Description of v1 9/13/2023

Note that the zip file includes all the code and output in the file structure as specified below. You will just need to add and populate the three pos subdirectories with their respective pos files if you want to run the code yourself on the same data. Or you can create a new subdirectory under /corpora to process a new corpus.

File structure for data to be processed:

…/AskMe/data/corpora/<corpus\_name>/pos

The corpus name should be one of bio, mol, geo.

Within the pos subdirectory, there should be a set of .txt files corresponding to the pos files produced by Marc (as, for example, "processed\_pos\_bio".) These contain a sentence followed by part of speech info for each word, a web address, and then noun phrase info.

The code directory contains a set of lexicons used for part of speech selection (each prefixed by lexicon\_) and two python modules: pos2phr.py and np.py.

In a python 3 shell within the code directory, run:

(hs, h, ms, m, hr, mr) = pos2phr.run\_all()

This creates a set of json files in the <corpus\_name> data directory, and also returns a set of dictionaries which can be queried interactively in python. It takes 5-10 minutes to run on a mac.

Json files produced:

heads.json

mods.json

heads\_trimmed.json

mods\_trimmed.json

head\_ratio.json

mod\_ratio.json

cf.json

df.json

The most useful of these is heads.json, which contains a list of head terms and, for each such term, the phrases containing the head, along with their corpus and document frequencies, e.g.:

{

"function": [

[

"wave function",

1653,

666

],

[

"basis function",

827,

426

],

[

"CASSCF wave function",

33,

21

],

[

"time correlation function",

182,

99

],

[

"correlation function",

1525,

574

],

…

The file np.py is designed to work on a subset of files for which the above data has been preprocessed. This subset might be a cluster of documents or a result set from a query. The subset is stored as a json file containing the cluster name and a list of files within that cluster, e.g.:

{

"3": [

"620d2300ad0e9c819b062291",

"620ddab5ad0e9c819b0a03d1",

…

"620db65fad0e9c819b0972bf"

],

"0": [

"620e687bad0e9c819b0c31c9",

"620d913cad0e9c819b08e598",

"620f2eb1ad0e9c819b108751",

…

np.py is set up to be used interactively in python and does not generate any output files.

np.heads("1") # run over a cluster name, such as “1”

# This outputs the head and its phrases for the top heads in a cluster

Currently, heads produced by np.heads are sorted by the number of different phrases they occur in. However, number alone may not be the best criteria for sorting head terms.

For the domain as a whole, head terms tend to provide useful sets of categories, or facets, for slicing the data. A hand-created summary of some of the most useful heads for the geo domain is shown below, where each head term is followed by its category and/or example modifiers. It would be interesting to see if chatgpt could be used to cluster some of the head terms which serve the same function (e.g., pull together all of those heads which refer to company names or place names.)

Report: type of report (feasibility, resource, property)

Resource: material (coal, zinc, brine)

Reserve: minable, open pit, water, oxide

Mine: name of mine

Property: name of property (Saxendrift, Abcourt\_Barvue, Vendome)

District: location (Beaverlodge, Batopilas)

Production: material produced (antimony, zinc, brine)

Project: name of project (Bisset Creek, Lamaque)

Estimate: coal resource, preliminary mineral resource, mineral reserve

Work: types of work (prospecting, historical, test, exploration, metallurgical)

Area: ? some names if filtered for capitalization

Subsidiary: ? mixed

Basin: location names

Group: company? Investors? (Windsor, Print Albert, Timiskaming)

Inc.: company name

Ltd: company name

Mining: mining company name (Adanac, Mantra, Patricia)

Energy: company names, if filtered by capitalization

Glacier/Mountain/Valley: geographical place names