Corpus tools

This project contains tools for extracting corpora from a limited number of datasets and converting them into a common XML format. The corpora contain links between documents or sections of the documents and can be used for training and validating models that generate semantic links between (parts of) documents.

A separate tool (written in Python) is available for each dataset. The tools use common libraries for reading the data and generating the corpora.

- `GWikimatchcorpus.py` converts data from the gWikiMatch dataset .
- `WikiDataCorpus.py` creates a corpus using WikiData and Wikipedia .
- `S2ORCCorpus.py` for obtained from the S2ORC dataset .

Data

The scripts combine input from various sources into a corpus. The combination of the sources per tool is depicted in the diagram below:

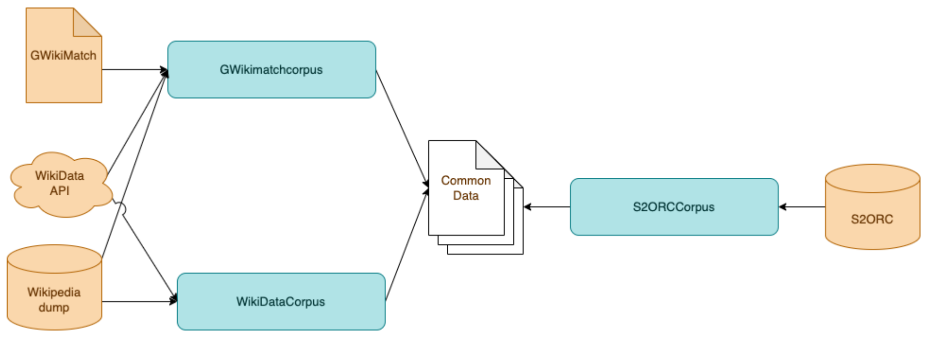

Figure 1: Architecture of CorpusCommand

GWikiMatch

GWikiMatch is a benchmark dataset for long-form document matching in the form of a TSV file where each record contains a source and destination URL from the English Wikipedia along with a similarty-label (0=not similar, 1=somewhat similar, and 2=strongly similar. The Wikipedia URL is translated to a WikiData ID, so it can easily be used in combination with data obtained from Wikidata.

WikiData API

The tools communicate with WikiData using the Query interface, a SPARQL endpoint that gives full access to the WikiData knowledge graph. The query's runtime on the public endpoint is limited to 60 seconds and a maximum number of queries per minute. For this reason, queries are cached on the filesystem to prevent the tool from accessing Wikidata unnecessarily. All Wikidata entities have an associated id that can be used to obtain Wikipedia articles in multiple languages.

Wikipedia dump

The tools do not access the Wikipedia API but instead use dumps of Wikipedia because many Wikipedia articles are required, and the Wikipedia API has restrictions to avoid overuse. Wikipedia dumps can be obtained from Wikimedia downloads, which are available in multiple languages.

The tools expect two files per language, a dump of all articles in a specific language on a specific date `*pages-articles-multistream.xml.bz2` and an index of all articles of this dump `*pages-articles-multistream.xml.bz2`. These files are being read directly from the `.bz2` files, so they do not need to be extracted beforehand. To extract text and sections from the articles, the Python library WikiTextParser is used.

Common file format

The selected data is written to files in a common format so they can be processed independently of the source of the data. The files are XML files, of which an artificial example is shown below:

```
<doc id="00009">
    <title>Title of the document</title>
    <links>
        <link id="09987" index="0.566" class="1" />
        <link id="33322" index="0.654" class="1"/>
        <link id="12223" index="0.777" class="1"/>
    </links>
    <section id="00009" subid="1">
        <title>Title of the section</title>
        <links>
            <link id="09987" index="0.796" class="1" />
        </links>
        <text>Text of the section</text>
    </section>
    <section id="00009" subid="2">
        <title>Title of the section</title>
        <links />
        <text>Text of the section</text>
    </section>
</doc>
```

Details

The sections below give a short description of the usage and details of each of the tools.

GWikiMatchCorpus.py

Uses the GWikiMatch file to generate pairs of Wikipedia articles with a label. The filenames of the output files contain the WikiDataID of the article. The links between sections of Wikipedia articles are not resolved and are, therefore, always empty. The variables `wikidata_enpoint`, `wikipedia_dumpdir`, and `gwikimatch_dir` must be changed in the script. The other variables are specified via the command line:

```
usage: GWikiMatchCorpus.py [-h] -l LANGUAGE -o OUTPUT

Read articles from Wikipedia based on the gWikiDataset.

Arguments:
  -h, --help            show this help message and exit
  -l LANGUAGE, --language LANGUAGE
                        Language code, for example "nl" or "en"
  -o OUTPUT,   --output OUTPUT
                        Output directory
```

WikiDataCorpus

The WikiDataCorpus tool generates a corpus about one or more subjects. It uses the WikiData knowledge graph to determine which Wikipedia articles contain information about the subject, which is why the subject is a list of WikiData ids. The tool generates semantic links by comparing the outgoing links for each article or section.

When a pair of articles is available in the GWikiMatch dataset, a line is written to the file `gwikimatch.tsv` in the output directory. Although this file usually will not contain many records, it can be used as a supplement to validate a model trained on this corpus. In addition, statistics about the corpus are available in the file `stats.txt` in the output directory. The variables `wikidata_enpoint`, `wikipedia_dumpdir`, and `gwikimatch_dir` must be changed in the script. The other variables are specified via the command line:

```
usage: WikiDataCorpus.py [-h] -s SUBJECTS -l LANGUAGE [-o OUTPUT]

Read articles from Wikipedia based on the WikiData knowledge graph.

Arguments:
  -h, --help            show this help message and exit
  -s SUBJECTS, --subjects SUBJECTS
                        Main wikidata subjects, a comma seperated
                        list of WikiData ids (for example "wd:Q7397")
  -l LANGUAGE, --language LANGUAGE
                        Language code, for example "nl" or "en"
  -o OUTPUT,   --output OUTPUT
                        Output directory
```

S2ORCCorpus

The S2ORCCorpus generates a corpus in the common file format from the S2ORC dataset. The tool assumes that (part of) the S2ORC has been downloaded and extracted. The S2ORC dataset comes with a script to extract its data, and S2ORCCorpus uses the directory structure created by this script.

```
usage: S2ORCCorpus.py [-h] -s SUBJECT -p PDFPARSES -m METADATA -o OUTPUT
```

```
Read articles from Wikipedia based on the gWikiDataset.

Arguments:
  -h, --help            show this help message and exit
  -s SUBJECT,   --subject SUBJECT
                        Subject, for example "Computer Science"
  -p PDFPARSES, --pdfparses PDFPARSES
                        Directory with pdf_parses
  -m METADATA,  --metadata METADATA
                        Directory with meta_data
  -o OUTPUT,    --output OUTPUT
                        Output directory
```