics. In Nils Reiter and Gerhard Kremer, editors, CUTE Workshop 2017 - CRETA Unshared Task zu Entitätenreferenzen. Workshop bei DHd2017, Berne, Switzerland, 2 2017. • Peter Bourgonje, Julián Moreno Schneider, Georg Rehm, and Felix Sasaki. Processing Document Collections to Automatically Extract Linked Data: Semantic Storytelling Technologies for Smart Curation Workflows. In Aldo Gangemi and Claire Gardent, editors, Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016), pages 13-16, Edinburgh, UK, 9 2016. The Association for Computational Linguistics.	English, German	N/A	N/A

Name	Description	Languages/variants supported	Licence	Code repository
	• Georg Rehm, Julián Moreno-Schneider, Jorge Gracia, Artem Revenko, Victor Mireles, Maria Khvalchik, Ilan Kernerman, Andis Lagzdins, Marcis Pinnis, Artus Vasilevskis, Elena Leitner, Jan Milde, and Pia Weißenhorn. Developing and Orchestrating a Portfolio of Natural Legal Language Processing and Document Curation Services. In Nikolaos Aletras, Elliott Ash, Leslie Barrett, Daniel Chen, Adam Meyers, Daniel Preotiuc-Pietro, David Rosenberg, and Amanda Stent, editors, Proceedings of Workshop on Natural Legal Language Processing (NLLP 2019), pages 55-66, Minneapolis, USA, 6 2019. Co-located with NAACL 2019. 7 June 2019.			
Lynx LER	The service for Legal Named Entity Recognition (LER) includes the elaboration of corresponding semantic classes and the preparation of a German language data set. Several state-of-the-art models were trained, i.e., Conditional Random Fields (CRFs) and bidirectional Long-Short Term Memory Networks (BiLSTMs). For training and evaluating the system we used a data set of German court decisions that was manually annotated with seven coarse-grained and 19 fine-grained classes: names and citations of people (person, judge, lawyer), location (country, city, street, area), organisation (organisation, company, institution, court, brand), legal norm (law, legal regulation, European legal norm), case-by-case regulation (regulation, contract), case law, and legal literature. The data set consists of approximately 67,000 sentences and around 54,000 annotated entities. For the experiment, two tools for sequence labeling were chosen. These are sklearn-crfsuite (CRFs) (https://sklearn-crfsuite.readthedocs.io/en/latest/) and UKPLab-BiLSTM (BiLSTMs) (https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf). Documentation: • Elena Leitner, Georg Rehm, and Julián Moreno-Schneider. Fine-grained	German	CC-BY 4.0	https://github.com/elenanereiss/Legal-Entity-Recognition

Name	Description	Languages/variants supported	Licence	Code repository
	Named Entity Recognition in Legal Documents. In Maribel Acosta, Philippe Cudré-Mauroux, Maria Maleshkova, Tassilo Pellegrini, Harald Sack, and York Sure-Vetter, editors, Semantic Systems. The Power of AI and Knowledge Graphs. Proceedings of the 15th International Conference (SEMANTiCS 2019), number 11702 in Lecture Notes in Computer Science, pages 272-287, Karlsruhe, Germany, 9 2019. Springer. 10/11 September 2019. • Leitner, E. (2019). Eigennamen- und Zitaterkennung in Rechtstexten. Bachelor's thesis, Universität Potsdam, Potsdam, 2.			
Lynx Summ	Our text summarization model is based on Neural Networks and computes an abstractive summary. Documentation: • Dmitrii Aksenov, Julián Moreno-Schneider, Peter Bourgonje, Robert Schwarzenberg, Leonhard Hennig, and Georg Rehm. Abstractive Text Summarization based on Language Model Conditioning and Locality Modeling. In: LREC 2020.	English	N/A	N/A
Qurator LangIdent	Generate language profiles from Wikipedia abstract xml. Detect language of a text using naive Bayesian filter. 99% over precision for 53 languages (af, ar, bg, bn, ca, cs, cy, da, de, el, en, es, et, fa, fi, fr, gu, he, hi, hr, hu, id, it, ja, kn, ko, lt, lv, mk, ml, mr, ne, nl, no, pa, pl, pt, ro, ru, sk, sl, so, sq, sv, sw, ta, te, th, tl, tr, uk, ur, vi, zh-cn, zh-tw). Documentation: • Shuyo, Nakatani. Language Detection Library for Java. http://code.google.com/p/language-detection/	53 languages (see description)	N/A	N/A
JTok	JTok provides a Java-based configurable tokenizer that identifies paragraphs, sentences and tokens of an input text. Tokens can be further classified into abbreviations, numbers, punctuation, etc. JTok currently supports English, German and Italian, but also comes with a default configuration that can be used for other languages. The output	English, German	GNU LGPL v2.1	https://github.com/dfki-mlt/jtok

Name	Description	Languages/variants supported	Licence	Code repository
	of JTok is an instance of de.dfki.lt.tools.tokenizer.annotate.AnnotatedString, but there are methods available that transform an AnnotatedString into an XML representation or into instances of Paragraph, TextUnit and Token classes.			
MMorph 3	MMorph is a morphological analyser, originally developed by ISSCO at the University of Geneva in the past MULTEXT project (1995). The tool can be used in a pipeline architecture to annotate words with a morphosyntactic description containing free-definable attribute and values. Documentation: • Dominique Petitpierre and Graham Russell, 1995. MMORPH: The Multext morphology program. Multext deliverable 2.3.1, ISSCO, University of Geneva . http://www.issco.unige.ch/downloads/multext/mmorph.doc.ps.tar.gz • Stefania Racioppa and Thierry Declerck, 2019 . Enriching Open Multilingual Wordnets with Morphological Features. In Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari 2019. http://ceur-ws.org/Vol-2481/paper62.pdf	Dutch, English, French, German, Italian, Spanish	N/A	https://gitlab.com/european-language-grid/dfki/mmorph3

Table 23. DFKI tools to be integrated in the ELG first release

5.4 IE Tools Integrated by SAIL

SAIL LABS brings to the ELG platform a set of IE components which are based on the same SAIL Media Mining Indexer (MMI, https://www.sail-labs.com/media-mining-system/#MMIndexer) that is the basis for the ASR tools introduced in Section 4.1, as well as an individual tool for the identification of language from text.

The component for the identification of language from text (*langClassifier*) is based on a component which forms part of SAIL's suite of tools for the processing of (textual) data from the Internet and Social Media.

Within the set of IE tools of ELG, SAIL provides the MMI for the recognition of named-entities (NER) and for Sentiment Analysis (SA) and the langClassifier (LID) for the detection of language from text.

- NER is supported for 6 languages: Czech, English, French, German, Greek, and Spanish.
- SA is supported for 4 languages: English, French, German and Spanish.
- LID: is supported for 6 languages: Czech, English, French, German, Greek, and Spanish (or *unknown*).

NER can be configured to be based on a set of patterns as well as to work on a set of features based on the sequence of words and morphological features. Currently, only the pattern-based functionality has been in-

cluded. It contains rudimentary morphological processing. NER within the Media Mining environment is typically used in combination with tokenization and text-cleaning components which have not been integrated into NER implementation for ELG (but this is planned for future updates of the respective components).

SA is based on a set of patterns of "sentiment carrying words and expressions" and performs a 4-way categorization into the classes "positive", "negative", "neutral" and "mixed". The differentiation between these four classes allows for the distinction between emotionally laden expressions as opposed to neutral ones. In the process, effects such as negation, boosting, the interpretation of idioms etc. are handled by the component. Sentiment is analysed on a per sentence level as well as on the overall document level. Different mechanisms for score-fusion have been implemented – the currently implemented algorithm for the ELG component uses the best-performing method according to a set of evaluations. Furthermore, the component can be parameterized for the processing of text from social media and also be parameterized to include domain-specific jargon and terminology (e.g., for crisis management or *radical content*). The component as included in the ELG only offers the basic functionality. Updates and extensions are planned for future releases.

LID is based on a set of components which are employed within the SAIL LABS Media Mining System for the identification as well as the verification of languages of text-content. The underlying models combine TF-IDF-derived word-based features with character-based n-gram features.

Integration approach

For each of the above-mentioned components concerning the MMI, the MMI and the respective model have been packaged into a container. Communication takes place via the ELG-API, parameters being passed according to the specifications of Task 2.5. Output is produced according to the same specifications.

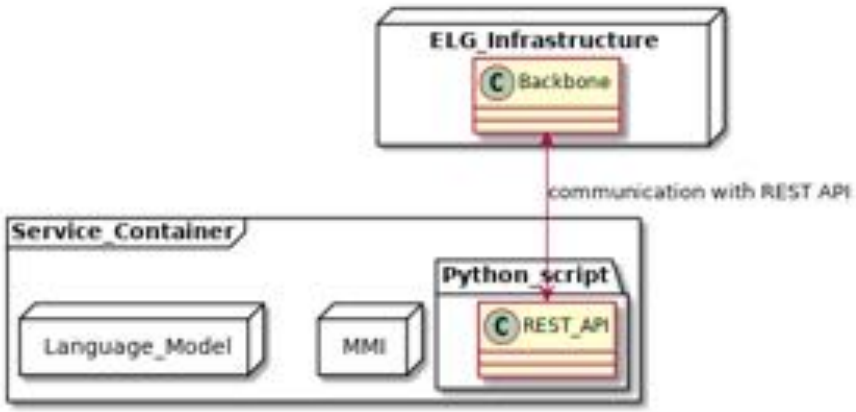

Figure 10. Integration approach for SAIL MMI-based tools

The component for LID has been implemented using a simpler structure as it does not require the MMI to be present. Rather, a wrapper-script has been written which performs the processing of assigning a set of languages and associated scores to an input text. This set of languages together with the scores is returned as the result of LID. As opposed to the MMI-based containers, the LID container combines LID for six languages in a single component.

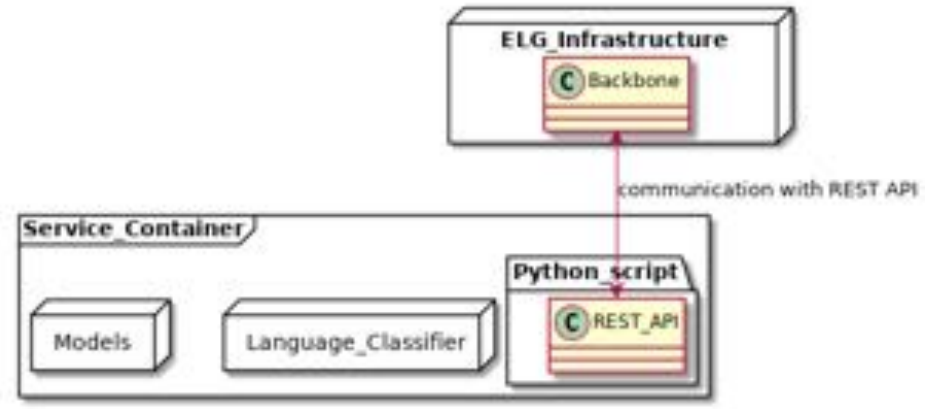

Figure 11. Integration approach for SAIL LID tool

The general approach followed for the integration of the respective component into the ELG included the following steps:

- Creation of a REST endpoint to consume the input from ELG and produce the result expected by ELG
- Each component (technology x language) was packaged as a separate docker image. In the future, all models are envisioned to be provided on shared storage, alleviating this constraint and making the creation of components more flexible (via parameters)
- Metadata records have been created for each service

Deployed components

Table 24 shows the software components that are deployed in ELG platform. The code is available in the ELG project repository in gitlab and the images in the container registry in the same platform. Though, all code and images are private projects as they are commercial software.
Container registry: sail/<image name> (full path: registry.gitlab.com/european-language-grid/sail/<image name>)
Code repository: sail/<image name> (full path: https://gitlab.com/european-language-grid/sail/<image name>)

Type	Tool Type	Image name	Service	Languages/variants supported	Licence
tool	NER	sail-ned-en	Detection of a set of named-entities	English	SAIL LABS
tool	NER	sail-ned-fr	Detection of a set of named-entities	French	SAIL LABS
tool	NER	sail-ned-de	Detection of a set of named-entities	German	SAIL LABS
tool	NER	sail-ned-el	Detection of a set of named-entities	Greek	SAIL LABS
tool	NER	sail-ned-es	Detection of a set of named-entities	Spanish	SAIL LABS
tool	NER	sail-ned-cs	Detection of a set of named-entities	Czech	SAIL LABS
tool	SA	sail-sed-en	Sentiment Detection	English	SAIL LABS
tool	SA	sail-sed-fr	Sentiment Detection	French	SAIL LABS
tool	SA	sail-sed-de	Sentiment Detection	German	SAIL LABS
tool	SA	sail-sed-es	Sentiment Detection	Spanish	SAIL LABS

Type	Tool Type	Image name	Service	Languages/variants supported	Licence
tool	LID	sail-lid	Language ID from text	English, German, French, Spanish, Greek, Czech	SAIL LABS

Table 24. Text processing tools provided by SAIL Labs

5.5 IE Tools Integrated by ILSP

ILSP brings to the ELG platform five information extraction (IE) tools; one for English and four for Greek texts. Each tool is actually a pipeline of several components that run one after the other. The first components of each pipeline perform basic linguistic preprocessing (e.g., tokenization, sentence splitting, POS Tagging, lemmatization etc); the output of this preprocessing is then fed to the tool-specific components that extract/generate the required annotations for the task (e.g., named annotations, sentiment targets etc). All the tool-specific components have been developed based on the GATE text engineering platform; mainly by using JAPE grammars and gazetteers. For basic linguistic preprocessing in English, widely known NLP tools wrapped with the GATE framework have been used. For Greek we employ tools developed by ILSP which are deployed as UIMA-AS services. The output of these UIMA-AS services (XMI) is translated to GATE documents which are then passed (as already said) to the tool-specific GATE components.

The five tools that ILSP provide are the following:

- Named Entity Recognition (English): A rule-based engine designed for the automatic recognition and classification of Named Entities of the following six types: PERSON, LOCATION, ORGANIZATION, FACILITY, GPE (Geo-political entity), and MISCELLANEOUS (for entities of other types).
- Named Entity Recognition (Greek): A rule-based engine designed for the automatic recognition and classification of Named Entities of the following five types: PERSON, LOCATION, ORGANIZATION, FACILITY, and GPE (Geo-political entity).
- Event Extractor for protest events (Greek): A rule-based engine designed to detect different types of protest events (e.g., demonstration/march, strike) recorded in Greek news data, along with their structural components (e.g., actors, targets, location, and time of the events) following a specific event coding schema.
- Event Extractor for physical attacks (Greek): A rule-based engine designed to detect different types of physical attacks (e.g., attack against personal property, assault against life, sexual assault) recorded in Greek news data, along with their structural components (e.g., actors, targets, location, and time of the events) following a specific event coding schema.
- Opinion mining / target detection for customer reviews (Greek): A rule-based engine designed for target-based sentiment analysis in Greek customer review texts. Given as input a review text about a particular target entity (e.g., a restaurant or a hotel), the engine automatically extracts the detected opinion expressions and their targets along with the respective (positive/negative) sentiment polarity label.

Integration approach

For integrating the aforementioned LT tools to the ELG platform the following steps were followed (as it is required in the ELG specifications):

- For each tool a java application that exposes a REST endpoint and consumes/produces messages that follow the ELG format was created. The application uses the respective IE tool to process the content of the received messages and of course it returns the appropriate responses. The development of this application was based on a Spring Boot "starter" (developed by USFD, see Section 5.2.1) that offers an easy way to expose ELG compatible REST endpoints. For Greek IE tools basic linguistic preprocessing is done by invoking the respective services deployed in separate containers; see the figure below.
- Each IE application was packaged as a separate Docker image. The download locations of the produced images are given in the next section. The code for creating the images and for exposing the ELG compatible endpoints is available in a common GitLab repository
- A separate namespace ("elg-srv-ilsp") with restricted access was allocated in ELG's Kubernetes cluster for the ILSP tools. Also, the appropriate resource files that specify how the ILSP containers will be deployed at Kubernetes were created.
- Finally, we have created a metadata record of each IE tool that will be uploaded to the ELG catalogue.

Figure 12. Deployed containers for ILSP IE tools

Deployed components

Table 25 contains the ILSP IE services that are deployed to ELG platform. They are all currently under a proprietary licence, but this is under review and may change in a future release.

Service/Application	Language	Docker image location	License
Named Entity Recognition	English	registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services/ie:nercgr	ILSP ToS
Named Entity Recognition	Greek	registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services/ie:nercen	ILSP ToS
Event Extractor for protest events	Greek	registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services/ie:protestsgr	ILSP ToS
Event Extractor for physical attacks	Greek	registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services/ie:attacksgr	ILSP ToS
Opinion mining/target detection for customer reviews	Greek	registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services/ie:opiniongr	ILSP ToS

Table 25. ILSP text analysis services in the ELG first release

5.6 IE Tools Integrated by CUNI

5.6.1 UDPipe

UDPipe is a trainable pipeline for tokenization, tagging, lemmatization and dependency parsing. UDPipe is language-agnostic and can be trained given annotated data in CoNLL-U format. As of UD 2.4, trained models are available for 60 languages: Afrikaans, Ancient Greek, Arabic, Armenian, Basque, Belarusian, Bulgarian, Catalan, Chinese, Classical Chinese, Coptic, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Galician, German, Gothic, Greek, Hebrew, Hindi, Hungarian, Indonesian, Irish, Italian, Japanese, Korean, Latin, Latvian, Lithuanian, Maltese, Marathi, North Sami, Norwegian Bokmål, Norwegian Nynorsk, Old Church Slavonic, Old French, Old Russian, Persian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swedish, Tamil, Telugu, Turkish, Ukrainian, Urdu, Uyghur, Vietnamese, Wolof.

License: UDPipe itself is available under MPL 2.0 open-source license. However, the trained models are distributed using the non-commercial CC BY-NC-SA 4.0 license

Integration approach

UDPipe itself provides a REST server able to process simultaneous requests. The requests can each require a different model (language), and the server automatically keeps the 20 recently least used models loaded. We have therefore decided to reuse the official UDPipe REST server, an implement an adapter to convert the input and output format to ELG annotations.

Given the rich set of annotations generated by UDPipe, we also implemented a GUI for visualizing all annotations produced by UDPipe.

Deployed components

All components used in ELG deployment of UDPipe are summarized in Table 26.

Docker Image	Description	Code Repository
registry.gitlab.com/european-language-grid/cuni/**srv-udpipe**	Official UDPipe REST server and UD 2.4 models.	https://gitlab.com/european-language-grid/cuni/**srv-udpipe**
registry.gitlab.com/european-language-grid/cuni/**srv-udpipe-adapter**	An adapter relaying requests to UDPipe REST server and converting the results to ELG format.	https://gitlab.com/european-language-grid/cuni/**srv-udpipe-adapter**
registry.gitlab.com/european-language-grid/cuni/**gui-udpipe**	UDPipe GUI	https://gitlab.com/european-language-grid/cuni/**gui-udpipe**

Table 26. UDPipe components deployed in ELG

We have packaged the official UDPipe REST server together with UD 2.4 models in the Docker image **srv-udpipe**. Because the UDPipe REST server communicates prevalently in CoNLL-U format, we have implemented an adapter, which converts the CoNLL-U format to ELG format. The adapter is an asynchronous Python HTTP server which routes the requests to the UDPipe REST server and then converts the results to annotations in ELG format. The adapter is available under MPL 2.0 license and is packages in the Docker image **srv-udpipe-adapter**.

For a given text, UDPipe generates sentences, tokens, words, lemmas, part-of-speech tags and labeled dependency trees. For visual presentation of all these annotations to the user, we have prepared a GUI application which for every input sentence draws a tree, and displays all information for a selected word in the tree. The GUI is packaged in the Docker image **gui-udpipe**.

5.6.2 NameTag

NameTag is an open-source tool for named entity recognition (NER). Currently, models for Czech and English are available. However, we have a prototype (not currently deployed) achieving amongst others state-of-the-art results on CoNLL Czech, English, Dutch and Spanish datasets .

License: NameTag itself is available under MPL 2.0 open-source license. However, the trained models are distributed using the non-commercial CC BY-NC-SA license

Integration approach

NameTag itself provides a REST server able to process simultaneous requests. We have therefore decided to reuse it an implement an adapter to convert the inputs and outputs to ELG format.

Given that NER is provided by several services in ELG, we reuse the existing GUI.

Deployed components

All components used in ELG deployment of NameTag are summarized in Table 27.

Docker Image	Description	Code Repository
registry.gitlab.com/european-language-grid/cuni/**srv-nametag**	Official NameTag REST server and Czech+English models.	https://gitlab.com/european-language-grid/cuni/**srv-nametag**
registry.gitlab.com/european-language-grid/cuni/**srv-nametag-adapter**	An adapter relaying requests to NameTag REST server and converting the results to ELG format.	https://gitlab.com/european-language-grid/cuni/**srv-nametag-adapter**

Table 27. NameTag components deployed in ELG

We have packaged the official NameTag REST server together with Czech and English models in the Docker image **srv-nametag**. Furthermore, we have implemented an adapter which converts the inputs and outputs of the NameTag REST server to ELG format. The adapter is an asynchronous Python HTTP which routes the requests to the NameTag REST server and then converts the XML outputs to annotations in ELG format. The adapter is available under MPL 2.0 license and is packages in the Docker image **srv-nametag-adapter**.

6 MT Tools, Services and Components (Task 4.4)

6.1 MT Tools and Models integrated by UEDIN

6.1.1 MT Toolkit development

UEDIN is a major contributor to the open-source Marian Toolkit for Neural Machine Translation[10]. While most toolkits for neural MT are based on python, Marian is written in C++. It is one of the fastest neural MT toolkits on the market. Within the context of ELG, major development effort at UEDIN went into the design and implementation of a Marian-based REST translation service that can be deployed in a Docker container. Marian itself is released under the MIT license; it relies on a number of 3rd-party components as shown in Table 28. We developed a process for building the Marian REST server Docker image entirely through Docker containers[11]. The license information regarding the Nvidia CUDA components in Table 28 refers to this setup.

Component	License
Marian	MIT
NVIDIA Container Toolkit	Apache 2.0
NVIDIA NCCL	NVIDIA Corporation; allows redistribution. Appears to be BSD without explicitly stating so.
Intel Math Kernel Library	Intel Simplified Software License[12]
Google Sentencepiece	Apache 2.0

[10] https://marian-nmt.github.io
[11] https://github.com/ugermann/marian-docker
[12] https://software.intel.com/en-us/license/intel-simplified-software-license

Component	License
FBGEMM	BSD
CROW HTTP Server	BSD
Sentence Splitter with language-specific resources from the Moses Toolkit	LGPL-2.1

Table 28. Marian and Marian third-party component licenses

The Marian REST server Docker Image is available on Dockerhub (mariannmt/marian-rest-server); instructions for building Images with integrated trained models are available on https://github.com/ugermann/marian-docker.

The Marian REST server API natively conforms with the overall ELG API design described in Section 3.1. Translation requests are sent to the server as JSON documents via the HTTP POST method. Translation requests containing a single translation request (text string) are structured as follows, using the "text" request format.

```
{
  "type":"text",
  "params":{
    "inputFormat":"sentence|paragraph|wrappedText"
  },
  "content":"The content, as a string inline",
  "mimeType":"text/plain",
}
```

Valid values for the inputFormat parameter are:

- "sentence": one sentence per line; no sentence splitting is performed by the server
- "paragraph": each line is a paragraph; sentences will be split but end-of-line counts as end-of-sentence
- "wrappedText": paragraphs are separated by a blank line; end-of-line (except for blank lines) is replaced by whitespace, then paragraphs are split into sentences.

Currently, the return format is the same as the input format.

Translation requests can also be batched into a single HTTP call using the "structuredText" request type from Section 3.1, again the "inputFormat" parameter is interpreted as above for each of the "texts" in the request

Successful translation requests return responses in the "texts" response format:

```
{
  "response":{
    "type":"texts",
    "texts":[
      {
        "content":"string of translated/transcribed text", // either
        "texts":[/* same structure, recursive */],         // or
       }
     ]
   }
}
```

Errors are reported using the usual "failure" response format.

6.1.2 MT Models

For the M14 release, UEDIN is providing two models for translation from German to English and vice versa. These are based on a "basic" transformer model (Vaswani et al, 2017) and trained on data released for the WMT 2019 Shared Task on News Translation[13] plus Release 5.0 of the Paracrawl Data collection for German/English[14].

First, initial models were trained in both translation directions on only the available high-quality parallel data: EuroParl v9.0, News Commentary, and the Rapid Corpus of EU Press Releases, all available from the shared task page referenced above. These models were then used to score the ParaCrawl data (a) with respect to translation quality according to Junczys-Dowmunt's (2018) Dual Conditional Cross-Entropy score, and monolingually with a 5-gram model trained with KenLM (Heafield, 2011) on the respective NewsCrawl data from 2007 to 2018[15]. All sentence pairs in the ParaCrawl corpus were ranked by a linear combination of Dual Conditional Cross-Entropy score and the per-word language model score in the respective source language. The top-ranking 10 million sentence pairs were added to the pool of training data. A new model was then trained from scratch.

Docker Images with integrated models for German-to-English and English-to-German translation have been published on Dockerhub as edinburghnlp/mt4elg-de-en and edinburghnl/mt4elg-en-de, respectively. These models are released under CC-BY-SA 4.0.

Metadata records and Kubernetes resource files for these MT Components were adapted from corresponding files for Greek-to-English and English-to-Greek translation provided by ILSP.

6.2 MT Tools Integrated by ILSP

ILSP is providing base transformer models for Greek-to-English and English-to-Greek translation, also trained and deployed with the Marian toolkit. They were trained over a Greek-English parallel corpus of 10.7M sentence pairs. The following table contains the ILSP MT tools that are deployed to ELG. The Docker images are in the repository.

Service/Application	Language	Docker Image[16]	License
ILSP-MT	English -> Greek	mt:engr	ILSP ToS
ILSP-MT	Greek -> English	mt:gren	ILSP ToS

Table 29. ILSP MT tools integrated in the first ELG release

[13] https://www.statmt.org/wmt19/translation-task.html#download
[14] https://paracrawl.eu/
[15] http://data.statmt.org/news-crawl/de/; http://data.statmt.org/news-crawl/en/
[16] Repository: registry.gitlab.com/ilsp-nlpli-elg/ilsp-nlp-mt-model/

6.3 MT Tools Integrated by CUNI

CUNI provides machine translation models trained using the Tensor2tensor toolkit[17]. The toolkit is implemented in Python and distributed under Apache License 2.0. It allows straightforward training and deployment of state-of-the-art MT models.

Currently, several MT models trained by CUNI are deployed on the Lindat website[18]. Additionally, we provide a deployable Docker image containing TensorFlow model server[19] as a model backend and MTMonkey frontend[20] for client-server communication.

However, since MTMonkey does not conform to the proposed ELG REST API design at the moment and is unnecessarily robust for the ELG deployment, we decided to replace it with a lightweight RESTful frontend written in Python. It processes translation requests in a JSON format which are sent via HTTP POST method and forwards them to the TensorFlow model server in a compatible format for translation. Later, it processes output from the model server and translates it to the ELG compatible response. For simplicity, we adapt Marian REST server API proposed by UEDIN in Section 6.1, including requests, responses and error messages. We do not plan any modification to the API since it suits our current MT system capabilities.

6.3.1 MT Models

CUNI currently provides several stable and experimental MT models between the following languages: English, German, French, Czech, Russian and Hindi. They are based on the "big" transformer model (Vaswani et al., 2017) and were trained on the WMT 2018 Shared Task on News Translation. Further improvements suggested by Popel (2018) were applied during the model training, namely back-translation of monolingual data with later filtering of the synthetic parallel data using coreference resolution. The CUNI submission for English-Czech language pair achieved the best performance on the shared task in 2018.

6.4 MT Tools Integrated by DFKI

DFKI provides a machine translation quality estimation (QE) tool from the Speech and Language Technology team in Berlin. It works for three language pairs, namely English to French, English to German and English to Spanish.

"Qualitative"[21] is a python toolkit for ranking and selection of sentence-level output by different MT systems using Quality Estimation. The toolkit implements a basic pipeline for annotating the given sentences with black-box features. Consequently, it applies a machine learning mechanism in order to rank data based on models pre-trained on human preferences. The preprocessing pipeline includes support for language models, PCFG parsing, language checking tools and various other pre-processors and feature generators. The code follows the principles of object-oriented programming to allow modularity and extensibility.

This comparative quality estimation software for machine translation has been described in Avramidis (2016a; 2016b) and Avramidis et al. (2014). Research experiments have been conducted and described in Avramidis

[17] https://github.com/tensorflow/tensor2tensor
[18] https://lindat.mff.cuni.cz/services/translation/
[19] https://www.tensorflow.org/tfx/guide/serving
[20] https://github.com/ufal/mtmonkey
[21] https://github.com/lefterav/qualitative

(2012a; 2012b; 2013; 2017), Avramidis et al. (2011; 2016) and Avramidis and Popovic (2013). Qualitative is exposed through the ELG platform using Docker containerization and REST API endpoints. It has a LGPL license.

6.5 MT Tools Integrated by TILDE

6.5.1 TILDE MT platform

The Tilde MT platform provides all the facilities for customizing neural machine translation (NMT) engines for specific languages and domains. The platform also includes a suite of sophisticated linguistic components that provide linguistic knowledge for MT systems – tokenizers, sentence breakers, morphological analysers and synthesizers, lemmatizers, part-of-speech and morpho-syntactic taggers, syntactic/dependency parsers, and named entity recognizers – as well as sophisticated tools for the correct processing of tags and placeholders, including HTML code.

The Tilde MT platform provides a full-service environment for private and publicly available systems that are hosted on the cloud and can be integrated into any platform or application. Alongside public MT systems, the user can convert its raw data into a fully customized MT system for specific language pairs, domains, and use cases.

The MT platform provides a full cycle of MT needs:

1. Data processing
2. Corpora collection
3. MT training and tuning
4. Terminology management
5. Quality estimation
6. Linguistic tool development
7. API integration
8. e-Services
9. Customer support

The platform consists of various components:

1. Administrative web interface – for training and managing translation systems
2. Public web interface for translating with public MT systems
3. API for accessing translation services using a machine interface (provides communication using the SOAP standard or RESTful approach)
4. MT integration into other platforms:
    a. HTML/JavaScript based widget
    b. Computer-aided translation tools
5. ITS 2.0 data categories support
6. Additional output segmentation information (phrases and their translations)

License: The Tilde MT solution is a commercial cloud platform that requires a commercial license to access it. While the ELG platform is under construction and there is no billing and payment option in place, the services are available free for testing and development purposes, request limits can be introduced at any time.

As the Tilde MT is cloud-based platform and cannot currently be deployed into the ELG platform directly, small proxies have been created. Each proxy is deployed in the ELG platform as Docker container. For each trained MT system, a separate Docker image has been provided containing system specific parameters, access credentials. As the existing Tilde MT platform provides original API endpoint, methods and request lifecycle, these small proxies transform the requests and results according to the ELG specification to be ready for a smooth integration with other ELG components.

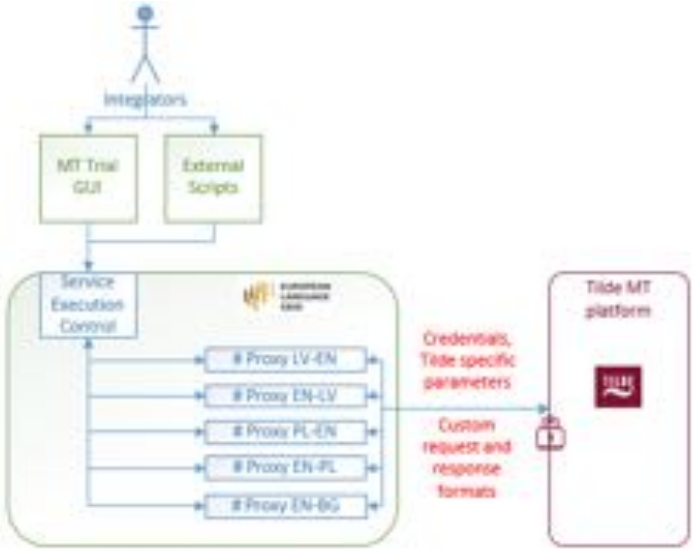

Figure 13. Tilde MT service execution flow

For each MT system (one for each language pair) a Docker image has been created. The source code is pushed to Azure DevOps code repository, and automatically an Azure Pipeline is triggered, that pulls the source code for the proxy, adds configuration values, builds, tests, and tags Docker images, and finally pushes newly created images to the private Azure Container Registry. Each Docker image version is tagged automatically with an auto-increasing version number and with a static tag to manually control release versions.

The ELG platform deployment pulls Docker images containing proxies to the Tilde MT platform, by providing image pull secrets.

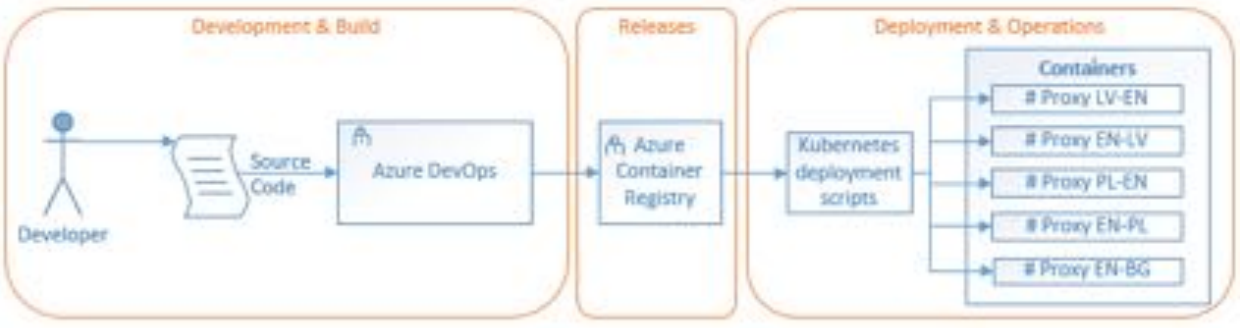

Figure 14. Development and deployment pipeline for Tilde MT integration

Following MT proxies has been deployed to the ELG cloud:

- Latvian-English
- English-Latvian
- Polish-English
- English-Polish
- English-Bulgarian

7 Integration of other Types of Tools, Services, Components (Task 4.5)

As explained above in Section 2, since other tools beyond the three principal classes are of lesser relevance to the call for pilot projects, their integration has mostly been postponed until later releases of the ELG platform. For this release the only "other" tool to be integrated is the text-to-speech system of Tilde.

7.1 Text to Speech Tools integrated by TILDE

Tilde TTS is part of Tilde's Speech platform which provides speech solutions[22] for the Baltic languages.

Tilde Speech platform is a scalable software platform for automatic speech recognition and speech synthesis that can be easily integrated using Web APIs. It can be deployed on premises and on cloud environments.

The Tilde Speech Platform Web API allows to synthesize any Latvian or Lithuanian text using different voices (two for Latvian, two for Lithuanian), change pitch and tempo of the synthesized speech, output synthesized audio as raw PCM or compressed MP3.

Tilde TTS automatically performs text normalization and converts numbers, dates, acronyms, and abbreviations. An automatic sentence boundary detection allows Tilde TTS to process text sentence by sentence and stream audio as it is being synthesized, thus minimizing latency.

License: Tilde TTS solution is a commercial cloud platform that requires a commercial license to access it. While the ELG platform is under construction and there is no billing and payment options in place, the services are available free for testing and development purposes, request limits can be introduced at any time.

Integration approach

As the Tilde TTS is a cloud based platform and cannot currently be deployed into the ELG platform directly, small proxies have been created. Each proxy is deployed in the ELG platform as Docker container. For each TTS voice (trained TTS system), a separate Docker image has been provided containing voice specific parameters, access credentials. As the existing Tilde TTS platform provides an original API endpoint, methods and request lifecycle, these small proxies transform the requests and results according to the ELG specification to be ready for a smooth integration with other ELG components.

---

[22] https://www.tilde.com/products-and-services/solutions-for-baltic-languages

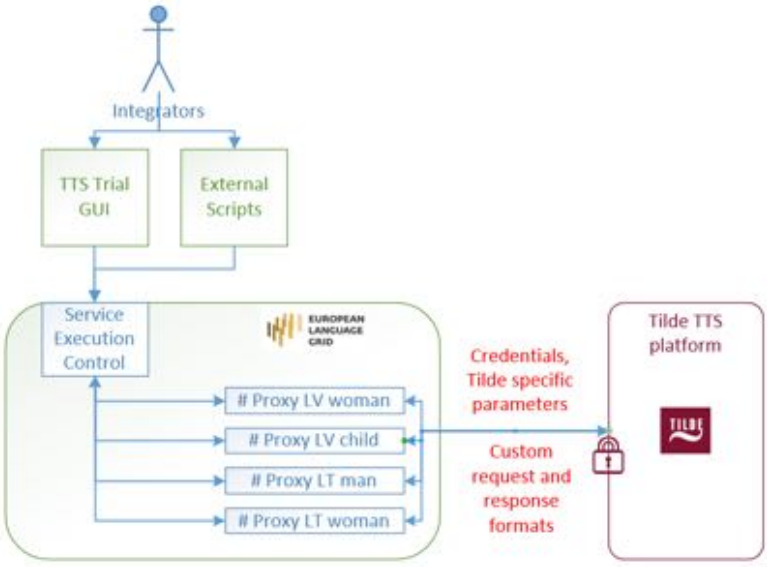

Figure 15. Tilde TTS service execution flow

Each Docker image contains access credentials to the production version of Tilde TTS platform, therefore all the Docker images are kept in a private image repository.

Deployed components

For each TTS system a Docker image has been created. The source code is pushed to Azure DevOps[23] code repository, and automatically an Azure Pipeline[24] is triggered, that pulls the source code for the proxy, adds configuration values, builds, tests, and tags Docker images, and finally pushes newly created images to the private Azure Container Registry[25]. Each Docker image version is tagged automatically with an auto-increasing version number and with a static tag to manually control release versions.

The ELG platform deployment pulls that Docker images containing proxies to the Tilde TTS platform, by providing image pull secrets.

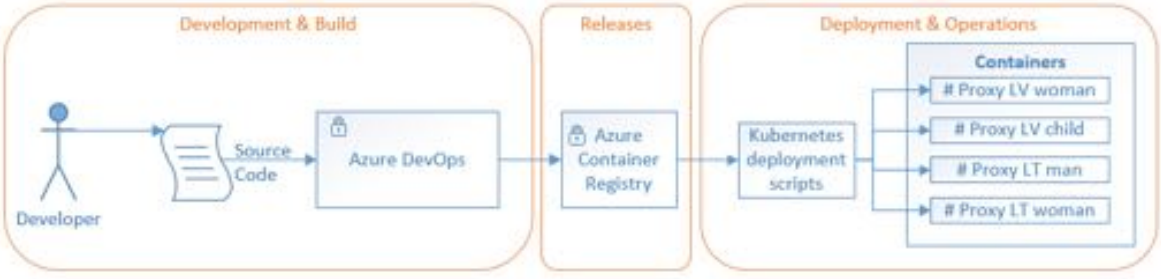

Figure 16. Development and deployment pipeline for Tilde TTS integration

[23] https://azure.microsoft.com/en-us/services/devops/
[24] https://azure.microsoft.com/en-us/services/devops/pipelines/
[25] https://azure.microsoft.com/en-us/services/container-registry/

The following TTS proxies has been deployed to the ELG cloud:

- Tilde Text-To-Speech (TTS) for Latvian Language, Child's Voice
- Tilde Text-To-Speech (TTS) for Latvian Language, Woman's Voice
- Tilde Text-To-Speech (TTS) for Lithuanian Language, Man's Voice
- Tilde Text-To-Speech (TTS) for Lithuanian Language, Woman's Voice

Screenshots

Each TTS service is registered in the ELG catalogue, thus a metadata view is available (see Figure 17 for details)

Figure 17. Tilde TTS service metadata view

To give the user the possibility to evaluate the results of the TTS services, a simple GUI was created. The GUI allows to enter text which is submitted to the TTS service. After the TTS process has been finished, the end-user can listen to the output through the browser or download the result as an audio file.

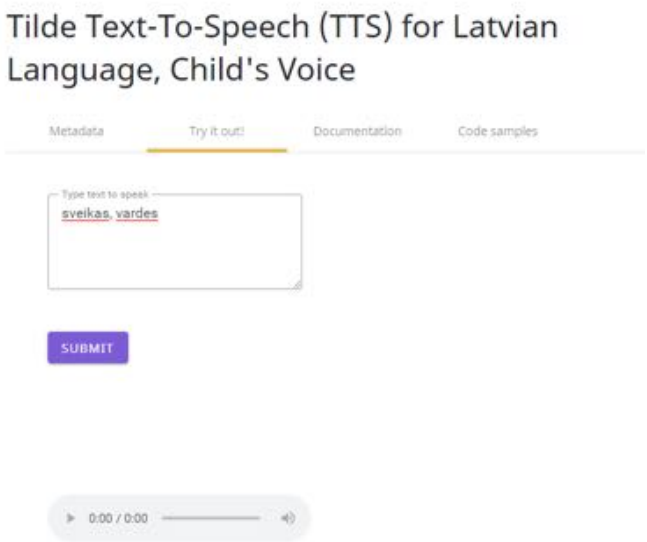

Figure 18. Tilde TTS service execution using simple trial GUI

8 Conclusion

Our survey (Section 2) of tools available for integration from among the project partners discovered a wide range of different tools and services covering many different functions, languages, and technical and licensing constraints. We believe we have determined a reasonable way of prioritising this long list of tools for integration at the various stages of the project and agreed an ambitious but manageable set to integrate into the first platform release at M16 using the API specification and containerisation framework described in Section 3. The services described in the remaining sections of this document will be augmented in the subsequent ELG Deliverables D4.2 and D4.3 as the set of integrated services grows through the second and third platform releases.

9 Bibliography

Avramidis, E. "Comparative Quality Estimation: Automatic Sentence-Level Ranking of Multiple Machine Translation Outputs". *Proceedings of 24th International Conference on Computational Linguistics. International Conference on Computational Linguistics (COLING-12), December 8-15, Mumbai, India*, 2012, pp. 115-132.

Avramidis, E. "Quality Estimation for Machine Translation output using linguistic analysis and decoding features". *Proceedings of the Seventh Workshop on Statistical Machine Translation. Workshop on Statistical Machine Translation (WMT-12), located at The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, June 7-8, Montreal, Canada*, 2012.

Avramidis, E. "Sentence-level ranking with quality estimation". *Machine Translation (MT), volume 28*, 2013, pp. 1-20.

Avramidis, E. "Interoperability in MT Quality Estimation or wrapping useful stuff in various ways". *Proceedings of the LREC 2016 Workshop "Translation Evaluation – From Fragmented Tools and Data Sets to an Integrated Ecosystem", Georg Rehm, Aljoscha Burchardt et al. (eds.)*, 2016 (a).

Avramidis, E. "Qualitative: Python Tool for MT Quality Estimation Supporting Server Mode and Hybrid MT". *The Prague Bulletin of Mathematical Linguistics No. 106*, 2016 (b), pp. 147–158.doi: 10.1515/pralin-2016-0014.

Avramidis, E. "Comparative Quality Estimation for Machine Translation Observations on Machine Learning and Features". *The Prague Bulletin of Mathematical Linguistics No. 108*, 2017, pp. 307–318.doi: 10.1515/pralin-2017-0029.

Avramidis, E.; Burchardt, A.; Macketanz, V.; Srivastava, A.. "DFKI's system for WMT16 IT-domain task, including analysis of systematic errors". *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 415-422.

Avramidis, E.; Popovic, M.. "Machine learning methods for comparative and time-oriented Quality Estimation of Machine Translation output". *Proceedings of the Eighth Workshop on Statistical Machine Translation. Workshop on Statistical Machine Translation (WMT-13), August 8-9, Sofia, Bulgaria*, 2013, pp. 329-336.

Avramidis, E.; Popovic, M.; Torres, D.V.; Burchardt, A.. "Evaluate with Confidence Estimation: Machine ranking of translation outputs using grammatical features". *Proceedings of the Sixth Workshop on Statistical Machine Translation. Workshop on Statistical Machine Translation (WMT-11), located at EMNLP, July 30-31, Edinburgh, United Kingdom*, 2011, pp. 65-70.

Junczys-Dowmunt, M. "Dual Conditional Cross-Entropy Filtering of Noisy Parallel Corpora". *Proceedings of the Third Conference on Machine Translation: Shared Task Papers, Association for Computational Linguistics*, 2018, 888-895

Heafield, K. KenLM: "Faster and Smaller Language Model Queries". *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Association for Computational Linguistics, 2011, 187-197

Popel, M. "CUNI Transformer Neural MT System for WMT18". In Proceedings of WMT 2018, Brussels, Belgium, 2018, 486–491

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, u. & Polosukhin, I. "Attention is All you Need". *Advances in Neural Information Processing Systems 30, Curran Associates, Inc.,* 2017, 5998-6008.

A. Appendix

		M14		
Provider	**Tool**	**Service**	**Lang. - Category**	
CUNI	NameTag	Named Entity Recognition	Czech	A
CUNI	NameTag	Named Entity Recognition	English	A
CUNI	UDPipe parser	Dependency Parsing	Czech	A
CUNI	UDPipe parser	Dependency Parsing	English	A
CUNI	UDPipe parser	Dependency Parsing	French	A
CUNI	UDPipe parser	Dependency Parsing	German	A
CUNI	UDPipe parser	Dependency Parsing	Greek	A
CUNI	UDPipe parser	Dependency Parsing	Latvian	A
CUNI	UDPipe parser	Dependency Parsing	Spanish	A
CUNI	UDPipe tagger	Lemmatisation	Czech	A
CUNI	UDPipe tagger	Lemmatisation	English	A
CUNI	UDPipe tagger	Lemmatisation	French	A
CUNI	UDPipe tagger	Lemmatisation	German	A
CUNI	UDPipe tagger	Lemmatisation	Greek	A
CUNI	UDPipe tagger	Lemmatisation	Latvian	A
CUNI	UDPipe tagger	Lemmatisation	Spanish	A
CUNI	UDPipe tagger	Morphological analyser	Czech	A
CUNI	UDPipe tagger	Morphological analyser	English	A
CUNI	UDPipe tagger	Morphological analyser	French	A
CUNI	UDPipe tagger	Morphological analyser	German	A
CUNI	UDPipe tagger	Morphological analyser	Greek	A
CUNI	UDPipe tagger	Morphological analyser	Latvian	A
CUNI	UDPipe tagger	Morphological analyser	Spanish	A
CUNI	UDPipe tagger	Part of Speech tagging	Czech	A
CUNI	UDPipe tagger	Part of Speech tagging	English	A
CUNI	UDPipe tagger	Part of Speech tagging	French	A
CUNI	UDPipe tagger	Part of Speech tagging	German	A
CUNI	UDPipe tagger	Part of Speech tagging	Greek	A
CUNI	UDPipe tagger	Part of Speech tagging	Latvian	A
CUNI	UDPipe tagger	Part of Speech tagging	Spanish	A
CUNI	UDPipe tokenizer	Tokenization	Czech	A
CUNI	UDPipe tokenizer	Tokenization	English	A
CUNI	UDPipe tokenizer	Tokenization	French	A
CUNI	UDPipe tokenizer	Tokenization	German	A
CUNI	UDPipe tokenizer	Tokenization	Greek	A
CUNI	UDPipe tokenizer	Tokenization	Latvian	A
CUNI	UDPipe tokenizer	Tokenization	Spanish	A
DFKI	geolocator	Categorization	English	A
DFKI	JTok	Sentence splitting	English	A
DFKI	JTok	Sentence splitting	German	A
DFKI	JTok	Tokenization	English	A
DFKI	JTok	Tokenization	German	A
DFKI	Lynx-Legal NER	Named Entity Recognition	German	A
DFKI	Lynx-NER	Named Entity Recognition	English	A
DFKI	Lynx-NER	Named Entity Recognition	German	A
DFKI	Lynx/QURATOR Sum	Summarization	English	A
DFKI	MMorph3	Morphological analyser	English	A
DFKI	MMorph3	Morphological analyser	French	A
DFKI	MMorph3	Morphological analyser	German	A
DFKI	MMorph3	Morphological analyser	Spanish	A
DFKI	Qurator-LangIdent	Language identification	Czech	A

		M14		
Provider	**Tool**	**Service**	**Lang. - Category**	
DFKI	Qurator-LangIdent	Language identification	English	A
DFKI	Qurator-LangIdent	Language identification	French	A
DFKI	Qurator-LangIdent	Language identification	German	A
DFKI	Qurator-LangIdent	Language identification	Greek	A
DFKI	Qurator-LangIdent	Language identification	Latvian	A
DFKI	Qurator-LangIdent	Language identification	Spanish	A
DFKI	SETH	Named Entity Recognition	English	A
Expert System	Cogito Discover	Categorization	English	A
Expert System	Cogito Discover	Categorization	Spanish	A
Expert System	Cogito Discover	Language identification	Czech	A
Expert System	Cogito Discover	Language identification	English	A
Expert System	Cogito Discover	Language identification	French	A
Expert System	Cogito Discover	Language identification	German	A
Expert System	Cogito Discover	Language identification	Greek	A
Expert System	Cogito Discover	Language identification	Latvian	A
Expert System	Cogito Discover	Language identification	Spanish	A
Expert System	Cogito Discover	Lemmatisation	English	A
Expert System	Cogito Discover	Lemmatisation	French	A
Expert System	Cogito Discover	Lemmatisation	German	A
Expert System	Cogito Discover	Lemmatisation	Spanish	A
Expert System	Cogito Discover	Named Entity Recognition	English	A
Expert System	Cogito Discover	Named Entity Recognition	French	A
Expert System	Cogito Discover	Named Entity Recognition	German	A
Expert System	Cogito Discover	Named Entity Recognition	Spanish	A
Expert System	Cogito Discover	Part of Speech tagging	English	A
Expert System	Cogito Discover	Part of Speech tagging	French	A
Expert System	Cogito Discover	Part of Speech tagging	German	A
Expert System	Cogito Discover	Part of Speech tagging	Spanish	A
Expert System	Cogito Discover	Sentiment Analysis	English	A
Expert System	Cogito Discover	Sentiment Analysis	French	A
Expert System	Cogito Discover	Summarization	English	A
Expert System	Cogito Discover	Summarization	French	A
Expert System	Cogito Discover	Summarization	German	A
Expert System	Cogito Discover	Summarization	Spanish	A
ILSP	ILSP-ABSA	Sentiment Analysis	Greek	A
ILSP	ILSP-Events-physical-attack	Information Extraction	Greek	A
ILSP	ILSP-Events-protest	Information Extraction	Greek	A
ILSP	ILSP-NER	Named Entity Recognition	English	A
ILSP	ILSP-NER	Named Entity Recognition	Greek	A
SAIL LABS	SAIL language ID	Language identification	Czech	A
SAIL LABS	SAIL language ID	Language identification	English	A
SAIL LABS	SAIL language ID	Language identification	French	A
SAIL LABS	SAIL language ID	Language identification	German	A
SAIL LABS	SAIL language ID	Language identification	Greek	A
SAIL LABS	SAIL language ID	Language identification	Spanish	A
SAIL LABS	SAIL NER	Named Entity Recognition	Czech	A
SAIL LABS	SAIL NER	Named Entity Recognition	English	A
SAIL LABS	SAIL NER	Named Entity Recognition	French	A
SAIL LABS	SAIL NER	Named Entity Recognition	German	A
SAIL LABS	SAIL NER	Named Entity Recognition	Greek	A
SAIL LABS	SAIL NER	Named Entity Recognition	Spanish	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	English	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	French	A

		M14		
Provider	**Tool**	**Service**	**Lang. - Category**	
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	German	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Spanish	A
USFD	BioYODIE (Full)	NER Disambiguation	English	A
USFD	BioYODIE (MeSH Only)	NER Disambiguation	English	A
USFD	BioYODIE (Snomed)	NER Disambiguation	English	A
USFD	Brexit Analyzer	Categorization	English	A
USFD	Brexit Analyzer	Named Entity Recognition	English	A
USFD	DecarboNET Environmental Annotator	Named Entity Recognition	English	A
USFD	DecarboNET Environmental Annotator	Named Entity Recognition	German	A
USFD	GATE Cloud: ANNIE	Named Entity Recognition	English	A
USFD	GATE Cloud: French NER	Named Entity Recognition	French	A
USFD	GATE Cloud: French NER for Tweets	Named Entity Recognition	French	A
USFD	GATE Cloud: Generic Opinion Mining	Opinion Mining	English	A
USFD	GATE Cloud: Generic Opinion Mining for Tweets	Opinion Mining	English	A
USFD	GATE Cloud: German NER	Named Entity Recognition	German	A
USFD	GATE Cloud: German NER for Tweets	Named Entity Recognition	German	A
USFD	GATE Cloud: Language ID for Tweets	Language identification	English	A
USFD	GATE Cloud: Language ID for Tweets	Language identification	French	A
USFD	GATE Cloud: Language ID for Tweets	Language identification	German	A
USFD	GATE Cloud: Language ID for Tweets	Language identification	Spanish	A
USFD	GATE Cloud: Measurement Annotator	Number annotation	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Named Entity Recognition	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Named Entity Recognition	German	A
USFD	GATE Cloud: OpenNLP Pipelines	Part of Speech tagging	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Part of Speech tagging	German	A
USFD	GATE Cloud: OpenNLP Pipelines	Sentence splitting	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Sentence splitting	German	A
USFD	GATE Cloud: OpenNLP Pipelines	Tokenization	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Tokenization	German	A
USFD	GATE Cloud: POS and Morph	Morphological analyser	English	A
USFD	GATE Cloud: POS and Morph	Part of Speech tagging	English	A

		M14		
Provider	**Tool**	**Service**	**Lang. - Category**	
USFD	GATE Cloud: Tweet POS Tagger	Part of Speech tagging	English	A
USFD	GATE Cloud: Tweet to-kenizer	Tokenization	English	A
USFD	Political Futures Tracker	Categorization	English	A
USFD	Political Futures Tracker	Named Entity Recognition	English	A
USFD	Rumour Veracity Classifier	Categorization	English	A
USFD	SUMMA	Summarization	English	A
USFD	SUMMA	Summarization	Spanish	A
USFD	Tweet User Classification	Categorization	English	A
USFD	Tweet User Classification	Named Entity Recognition	English	A
USFD	TwitIE	Named Entity Recognition	English	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Czech	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	French	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Greek	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Latvian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Spanish	A
USFD	YODIE	NER Disambiguation	English	A
USFD	YODIE	NER Disambiguation	French	A
USFD	YODIE	NER Disambiguation	German	A
USFD	YODIE	NER Disambiguation	Spanish	A

Table 30. IE and Text Analysis tools and services to integrate in the first release (full list)

		M22		
Provider	**Tool**	**Service**	**Lang. - Cate-gory**	
CUNI	UDPipe parser	Dependency Parsing	Bulgarian	A
CUNI	UDPipe parser	Dependency Parsing	Croatian	A
CUNI	UDPipe parser	Dependency Parsing	Danish	A
CUNI	UDPipe parser	Dependency Parsing	Dutch	A
CUNI	UDPipe parser	Dependency Parsing	Estonian	A
CUNI	UDPipe parser	Dependency Parsing	Finnish	A
CUNI	UDPipe parser	Dependency Parsing	Hungarian	A
CUNI	UDPipe parser	Dependency Parsing	Irish	A
CUNI	UDPipe parser	Dependency Parsing	Italian	A
CUNI	UDPipe parser	Dependency Parsing	Lithuanian	A
CUNI	UDPipe parser	Dependency Parsing	Maltese	A
CUNI	UDPipe parser	Dependency Parsing	Polish	A
CUNI	UDPipe parser	Dependency Parsing	Portuguese	A
CUNI	UDPipe parser	Dependency Parsing	Romanian	A
CUNI	UDPipe parser	Dependency Parsing	Slovak	A
CUNI	UDPipe parser	Dependency Parsing	Slovenian	A
CUNI	UDPipe parser	Dependency Parsing	Swedish	A
CUNI	UDPipe parser	Dependency Parsing	Basque	B

M22

Provider	Tool	Service	Lang. - Category	
CUNI	UDPipe parser	Dependency Parsing	Catalan	B
CUNI	UDPipe parser	Dependency Parsing	Galician	B
CUNI	UDPipe parser	Dependency Parsing	Norwegian	B
CUNI	UDPipe parser	Dependency Parsing	Serbian	B
CUNI	UDPipe parser	Dependency Parsing	Turkish	B
CUNI	UDPipe parser	Dependency Parsing	Ukrainian	B
CUNI	UDPipe tagger	Lemmatisation	Bulgarian	A
CUNI	UDPipe tagger	Lemmatisation	Croatian	A
CUNI	UDPipe tagger	Lemmatisation	Danish	A
CUNI	UDPipe tagger	Lemmatisation	Dutch	A
CUNI	UDPipe tagger	Lemmatisation	Estonian	A
CUNI	UDPipe tagger	Lemmatisation	Finnish	A
CUNI	UDPipe tagger	Lemmatisation	Hungarian	A
CUNI	UDPipe tagger	Lemmatisation	Irish	A
CUNI	UDPipe tagger	Lemmatisation	Italian	A
CUNI	UDPipe tagger	Lemmatisation	Lithuanian	A
CUNI	UDPipe tagger	Lemmatisation	Maltese	A
CUNI	UDPipe tagger	Lemmatisation	Polish	A
CUNI	UDPipe tagger	Lemmatisation	Portuguese	A
CUNI	UDPipe tagger	Lemmatisation	Romanian	A
CUNI	UDPipe tagger	Lemmatisation	Slovak	A
CUNI	UDPipe tagger	Lemmatisation	Slovenian	A
CUNI	UDPipe tagger	Lemmatisation	Swedish	A
CUNI	UDPipe tagger	Lemmatisation	Basque	B
CUNI	UDPipe tagger	Lemmatisation	Catalan	B
CUNI	UDPipe tagger	Lemmatisation	Galician	B
CUNI	UDPipe tagger	Lemmatisation	Norwegian	B
CUNI	UDPipe tagger	Lemmatisation	Serbian	B
CUNI	UDPipe tagger	Lemmatisation	Turkish	B
CUNI	UDPipe tagger	Lemmatisation	Ukrainian	B
CUNI	UDPipe tagger	Morphological analyser	Bulgarian	A
CUNI	UDPipe tagger	Morphological analyser	Croatian	A
CUNI	UDPipe tagger	Morphological analyser	Danish	A
CUNI	UDPipe tagger	Morphological analyser	Dutch	A
CUNI	UDPipe tagger	Morphological analyser	Estonian	A
CUNI	UDPipe tagger	Morphological analyser	Finnish	A
CUNI	UDPipe tagger	Morphological analyser	Hungarian	A
CUNI	UDPipe tagger	Morphological analyser	Irish	A
CUNI	UDPipe tagger	Morphological analyser	Italian	A
CUNI	UDPipe tagger	Morphological analyser	Lithuanian	A
CUNI	UDPipe tagger	Morphological analyser	Maltese	A
CUNI	UDPipe tagger	Morphological analyser	Polish	A
CUNI	UDPipe tagger	Morphological analyser	Portuguese	A
CUNI	UDPipe tagger	Morphological analyser	Romanian	A
CUNI	UDPipe tagger	Morphological analyser	Slovak	A
CUNI	UDPipe tagger	Morphological analyser	Slovenian	A
CUNI	UDPipe tagger	Morphological analyser	Swedish	A
CUNI	UDPipe tagger	Morphological analyser	Basque	B
CUNI	UDPipe tagger	Morphological analyser	Catalan	B

M22

Provider	Tool	Service	Lang. - Category	
CUNI	UDPipe tagger	Morphological analyser	Galician	B
CUNI	UDPipe tagger	Morphological analyser	Norwegian	B
CUNI	UDPipe tagger	Morphological analyser	Serbian	B
CUNI	UDPipe tagger	Morphological analyser	Turkish	B
CUNI	UDPipe tagger	Morphological analyser	Ukrainian	B
CUNI	UDPipe tagger	Part of Speech tagging	Bulgarian	A
CUNI	UDPipe tagger	Part of Speech tagging	Croatian	A
CUNI	UDPipe tagger	Part of Speech tagging	Danish	A
CUNI	UDPipe tagger	Part of Speech tagging	Dutch	A
CUNI	UDPipe tagger	Part of Speech tagging	Estonian	A
CUNI	UDPipe tagger	Part of Speech tagging	Finnish	A
CUNI	UDPipe tagger	Part of Speech tagging	Hungarian	A
CUNI	UDPipe tagger	Part of Speech tagging	Irish	A
CUNI	UDPipe tagger	Part of Speech tagging	Italian	A
CUNI	UDPipe tagger	Part of Speech tagging	Lithuanian	A
CUNI	UDPipe tagger	Part of Speech tagging	Maltese	A
CUNI	UDPipe tagger	Part of Speech tagging	Polish	A
CUNI	UDPipe tagger	Part of Speech tagging	Portuguese	A
CUNI	UDPipe tagger	Part of Speech tagging	Romanian	A
CUNI	UDPipe tagger	Part of Speech tagging	Slovak	A
CUNI	UDPipe tagger	Part of Speech tagging	Slovenian	A
CUNI	UDPipe tagger	Part of Speech tagging	Swedish	A
CUNI	UDPipe tagger	Part of Speech tagging	Basque	B
CUNI	UDPipe tagger	Part of Speech tagging	Catalan	B
CUNI	UDPipe tagger	Part of Speech tagging	Galician	B
CUNI	UDPipe tagger	Part of Speech tagging	Norwegian	B
CUNI	UDPipe tagger	Part of Speech tagging	Serbian	B
CUNI	UDPipe tagger	Part of Speech tagging	Turkish	B
CUNI	UDPipe tagger	Part of Speech tagging	Ukrainian	B
CUNI	UDPipe tokenizer	Tokenization	Bulgarian	A
CUNI	UDPipe tokenizer	Tokenization	Croatian	A
CUNI	UDPipe tokenizer	Tokenization	Danish	A
CUNI	UDPipe tokenizer	Tokenization	Dutch	A
CUNI	UDPipe tokenizer	Tokenization	Estonian	A
CUNI	UDPipe tokenizer	Tokenization	Finnish	A
CUNI	UDPipe tokenizer	Tokenization	Hungarian	A
CUNI	UDPipe tokenizer	Tokenization	Irish	A
CUNI	UDPipe tokenizer	Tokenization	Italian	A
CUNI	UDPipe tokenizer	Tokenization	Lithuanian	A
CUNI	UDPipe tokenizer	Tokenization	Maltese	A
CUNI	UDPipe tokenizer	Tokenization	Polish	A
CUNI	UDPipe tokenizer	Tokenization	Portuguese	A
CUNI	UDPipe tokenizer	Tokenization	Romanian	A
CUNI	UDPipe tokenizer	Tokenization	Slovak	A
CUNI	UDPipe tokenizer	Tokenization	Slovenian	A
CUNI	UDPipe tokenizer	Tokenization	Swedish	A
CUNI	UDPipe tokenizer	Tokenization	Basque	B
CUNI	UDPipe tokenizer	Tokenization	Catalan	B
CUNI	UDPipe tokenizer	Tokenization	Galician	B

M22

Provider	Tool	Service	Lang. - Category	
CUNI	UDPipe tokenizer	Tokenization	Norwegian	B
CUNI	UDPipe tokenizer	Tokenization	Serbian	B
CUNI	UDPipe tokenizer	Tokenization	Turkish	B
CUNI	UDPipe tokenizer	Tokenization	Ukrainian	B
DFKI	Dependency Tree Parser for German Clinical Text	Parsing	German	A
DFKI	Excitement Open Platform	Textual Entailment	English	A
DFKI	Excitement Open Platform	Textual Entailment	German	A
DFKI	Excitement Open Platform	Textual Entailment	Italian	A
DFKI	JTok	Sentence splitting	Italian	A
DFKI	JTok	Tokenization	Italian	A
DFKI	Lynx-TIMEX	Date detection	English	A
DFKI	Lynx-TIMEX	Date detection	German	A
DFKI	Lynx-TIMEX	Time annotation	English	A
DFKI	Lynx-TIMEX	Time annotation	German	A
DFKI	mEx	Relation Extraction	German	A
DFKI	MMorph3	Morphological analyser	Dutch	A
DFKI	MMorph3	Morphological analyser	Italian	A
DFKI	Negation Detection	Negation Detection	German	A
DFKI	Qurator-LangIdent	Language identification	Bulgarian	A
DFKI	Qurator-LangIdent	Language identification	Croatian	A
DFKI	Qurator-LangIdent	Language identification	Danish	A
DFKI	Qurator-LangIdent	Language identification	Dutch	A
DFKI	Qurator-LangIdent	Language identification	Estonian	A
DFKI	Qurator-LangIdent	Language identification	Finnish	A
DFKI	Qurator-LangIdent	Language identification	Hungarian	A
DFKI	Qurator-LangIdent	Language identification	Italian	A
DFKI	Qurator-LangIdent	Language identification	Lithuanian	A
DFKI	Qurator-LangIdent	Language identification	Polish	A
DFKI	Qurator-LangIdent	Language identification	Portuguese	A
DFKI	Qurator-LangIdent	Language identification	Romanian	A
DFKI	Qurator-LangIdent	Language identification	Slovak	A
DFKI	Qurator-LangIdent	Language identification	Slovenian	A
DFKI	Qurator-LangIdent	Language identification	Swedish	A
DFKI	Qurator-LangIdent	Language identification	Albanian	B
DFKI	Qurator-LangIdent	Language identification	Catalan	B
DFKI	Qurator-LangIdent	Language identification	Norwegian	B
DFKI	Qurator-LangIdent	Language identification	Turkish	B
DFKI	Qurator-LangIdent	Language identification	Ukrainian	B
DFKI	Qurator-LangIdent	Language identification	Welsh	B
Expert System	Cogito Discover	Key phrase Extraction	Dutch	A
Expert System	Cogito Discover	Key phrase Extraction	English	A
Expert System	Cogito Discover	Key phrase Extraction	French	A
Expert System	Cogito Discover	Key phrase Extraction	German	A
Expert System	Cogito Discover	Key phrase Extraction	Italian	A
Expert System	Cogito Discover	Key phrase Extraction	Portuguese	A
Expert System	Cogito Discover	Key phrase Extraction	Spanish	A
Expert System	Cogito Discover	Language identification	Bulgarian	A
Expert System	Cogito Discover	Language identification	Croatian	A

M22

Provider	Tool	Service	Lang. - Category	
Expert System	Cogito Discover	Language identification	Danish	A
Expert System	Cogito Discover	Language identification	Dutch	A
Expert System	Cogito Discover	Language identification	Estonian	A
Expert System	Cogito Discover	Language identification	Finnish	A
Expert System	Cogito Discover	Language identification	Hungarian	A
Expert System	Cogito Discover	Language identification	Italian	A
Expert System	Cogito Discover	Language identification	Lithuanian	A
Expert System	Cogito Discover	Language identification	Polish	A
Expert System	Cogito Discover	Language identification	Portuguese	A
Expert System	Cogito Discover	Language identification	Romanian	A
Expert System	Cogito Discover	Language identification	Slovak	A
Expert System	Cogito Discover	Language identification	Slovenian	A
Expert System	Cogito Discover	Language identification	Swedish	A
Expert System	Cogito Discover	Language identification	Albanian	B
Expert System	Cogito Discover	Language identification	Norwegian	B
Expert System	Cogito Discover	Language identification	Turkish	B
Expert System	Cogito Discover	Language identification	Ukrainian	B
Expert System	Cogito Discover	Lemmatisation	Dutch	A
Expert System	Cogito Discover	Lemmatisation	Italian	A
Expert System	Cogito Discover	Lemmatisation	Portuguese	A
Expert System	Cogito Discover	Named Entity Recognition	Dutch	A
Expert System	Cogito Discover	Named Entity Recognition	Italian	A
Expert System	Cogito Discover	Named Entity Recognition	Portuguese	A
Expert System	Cogito Discover	Part of Speech tagging	Dutch	A
Expert System	Cogito Discover	Part of Speech tagging	Italian	A
Expert System	Cogito Discover	Part of Speech tagging	Portuguese	A
Expert System	Cogito Discover	Sentiment Analysis	Italian	A
Expert System	Cogito Discover	Summarization	Dutch	A
Expert System	Cogito Discover	Summarization	Italian	A
Expert System	Cogito Discover	Summarization	Portuguese	A
Expert System	Cogito Discover	Word sense disambiguation	Dutch	A
Expert System	Cogito Discover	Word sense disambiguation	English	A
Expert System	Cogito Discover	Word sense disambiguation	French	A
Expert System	Cogito Discover	Word sense disambiguation	German	A
Expert System	Cogito Discover	Word sense disambiguation	Italian	A
Expert System	Cogito Discover	Word sense disambiguation	Portuguese	A
Expert System	Cogito Discover	Word sense disambiguation	Spanish	A
SAIL LABS	SAIL KWS	Keyword extraction	Dutch	A
SAIL LABS	SAIL KWS	Keyword extraction	English	A
SAIL LABS	SAIL KWS	Keyword extraction	French	A
SAIL LABS	SAIL KWS	Keyword extraction	German	A
SAIL LABS	SAIL KWS	Keyword extraction	Greek	A
SAIL LABS	SAIL KWS	Keyword extraction	Italian	A
SAIL LABS	SAIL KWS	Keyword extraction	Polish	A
SAIL LABS	SAIL KWS	Keyword extraction	Romanian	A
SAIL LABS	SAIL KWS	Keyword extraction	Spanish	A
SAIL LABS	SAIL KWS	Keyword extraction	Albanian	B
SAIL LABS	SAIL KWS	Keyword extraction	Norwegian	B
SAIL LABS	SAIL KWS	Keyword extraction	Turkish	B

M22

Provider	Tool	Service	Lang. - Category	
SAIL LABS	SAIL language ID	Language identification	Bulgarian	A
SAIL LABS	SAIL language ID	Language identification	Dutch	A
SAIL LABS	SAIL language ID	Language identification	Hungarian	A
SAIL LABS	SAIL language ID	Language identification	Italian	A
SAIL LABS	SAIL language ID	Language identification	Polish	A
SAIL LABS	SAIL language ID	Language identification	Portuguese	A
SAIL LABS	SAIL language ID	Language identification	Romanian	A
SAIL LABS	SAIL language ID	Language identification	Slovak	A
SAIL LABS	SAIL language ID	Language identification	Swedish	A
SAIL LABS	SAIL language ID	Language identification	Albanian	B
SAIL LABS	SAIL language ID	Language identification	Norwegian	B
SAIL LABS	SAIL language ID	Language identification	Turkish	B
SAIL LABS	SAIL NER	Named Entity Recognition	Bulgarian	A
SAIL LABS	SAIL NER	Named Entity Recognition	Croatian	A
SAIL LABS	SAIL NER	Named Entity Recognition	Dutch	A
SAIL LABS	SAIL NER	Named Entity Recognition	Hungarian	A
SAIL LABS	SAIL NER	Named Entity Recognition	Italian	A
SAIL LABS	SAIL NER	Named Entity Recognition	Polish	A
SAIL LABS	SAIL NER	Named Entity Recognition	Portuguese	A
SAIL LABS	SAIL NER	Named Entity Recognition	Romanian	A
SAIL LABS	SAIL NER	Named Entity Recognition	Slovak	A
SAIL LABS	SAIL NER	Named Entity Recognition	Swedish	A
SAIL LABS	SAIL NER	Named Entity Recognition	Albanian	B
SAIL LABS	SAIL NER	Named Entity Recognition	Catalan	B
SAIL LABS	SAIL NER	Named Entity Recognition	Norwegian	B
SAIL LABS	SAIL NER	Named Entity Recognition	Turkish	B
SAIL LABS	SAIL polarity analysis	Polarity detection	English	A
SAIL LABS	SAIL polarity analysis	Polarity detection	French	A
SAIL LABS	SAIL polarity analysis	Polarity detection	German	A
SAIL LABS	SAIL polarity analysis	Polarity detection	Italian	A
SAIL LABS	SAIL polarity analysis	Polarity detection	Polish	A
SAIL LABS	SAIL polarity analysis	Polarity detection	Portuguese	A
SAIL LABS	SAIL polarity analysis	Polarity detection	Spanish	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Italian	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Polish	A
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Portuguese	A
USFD	DecarboNET Environmental Annotator	Entity linking	English	A
USFD	DecarboNET Environmental Annotator	Entity linking	German	A
USFD	GATE Cloud: Language ID for Tweets	Language identification	Dutch	A
USFD	GATE Cloud: Measurement Annotator	Measurement annotation	English	A
USFD	GATE Cloud: Measurement Annotator	Measurement normalisation	English	A
USFD	GATE Cloud: Measurement Annotator	Number normalisation	English	A
USFD	GATE Cloud: NP Chunker	Noun phrase extraction	English	A
USFD	GATE Cloud: OpenNLP Pipelines	Named Entity Recognition	Dutch	A
USFD	GATE Cloud: OpenNLP Pipelines	Part of Speech tagging	Dutch	A
USFD	GATE Cloud: OpenNLP Pipelines	Sentence splitting	Dutch	A
USFD	GATE Cloud: OpenNLP Pipelines	Tokenization	Dutch	A
USFD	GATE Cloud: Romanian NER	Named Entity Recognition	Romanian	A
USFD	GATE Cloud: Welsh NER	Named Entity Recognition	Welsh	B

		M22		
Provider	**Tool**	**Service**	**Lang. - Category**	
USFD	TermRaider	Text extraction	English	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Bulgarian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Croatian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Danish	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Dutch	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Estonian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Finnish	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Polish	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Portuguese	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Romanian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Slovak	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Slovenian	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Swedish	A
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Basque	B
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Catalan	B

Table 31. IE and Text Analysis tools and services to integrate in the second release (full list)

		M34		
Provider	**Tool**	**Service**	**Lang. - Category**	
CUNI	Entity Linker	Entity linking	Lang. independent	E
CUNI	UDPipe parser	Dependency Parsing	Afrikaans	C
CUNI	UDPipe parser	Dependency Parsing	Arabic	C
CUNI	UDPipe parser	Dependency Parsing	Chinese	C
CUNI	UDPipe parser	Dependency Parsing	Hebrew	C
CUNI	UDPipe parser	Dependency Parsing	Hindi/Urdu	C
CUNI	UDPipe parser	Dependency Parsing	Indonesian	C
CUNI	UDPipe parser	Dependency Parsing	Japanese	C
CUNI	UDPipe parser	Dependency Parsing	Korean	C
CUNI	UDPipe parser	Dependency Parsing	Latin	C
CUNI	UDPipe parser	Dependency Parsing	Persian	C
CUNI	UDPipe parser	Dependency Parsing	Russian	C
CUNI	UDPipe parser	Dependency Parsing	Tamil	C
CUNI	UDPipe parser	Dependency Parsing	Vietnamese	C
CUNI	UDPipe tagger	Lemmatisation	Afrikaans	C
CUNI	UDPipe tagger	Lemmatisation	Arabic	C
CUNI	UDPipe tagger	Lemmatisation	Chinese	C
CUNI	UDPipe tagger	Lemmatisation	Hebrew	C
CUNI	UDPipe tagger	Lemmatisation	Hindi/Urdu	C
CUNI	UDPipe tagger	Lemmatisation	Indonesian	C
CUNI	UDPipe tagger	Lemmatisation	Japanese	C
CUNI	UDPipe tagger	Lemmatisation	Latin	C
CUNI	UDPipe tagger	Lemmatisation	Persian	C
CUNI	UDPipe tagger	Lemmatisation	Russian	C
CUNI	UDPipe tagger	Lemmatisation	Tamil	C
CUNI	UDPipe tagger	Lemmatisation	Vietnamese	C
CUNI	UDPipe tagger	Morphological analyser	Afrikaans	C
CUNI	UDPipe tagger	Morphological analyser	Arabic	C
CUNI	UDPipe tagger	Morphological analyser	Chinese	C

Provider	Tool	M34 Service	Lang. - Category	
CUNI	UDPipe tagger	Morphological analyser	Hebrew	C
CUNI	UDPipe tagger	Morphological analyser	Hindi/Urdu	C
CUNI	UDPipe tagger	Morphological analyser	Indonesian	C
CUNI	UDPipe tagger	Morphological analyser	Japanese	C
CUNI	UDPipe tagger	Morphological analyser	Korean	C
CUNI	UDPipe tagger	Morphological analyser	Latin	C
CUNI	UDPipe tagger	Morphological analyser	Persian	C
CUNI	UDPipe tagger	Morphological analyser	Russian	C
CUNI	UDPipe tagger	Morphological analyser	Tamil	C
CUNI	UDPipe tagger	Morphological analyser	Vietnamese	C
CUNI	UDPipe tagger	Part of Speech tagging	Afrikaans	C
CUNI	UDPipe tagger	Part of Speech tagging	Arabic	C
CUNI	UDPipe tagger	Part of Speech tagging	Chinese	C
CUNI	UDPipe tagger	Part of Speech tagging	Hebrew	C
CUNI	UDPipe tagger	Part of Speech tagging	Hindi/Urdu	C
CUNI	UDPipe tagger	Part of Speech tagging	Indonesian	C
CUNI	UDPipe tagger	Part of Speech tagging	Japanese	C
CUNI	UDPipe tagger	Part of Speech tagging	Korean	C
CUNI	UDPipe tagger	Part of Speech tagging	Latin	C
CUNI	UDPipe tagger	Part of Speech tagging	Persian	C
CUNI	UDPipe tagger	Part of Speech tagging	Russian	C
CUNI	UDPipe tagger	Part of Speech tagging	Tamil	C
CUNI	UDPipe tagger	Part of Speech tagging	Vietnamese	C
CUNI	UDPipe tokenizer	Tokenization	Afrikaans	C
CUNI	UDPipe tokenizer	Tokenization	Arabic	C
CUNI	UDPipe tokenizer	Tokenization	Chinese	C
CUNI	UDPipe tokenizer	Tokenization	Hebrew	C
CUNI	UDPipe tokenizer	Tokenization	Hindi/Urdu	C
CUNI	UDPipe tokenizer	Tokenization	Indonesian	C
CUNI	UDPipe tokenizer	Tokenization	Japanese	C
CUNI	UDPipe tokenizer	Tokenization	Korean	C
CUNI	UDPipe tokenizer	Tokenization	Latin	C
CUNI	UDPipe tokenizer	Tokenization	Persian	C
CUNI	UDPipe tokenizer	Tokenization	Russian	C
CUNI	UDPipe tokenizer	Tokenization	Tamil	C
CUNI	UDPipe tokenizer	Tokenization	Vietnamese	C
DFKI	MDparser	Dependency Parsing	Lang. independent	E
DFKI	MDparser	Part of Speech tagging	Lang. independent	E
DFKI	MDparser	Sentence splitting	Lang. independent	E
DFKI	MDparser	Tokenization	Lang. independent	E
DFKI	Qurator-LangIdent	Language identification	Afrikaans	C
DFKI	Qurator-LangIdent	Language identification	Arabic	C
DFKI	Qurator-LangIdent	Language identification	Chinese	C
DFKI	Qurator-LangIdent	Language identification	Hebrew	C
DFKI	Qurator-LangIdent	Language identification	Hindi/Urdu	C
DFKI	Qurator-LangIdent	Language identification	Indonesian	C
DFKI	Qurator-LangIdent	Language identification	Japanese	C
DFKI	Qurator-LangIdent	Language identification	Korean	C
DFKI	Qurator-LangIdent	Language identification	Malay	C
DFKI	Qurator-LangIdent	Language identification	Persian	C
DFKI	Qurator-LangIdent	Language identification	Russian	C
DFKI	Qurator-LangIdent	Language identification	Tamil	C
DFKI	Qurator-LangIdent	Language identification	Vietnamese	C

Provider	Tool	M34 Service	Lang. - Category	
DFKI	Qurator-LangIdent	Language identification	Bengali	D
DFKI	Qurator-LangIdent	Language identification	Gujarati	D
DFKI	Qurator-LangIdent	Language identification	Kannada	D
DFKI	Qurator-LangIdent	Language identification	Macedonian	D
DFKI	Qurator-LangIdent	Language identification	Marahati	D
DFKI	Qurator-LangIdent	Language identification	Nepali	D
DFKI	Qurator-LangIdent	Language identification	Panjabi	D
DFKI	Qurator-LangIdent	Language identification	Somali	D
DFKI	Qurator-LangIdent	Language identification	Swahili	D
DFKI	Qurator-LangIdent	Language identification	Tagalog	D
DFKI	Qurator-LangIdent	Language identification	Telugu	D
DFKI	Qurator-LangIdent	Language identification	Thai	D
DFKI	Qurator-LangIdent	Language identification	Urdu	D
Expert System	Cogito Discover	Key phrase Extraction	Arabic	C
Expert System	Cogito Discover	Key phrase Extraction	Chinese	C
Expert System	Cogito Discover	Key phrase Extraction	Japanese	C
Expert System	Cogito Discover	Key phrase Extraction	Korean	C
Expert System	Cogito Discover	Key phrase Extraction	Russian	C
Expert System	Cogito Discover	Language identification	Afrikaans	C
Expert System	Cogito Discover	Language identification	Arabic	C
Expert System	Cogito Discover	Language identification	Chinese	C
Expert System	Cogito Discover	Language identification	Hebrew	C
Expert System	Cogito Discover	Language identification	Hindi/Urdu	C
Expert System	Cogito Discover	Language identification	Indonesian	C
Expert System	Cogito Discover	Language identification	Japanese	C
Expert System	Cogito Discover	Language identification	Korean	C
Expert System	Cogito Discover	Language identification	Malay	C
Expert System	Cogito Discover	Language identification	Persian	C
Expert System	Cogito Discover	Language identification	Russian	C
Expert System	Cogito Discover	Language identification	Tamil	C
Expert System	Cogito Discover	Language identification	Vietnamese	C
Expert System	Cogito Discover	Language identification	Bengali	D
Expert System	Cogito Discover	Language identification	Gujarati	D
Expert System	Cogito Discover	Language identification	Kannada	D
Expert System	Cogito Discover	Language identification	Macedonian	D
Expert System	Cogito Discover	Language identification	Marahati	D
Expert System	Cogito Discover	Language identification	Nepali	D
Expert System	Cogito Discover	Language identification	Panjabi	D
Expert System	Cogito Discover	Language identification	Somali	D
Expert System	Cogito Discover	Language identification	Swahili	D
Expert System	Cogito Discover	Language identification	Tagalog	D
Expert System	Cogito Discover	Language identification	Telugu	D
Expert System	Cogito Discover	Language identification	Thai	D
Expert System	Cogito Discover	Language identification	Urdu	D
Expert System	Cogito Discover	Lemmatisation	Arabic	C
Expert System	Cogito Discover	Lemmatisation	Chinese	C
Expert System	Cogito Discover	Lemmatisation	Japanese	C
Expert System	Cogito Discover	Lemmatisation	Korean	C
Expert System	Cogito Discover	Lemmatisation	Russian	C
Expert System	Cogito Discover	Named Entity Recognition	Arabic	C
Expert System	Cogito Discover	Named Entity Recognition	Chinese	C
Expert System	Cogito Discover	Named Entity Recognition	Japanese	C
Expert System	Cogito Discover	Named Entity Recognition	Korean	C

Provider	Tool	M34 Service	Lang. - Category	
Expert System	Cogito Discover	Named Entity Recognition	Russian	C
Expert System	Cogito Discover	Part of Speech tagging	Arabic	C
Expert System	Cogito Discover	Part of Speech tagging	Chinese	C
Expert System	Cogito Discover	Part of Speech tagging	Japanese	C
Expert System	Cogito Discover	Part of Speech tagging	Korean	C
Expert System	Cogito Discover	Part of Speech tagging	Russian	C
Expert System	Cogito Discover	Summarization	Arabic	C
Expert System	Cogito Discover	Summarization	Chinese	C
Expert System	Cogito Discover	Summarization	Japanese	C
Expert System	Cogito Discover	Summarization	Korean	C
Expert System	Cogito Discover	Summarization	Russian	C
Expert System	Cogito Discover	Text extraction	Lang. independent	E
Expert System	Cogito Discover	Word sense disambiguation	Arabic	C
Expert System	Cogito Discover	Word sense disambiguation	Chinese	C
Expert System	Cogito Discover	Word sense disambiguation	Japanese	C
Expert System	Cogito Discover	Word sense disambiguation	Korean	C
Expert System	Cogito Discover	Word sense disambiguation	Russian	C
SAIL LABS	SAIL KWS	Keyword extraction	Arabic	C
SAIL LABS	SAIL KWS	Keyword extraction	Chinese	C
SAIL LABS	SAIL KWS	Keyword extraction	Hebrew	C
SAIL LABS	SAIL KWS	Keyword extraction	Hindi/Urdu	C
SAIL LABS	SAIL KWS	Keyword extraction	Indonesian	C
SAIL LABS	SAIL KWS	Keyword extraction	Malay	C
SAIL LABS	SAIL KWS	Keyword extraction	Pashto	C
SAIL LABS	SAIL KWS	Keyword extraction	Persian	C
SAIL LABS	SAIL KWS	Keyword extraction	Russian	C
SAIL LABS	SAIL language ID	Language identification	Arabic	C
SAIL LABS	SAIL language ID	Language identification	Hebrew	C
SAIL LABS	SAIL language ID	Language identification	Hindi/Urdu	C
SAIL LABS	SAIL language ID	Language identification	Indonesian	C
SAIL LABS	SAIL language ID	Language identification	Malay	C
SAIL LABS	SAIL language ID	Language identification	Pashto	C
SAIL LABS	SAIL language ID	Language identification	Persian	C
SAIL LABS	SAIL language ID	Language identification	Russian	C
SAIL LABS	SAIL language ID	Language identification	Lang. independent	E
SAIL LABS	SAIL NER	Named Entity Recognition	Arabic	C
SAIL LABS	SAIL NER	Named Entity Recognition	Chinese	C
SAIL LABS	SAIL NER	Named Entity Recognition	Hebrew	C
SAIL LABS	SAIL NER	Named Entity Recognition	Hindi/Urdu	C
SAIL LABS	SAIL NER	Named Entity Recognition	Indonesian	C
SAIL LABS	SAIL NER	Named Entity Recognition	Malay	C
SAIL LABS	SAIL NER	Named Entity Recognition	Pashto	C
SAIL LABS	SAIL NER	Named Entity Recognition	Persian	C
SAIL LABS	SAIL NER	Named Entity Recognition	Russian	C
SAIL LABS	SAIL polarity analysis	Polarity detection	Arabic	C
SAIL LABS	SAIL polarity analysis	Polarity detection	Indonesian	C
SAIL LABS	SAIL polarity analysis	Polarity detection	Malay	C
SAIL LABS	SAIL polarity analysis	Polarity detection	Russian	C
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Arabic	C
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Indonesian	C
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Malay	C
SAIL LABS	SAIL polarity analysis	Sentiment Analysis	Russian	C
SAIL LABS	SAIL summarization	Summarization	Lang. independent	E

Provider	Tool	M34 Service	Lang. - Category	
USFD	GATE Cloud: Russian NER	Named Entity Recognition	Russian	C
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Indonesian	C
USFD	Universal Dependencies POS Tagger	Part of Speech tagging	Russian	C

Table 32. IE and Text Analysis tools and services to integrate in the third release (full list)