# Common Crawl – Language Classification Howto

The aim is to create a linguistic repository by categorizing the common crawl data to appropriate languages, focusing on rare languages. For that we used Wikipedia data dumps to train the classifier.

## Getting the code

The repository contains wikiextractor (used for extracting wikipedia data) as a submodule, so the simplest way to get the right version of wikiextractor is to run `git submodule update --init --recursive` once the base repository has been cloned.

## Wikipedia Data pull

We used the xml dumps that can be accessed at

`https://dumps.wikimedia.org/<lang>wiki/latest/`

(where *<lang>* is replaced by the code for the language of interest). The dump of most interest is

`<lang>wiki-latest-pages-articles-multistream.xml.bz2`,

which contains the text of all pages in a single (compressed) XML file. This includes articles, as well as template pages, talk pages, user pages, etc. All the possible language codes can be found at

`http://wikistats.wmflabs.org/display.php?t=wp`

and are saved into a text file called `wikipedia-list.txt` in the `lorelei/wikipedia` directory of the repository.

The `get-wikipedia.py` script generates a list of dump urls in for all the languages that are smaller than the threshold size given (the threshold can be set very high so as to get all the urls). It is then possible to pull the data for all language dumps by running the `wget` command with the `-i` flag to pass the filename containing the generated list.

Store these compressed files in a folder named `compressed`. Now, Create a folder named `uncompressed` in the same directory, create a link to (or move) wikiextractor and run the script `decompress-all.sh` in that directory.

This will store the extracted text of the articles for each language in the `uncompressed` directory.

## Training the classifier

The first step is to configure pyspark to use more memory: in `spark/conf/spark-defaults.conf`, add the following two lines at the end:

```
spark.driver.memory 9g
spark.driver.maxResultSize 4g
```

The classifier expects the uncompressed, extracted wikipedia dumps to be present in the directory from which it is run, in the `wp-data` directory. The simplest way to do this is normally to create a symbolic link.

Once this is done, the classifier can be trained by opening a pyspark shell and running the file `spark-wikipedia.py` (just run `execfile("spark-wikipedia.py")`). This will train the model and save it to the `model.save` directory, from which it can be loaded later. It will also generate a confusion matrix (in CSV format) from the Wikipedia testing set, which can be used to check model performance.

If the `model.save` directory already exists, the script will stop without overwriting it (but all training will be lost).

## Classifying documents

Once the model is trained and saved, the `warc-classify.py` script can be used to run it on a WARC file.