# Practical Assignments
# Parallel Corpora (Corpus Linguistics with R)

Maria Kunilovskaya

Wolverhampton, April 01, 2020

"The essence of the corpus as against the text is that you do not observe it directly; instead you use tools of indirect observation, like query languages, concordancers, collocators, parsers, and aligners..." (Sinclair 2004: 189)

## 1 Minimal requirements and recommendations for Windows

- Port of the most important GNU shell utilities to Windows 64 bit (ls, word count (wc), head/tail, grep/egrep, sed, tree, zcat, gzip/unzip) UnxUtils (for Section 2)
- install python libraries requests, justext (for Section 3.1)
- install UDpipe (for annotation in Section 3.2)
  - test instalation:

    ```
    echo 'We are going to see the parsed output for this sentence.' | /your
        /path/to/udpipe/binary file --tokenize --tag --parse /your/path/to/
        udpipe/model-for-your-language
    ```

    For example on my machine:

    ```
    echo 'Colorless green ideas sleep furiously.' | /home/u2/tools/udpipe/
        udpipe-1.2.0-bin/bin-linux64/udpipe --tokenize --tag --parse /home/
        u2/tools/udpipe/udpipe-ud-2.3-181115/english-gum-ud-2.3-181115.
        udpipe
    ```

- install perl (necessary for LF Aligner in Section 3.3) and download LF Aligner; no installation required
- install (native Windows!) Notepad++
- Clone or download the GitHub repository parcorp with materials and scripts for this session

# 2 Handling available resources

NB! where possible use the data from the latest release by Workshop on Machine Translation (WMT)

## 2.1 Collect the plain texts for your language pair from the UN corpus and estimate the data size

1. download all parts of the XML archives for each language
2. re-assemble the archive:

   ```
   $ cat UNv1.0-TEI.en.tar.gz.* > UNv1.0.tei_en.tar.gz
   ```

3. extract the trees of folders:

   ```
   $ tar -xzf UNv1.0.tei_en.tar.gz
   ```

4. adjust language indices and run the script:

   ```
   $ python3 extract_SL-TL_UNcorp.py /path/to/extracted/SL/
   ```

5. count TL originals (for TL reference corpus):

   ```
   $ grep -r --include "*.xml" 'Original: RUSSIAN' | wc -l
   ```

6. adjust languages and extract reference texts

   ```
   $ python3 extract_TLsources_UNcorp.py /path/to/TL/folder/
   ```

7. manually inspect the texts in the resulting folders and decide whether you want any filtering or downsampling
8. look into filtering and size-reducing (sample-size normalisation) options in `preprocess/UNcorpus/filter_UNcorp.py` and `preprocess/UNcorpus/reduceUN_to_2010-2014.py`
9. use `wc_walks_rawfolders.py` to estimate corpus parameters
10. to get sentence counts tokenise sentences with NLTK (`build_your_own/en_tokenise_sentences.py`) and count lines in all files (`$ wc -l /path/to/folder/*.txt`)

NB! Note: even if a text is translated RU>EN, it does not mean that it is a Russian original!
Proof:

- from add_1.lnk:
  <linkGrp fromDoc="Xml/ru/1990/trans/wp_29/1999..." toDoc="Xml/en/1990/trans/wp_29/1999/14/add_1.xml" score="0.434254»
- $ grep Original add_1.xml:
  <s id="11:1" lang="ru" >Original: FRENCH</s>

### 2.2 News Commentary corpus: inspect single-member archives in terminal

download tsv (up to 50M)
download one-doc-per-line monolingual txt (around 40M)

- look inside gz archives without extracting them:
  `$ zcat news-commentary-v15.en-ru.tsv.gz | head -100`
- produce sentence-pair counts:
  `$ zcat news-commentary-v15.en-ru.tsv.gz | wc -l`
- produce document counts:
  `$ zcat news-commentary-v15.ru.gz | egrep '^$' | wc -l`

### 2.3 ParaCrawl: inspect multiple-member archives (tar.gz, tgz) in terminal

download ParaCrawl version (5.1) from WMT page (en-ru: around 668MB)

- List the contents of a tar.gz file
  `$ tar -tvf paracrawl-release1.en-ru.zipporah0-dedup-clean.tgz`
- Extract a single file from a tar.gz file (1.6 GB!)

  ```
  $tar -zxvf paracrawl-release1.en-ru.zipporah0-dedup-clean.tgz paracrawl-
      release1.en-ru.zipporah0-dedup-clean.ru | head
  ```

- Look inside a file inside a multi-member archive without decompressing it to a file:

  ```
  $ tar -axf paracrawl-release1.en-ru.zipporah0-dedup-clean.tgz paracrawl-
      release1.en-ru.zipporah0-dedup-clean.ru -O | head -50
  ```

### 2.4 Further pre-processing tasks before annotation for re-using downloads

- filtering: get rid of noise (short sentences, faulty line breaks, encoding errors)
- normalisation: split the data into chunks (same-size samples, if necessary)
- standardisation: unify spelling, inc. punctuation (e.g. «'' to ")

## 3 Building from scratch

### 3.1 Scraping the websites

- collect texts from a list of urls with `build_your_own/extract-text-from-links.py` (requires `requests, justext` libraries)

  - go to `build_your_own/`
  - run `python3 extract-text-from-links.py urls/en.links`
  - adjust the language in the last line of the script and extract translations for these texts from urls/ru.links and urls/es.links

- (advanced) follow the 7-step instructions in see scrape repository and produce a collection of parallel texts from a website (with multilingual BBC as example)

Proceed to cleaning the data as described in Section 2.4

### 3.2 Annotation with UDpipe

(a) Test the suggested pipeline (`raw2txt2conllu2lempos.py`) to produce versions of the corpus from your clean data

- one-sent-per-line punctuation-tokenized txt
- a conllu-format corpus and
- a lempos-represented corpus

(b) The script expects:

- a path to files to be pre-processed (assumingly, a subcorpus name: e.g. en, ru)
- a path where to store the output; you don't have to create it, just say where to create it
- the UD models for en, ru in the working folder (from which this script is run)
- the `preprocess_imports.py` module with the functions to be imported

(c) Adjust settings in the `preprocess_imports.py`: (full list in the order of appearance in the above support modules)

- sibstitute 58,000 with 58000 for it to be treated as ONE number, not TWO with a comma
- take care of English contracted forms
- properly separate punctuation with a space
- get rid of fragmets of xml code
- replace 2019 with xxxx
- filter out all weird symbols (inc. encoding errors, except allowed set)
- filter out one symbol tokens that are not recognized as a valid PoS

For the lempos output (such as: `city_NOUN that_PRON be_AUX cheaper_ADJ than_ADP ever_ADV ._PUNCT`):

- functionality to filter out listed items (nofunc=None, nopunct=None, noshort=True, stopwords=None)
- skip short sentences
- retain sentence breaks in the lempos output
- tag named entities based on case agreement and liniarity; mostly PROPN (`Chris::Norton_PROPN`)

(d) How to run the script:

- go to `parsing` folder
- for demonstration purposes run it on English texts in `cleandata/moc_data/media/source/en/`

```
python3 raw2txt2conllu2lempos.py --raw cleandata/mock_data/media/
    source/en/ --outto en_media_parsed/ --lang en
```

Advanced: apply the multilingual pipeline to your prepared parallel corpus (see `parsing/raw_multi-ling-tree2txt2conllu2lempos.py`)

### 3.3 Alignment

(a) To get a standard bitext format (why do you want it?) from several pairs of source texts and their translation with LF Aligner:

- adjust languages in LF_aligner_setup.txt
- create shell script that calls perl for every text pair
  open and edit build_your_own/aligner_demo.sh in Notepad++
- concatenate all TMXs
  `$ cat /path/to/align20200401/*.tmx > my_bitext.tmx`
- open `my_bitext.tmx` in Notepad++ and delete TMX headers, except the first one
- open the resultin TMX in Heartsome TMX Editor to manually correct it

(b) To correct the alignment manually use Heartsome TMX Editor8:

- download from Developer's github
  (use All installer link; select the correct version!)
- extract the archive
- make Heartsome TMX Editor file executable (Properties > Permissions > Allow executing file as a program)
- setup the correct Java (it did not run on the current Java 11.0.6, I fell back for java 1.8 (>= 1.6 is required)
- Video tutorial

### 3.4 Building a TreeTagger tagged corpus with Sketch Engine

- prepare monolingual folders or texts or corrected TMX
- login -> Institutional login
- go to `New corpus`, name the corpus, select its type and upload your files to the corpus builder
- in `Expert` settings select the tagset (for English they recomment modified TreeTagger tagset); press `Compile`
- to download the tagged result (and lose document separation!) go to `Manage corpus` and select `Vertical`
- produce a lemmatised or lempos corpus by pulling out the contents of respective columns (to search your corpus in AntConc or otherwise work with it outside Sketch Engine)
- understand their tagset for further analysis!

## 4 Information extraction and analysis

I am sure we will not have time to get here :-)

### 4.1 Their tools

### 4.2 Your own tools