elexis european lexicographic infrastructure

D1.3

Tools for the automatic segmentation and identification of lexicographic content

Author(s): Andraž Repar, Simon Krek

Date: January 31, 2020

H2020-INFRAIA-2016-2017

Grant Agreement No. 731015

ELEXIS - European Lexicographic Infrastructure

D1.3 Tools for the automatic segmentation and identification of lexicographic content

Deliverable Number:	D1.3
Dissemination Level:	Public
Delivery Date:	January 31, 2019
Version:	1
Author(s):	Andraž Repar, Simon Krek

Project Acronym:	ELEXIS
Project Full Title:	European Lexicographic Infrastructure
Grant Agreement No.:	731015

Deliverable/Document Information

Project Acronym:	ELEXIS
Project Full Title:	European Lexicographic Infrastructure
Grant Agreement No.:	731015

Document History

Version Date	Changes/Approval	Author(s)/Approved by
January 20, 2020	Initial draft	Andraž Repar
January 20, 2020	Internal revision	Simon Krek
January 20, 2020	External evaluation	Miloš Jakubíček

Table of Contents

1 About this report

This report describes the tools for the automatic segmentation and identification of lexicographic content, developed as part of the LEX1 infrastructure of ELEXIS.

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

2 Elexifier

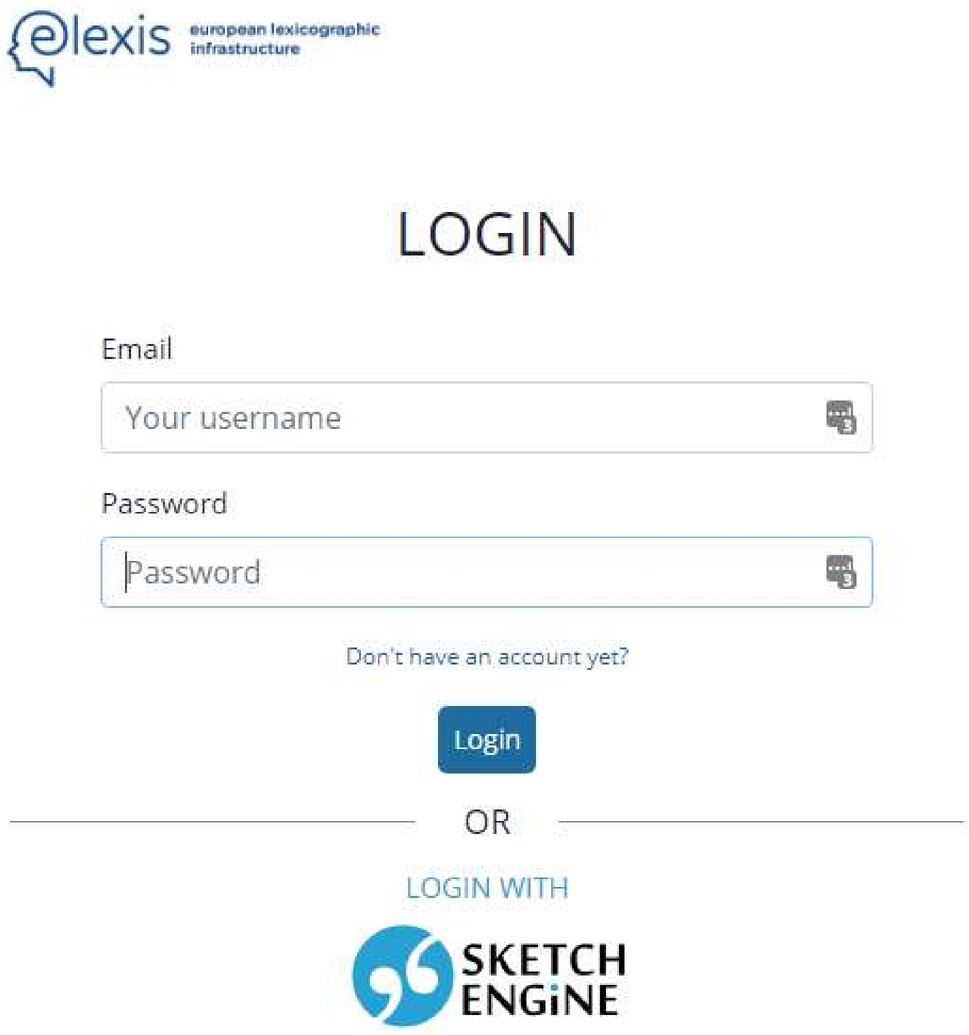
elexis european lexicographic infrastructure
LOGIN
Email
Your username
Password
Password
Don't have an account yet?
Login
OR
LOGIN WITH
SKETCH ENGINE

Figure 1: Elexifier login screen.

Elexifier (elexifier.elex.is) is a cloud-based dictionary conversion service for conversion of legacy XML and PDF dictionaries into a shared data format based on the Elexis Data Model (defined in deliverable D1.2). It takes as input an XML or PDF dictionary and produces a TEI-compliant XML file in line with the specifications described in the Elexis Data Model.

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

2.1 Infrastructure

The application consists of two Docker containers:

- Frontend: https://github.com/elexis-eu/elexifier
- Backend: https://github.com/elexis-eu/elexifier-api

The frontend is written in Angular. The backend is written in Python Flask and uses a Postgres database. For local installation, see the instructions in the Github repository.

2.2 Use

On the login screen, create a new account or login with your Sketch Engine credentials. Then select the XML or PDF module and upload a dictionary to get started. For detailed instructions, check the User Guide.

2.3 XML transformation - basic concepts

To transform a custom XML dictionary into an Elexis Data Model compliant format, you need to define a transformation, which specifies rules for transforming custom XML elements into Elexis Data model core elements. The script *https://github.com/elexis-eu/elexifier-api/blob/master/app/transformator/dictTransformations3.py* takes as input a JSON object with the following members:

- ***entry*** — describes the selector for entry elements
- ***entry_lang*** — describes the transformer for the language attribute of the entries
- ***sense*** — describes the selector for sense elements
- ***hw*** — describes the transformer for headwords
- ***sec_hw*** — describes the transformer for secondary headwords
- ***pos*** — describes the transformer for part-of-speech tags
- ***hw_tr*** — describes the transformer for translations of headwords
- ***hw_tr_lang*** — describes the transformer for the language of the translations of headwords
- ***ex*** — describes the transformer for examples
- ***ex_tr*** — describes the transformer for translations of examples
- ***ex_tr_lang*** — describes the transformer for the language of the translations of examples

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

- ***def*** — describes the transformer for definitions

2.3.1 Selector descriptions

A selector is a rule that selects 0 or more elements in the input XML tree.

The description of a selector must be a JSON object. This object must contain an attribute named type, whose value specifies the type the selector, plus one or more other attributes whose name and meaning depends on the selector type.

The following types of selectors are currently supported:

Xpath selector: selects the nodes that match a given xpath expression (given in an attribute named expr).

Example:{"type": "xpath", "expr": ".//example/text"}

Union selector: combines the results of several selectors (whose descriptions must be given as a JSON array in an attribute named selectors).

Example:{"type": "union", "selectors": [...]}

Exclude selector: takes two selectors, left and right, and selects all those nodes which were selected by left but not by right.

Example:{"type": "exclude", "left": {...}, "right": {...}}

2.3.2 Transformer descriptions

A transformer is a rule that describes which data from the input document must be transformed into a certain type of element in the output document.

The description of a transformer must be a JSON object. This object must contain an attribute named type, whose value specifies the type the transformer, plus one or more other attributes whose name and meaning depends on the transformer type.

The following types of transformers are currently supported:

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

(1) Simple transformers

A simple transformer selects a set of elements and extracts an attribute or the inner text from these elements; optionally applies a regular expression to the resulting text and returns the substring matched by a specific group within the regular expression.

The JSON object that describes a simple transformer must contain the following attributes:

type: this must be the string "simple".

selector: a JSON object describing a selector.

attr: the name of an attribute (from the elements selected by the selector) whose value is to be extracted.To extract the inner text of the element, instead of an attribute, use the pseudo-attribute name "{http://elex.is/wp1/teiLex0Mapper/meta}innerText".To extract the inner text of the element and all of its descendants, use "{http://elex.is/wp1/teiLex0Mapper/meta}innerTextRec".To return a constant value instead of extracting the value of an attribute, use the pseudo-attribute name "{http://elex.is/wp1/teiLex0Mapper/meta}constant".

rex: a regular expression that is applied to the value of the attribute attr. If this string does not contain any match for this regular expression, the current element is not transformed (i.e. it is as if it hadn't been selected by the selector at all). If there are several matches, the first one is used. This attribute is optional. If present, it must use the Python regular expression syntax.

rexGroup: this attribute is optional. If present, it must be the name of one of the named groups (?P<name>...) from the regular expression given by the attribute rex. In this case, only the string that matched this named group will be used, rather than the entire value of the attribute attr.

const: this attribute should be present it attr was set to "{http://elex.is/wp1/teiLex0Mapper/meta}constant", and should provide the constant value that you want to return as the result of the transformation.

xlat: this attribute is optional. If present, it should be a hash table that will be used to transform the string obtained from the previous steps (attribute lookup, regex matching). In other words, the string s will be replaced by xlat[s] if s appears as a key in xlat (otherwise, s will remain unchanged, just as if xlat had not been provided at all).

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

A simple example:

```
{ "type": "simple",

"selector": {"type": "xpath", "expr": ".//ExampleCtn//Locale"},

"attr": "lang" }
```

A more complex example:

```
{ "type": "simple",

"selector": {"type": "xpath", "expr": ".//sense/seg[1][@type='beleg']"},

"attr": "{http://elex.is/wp1/teiLex0Mapper/meta}innerTextRec"

"rex": "'(?P<insideQuotes>[^']*)'",

"rexGroup: "insideQuotes" }
```

This transformer selects the first <seg> in each <sense>, builds the inner text and extracts the first substring delimited by single quote marks.

An example of a constant-output transformer (i.e. to assign language codes to XML elements):

```
{ "type": "simple",

  "selector": {"type: "xpath", "expr": ".//type"},

  "attr": "{http://elex.is/wp1/teiLex0Mapper/meta}constant",

  "const": "en" }
```

(2) Union transformers

A union transformer takes a set of simple transformers and performs all of their transformations. This might be useful if you need to combine several different transformation rules, e.g. extract attribute @a from instances of the element <b> and also extract attribute @c from instances of the element <d>.

The JSON object that describes a union transformer must contain the following attributes:

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

type: this must be the string "union".

transformers: an array of JSON objects describing the transformers that are to be combined.

2.4 PDF transformation - basic concepts

To transform a PDF dictionary, you need to annotate a sample of the PDF file. The PDF is first transformed in flat structure using a pdf2xml conversion script (based on https://github.com/kermitt2/pdf2xml). Then, a chunk of the resulting XML file is sent to Lexonomy for manual annotation. In the next step, the annotations act as training data for the machine learning algorithm. The following features are used by the algorithm: font, font-size, bold, italic, newline and the token content itself.

Machine learning assumes a three-level structure with pages as first level base, entries as second level base and senses as third level base. On the first level, entries are predicted for the second level to work on, which in turn generates third level base - senses. A model is constructed for each level and trained on 75% of the data annotated in lexonomy. Afterwards, labels for each token (separate word or symbol in the dictionary) of the unlabelled data are predicted for each level. At first, unlabelled data is only available for the first level, but through prediction, second and third level data is generated as well, along with the labels. Labels are then used to wrap tokens into correct containers.

The model used is a recurrent neural network with two inputs for each input token: one-hot encoded token features (such as font, size and so forth) and LSTM-encoded token contents. The two inputs are merged and fed into a bidirectional LSTM, which then outputs a one-hot encoded label. Labels are defined in the annotation and the model can adapt to different labels at different levels, depending on the annotation structure.

Current results show great promise as they often exceed 90% f1 score (varies between levels and datasets) and are achieved within a short training time. However, results are, as always in the field of machine learning, significantly influenced by the quality of the annotation

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

3 User feedback

During development, we collected three batches of user feedback from a select group of project partners: August 2019, November 2019 and January 2020. Several of the ideas presented by the users were implemented in the application and we will continue to monitor user behaviour for possible improvements.

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

4 Expected impact

As part of the LEX1 infrastructure, Elexifier will play an important role in the future of the ELEXIS project, specifically in terms of feeding data to the Matrix Dictionary. It will allow lexicographers with limited computer programming skills to efficienctly convert their legacy dictionaries into a standardized common format and upload them to the Matrix Dictionary. The official launch of the application is scheduled for the Project Management Board meeting in March in Ljubljana, Slovenia.

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.