WIKI HACKATHON FOR IPCC

Our current Wiki Hackathon is a project that aims to perform text and data mining on IPCC report of 2021 (a global climate report) to extract knowledge using tools that are created by us, and helps us save time, and gain meaningful insights by performing several analysis.

On a document that has 10,000 pages and several 100 digrams and tables, extracting meaning can be daunting. But not, if the machines do the reading of the document for us. And not just reading, but also sectioning it into images, tables and chapter subsections. And not just this, but analysing the text for us, and sending us the analysis in a way that can be understood by us, and meaningful knowledge extracted out of the document.

And the time it takes for a machine to do it? It definitely less that reading the entire document. Much less. In fact, if the tools are installed properly, and run in the right way, we can have meaning from 10,000 pages in mere hours, or even one hour!!!!

We do this currently using 3 tools, PYAMI, DOCANALYSIS and PYGETPAPERS.

These have been developed by us, and are currently going through further improvements. But what we have right now is a very powerful symphony of tools that goes through texts and gives out “knowledge” and not just “information”.

Following is a detailed description of tools, and how to use them.

PYAMI has been used for:

- Converting pdf to html

- Validating whether the created dictionary is correct or not

- Annotating the dictionary, which means, one can find the word from the dictionary in the generated html file, and then go to the wikidata page for the word, just by clicking on the word.

It can be used either by downloading and installing directly using “pip install”, or by cloning it from Peter’s repository. Currently, both methods are being optimised, but the following commands are the ones, that have given us the result.

Thus, the tool allows not only to find the words that are most often used and are important for the document, but also gives a method to understand in detail the meaning of that particular word.

DOCANALYSIS has been used for:

- Creating automated dictionary from a html file.

- Creating different entity dictionaries (abbreviation dictionary, organisation dictionary etc.) from the same html file

It can be dowloaded and installed directly using “pip install”

PYGETPAPERS has been used for:

- downloading the xml, pdf and html version of document from the web using keyword search.

It can be dowloaded and installed directly using “pip install”

THE PROTOCOL:

- **Download the IPCC/ar6/wg3 Chapter* pdf:**
  - from IPCC website (https://www.ipcc.ch/report/ar6/wg3/)
  - from Peter’s github repository (https://github.com/petermr/semanticClimate/tree/main/ipcc/ar6/wg3)

- **Download Peter’s semanticClimate github repository:**
  ``` git clone https://github.com/petermr/semanticClimate.git ```

- **Use pycharm or a similar IDE environment that can be used to**
  - make dictionaries
  - identify the problems in the dictionary
  - upload the changes made in the dictionary or other texts in Peter’s github repository.

- **Converting a pdf into html (we use pyami from Peter’s repository)**

  Download pyami repository:
  - git install the latest pyami from Peter’s repository:
    ``` git install https://github.com/petermr/pyami.git ```

  - Then enter inside pyami directory:
    ``` cd /some/where/pyami```

  Convert pdf to html:
  - Use following command for pdf to html conversion:

```
python -m py4ami.ami_pdf \
  --inpath /users/……/ipcc/ar6/wg3/Chapter*/Chapter.pdf \
  --outdir /users/……/ipcc/ar6/wg3/Chapter*/ --maxpage 100
```

  where,
  - inpath – path of the pdf file inside your machine
  - outdir – path of the directory inside which the converted html will be saved
  - maxpage – number of pages of the pdf that should be converted into html

Note: Currently, for the maxpage argument, we only give the number of pages before the reference section.

- **Create manual dictionary from pdf**
```

This is done by manually reading the pdf, and manually entering any term or abbreviaion that you donot understand, or you think is important for the chapter. The dictionary is a xml file (emissions.xml). We use “pycharm” for making our manual dictionaries.

- Read the pdf, and as you go through the text, enter any word or abbreviation that you do not understand in the dictionary, with descriptions.

- **Dictionary descriptions**:

- eg of a dictionary:

```
<dictionary title="emissions" version="0.0.5" >
  <desc> manually created by Abhishek Prasad 2022-08-23 </desc>
  <entry term="carbon budget" wikidataID="Q16722080"
         wikipediaPage="https://en.wikipedia.org/wiki/Carbon_budget" desc="limit
on CO2 emission for a given climate impact"/>
  <entry term="FFI" name="fossil fuel combustion and industrial processes"
wikidataID="" desc=""/>
  <entry term="LULUCF" name="land use, land-use change, and forestry"
wikidataID="Q3348639"
         wikipediaPage="https://en.wikipedia.org/wiki/Land_use,_land-
use_change,_and_forestry"
         desc="Greenhouse gas inventory sector that covers emissions and
removals of greenhouse gases resulting
               from direct human-induced land use"/>
  <entry term="FOLU" name="forestry and other land-use" wikidataID="Q3348639"
         wikipediaPage="https://en.wikipedia.org/wiki/Land_use,_land-
use_change,_and_forestry"
         desc="Greenhouse gas inventory sector that covers emissions and
removals of greenhouse gases resulting
               from direct human-induced land use"/>

</dictionary>
```

- **Create a xml dictionary**, usually as chapter.xml (eg. emissions.xml, urban.xml)

- On the first line, **enter the dictionary title and the versions**:

```
<dictionary title="emissions" version="0.0.5" >
```

- Enter the term in <entry> and then, enter the attributes like <term>, <name> etc.

- Following are the meaning of each attribute:

<entry> the entry term. It can be a word, or the short of an abbreviation

<name> full form of an abbreviation

<wikidataID> the id from wikipedia of each entered word or abbreviation. If not found, leave empty

<wikipediaPage> the web address of the wikipedia page of the term.

<desc> a one line description of the term.

- If available, copy it from the wikidataID descriptions

- If not available, give a one line description from wikipedia page

- If the wikipedia page does not exist for the word, give a one line description from a different web page.

< descPage> web address of the description page if the wikipedia page does not exist for the word.

- **Close the dictionary** </dictionary>.

- **Create automated dictionary from pdf**

For this, we use the tool “docanalysis”.

- Docanalysis installation:

- Create a separate directory

``` mkdir /user/.../docanalysis

**- Run the following installation command:**

```pip install docanalysis```

**- Test if installation is successful using the following command:**

```docanalysis –help```

**- If the tool is successfully installed, perform following commands:**

```pwd```

**- Creating sections in the docanalysis**:
This step is required, as currently docanalysis works by sectioning the xml or html file into sections. But, making sections on a html file using docanalysis shows error. So, we do it manually, as follows:

**- If you are inside docanalysis directory, then do the following:**

```mkdir wiki_hackathon
cd wiki_hackathon
mkdir Chapter02
cd Chapter02
mkdir sections
cd sections
mkdir 0_main_body```

- Place the html of your chapter inside the 0_main_body directory.

**- After placing it, use ```pwd```**

- the result should be
```

"/user/…./docanalysis/wiki_hackathon/Chapter02/sections/0_main_body/"

- Running docanalysis for creating automated dictionary:

- Now, come at the directory where the wiki_hackathon folder is there:

```pwd```

Result: "/user/…./docanalysis/"

-Here, run the following command, which prepares the html for dictionary creation:

```docanalysis --project_name wiki_hackathon --output entities.csv --make_ami_dict entities.xml```

Then, run the following command, which creates the abbreviation dictionary:

```docanalysis --project_name wiki_hackathon --output dict_search_5.csv --make_json dict_search_5.json --make_ami_dict entities --extract_abb emissions_abb```

where,

--project name – the name of the project (here, wiki_hackathon)
--output - a csv for dictionary search (not of our use, but required to be created)
--make_json - just enter this. Not of current use, but required.
--make_ami_dict – uses the entities created in the above command
--extract_abb - the abbreviation dictionary that is the output.

Note: While using the above two commands, don't worry about the output paths. The command searches and produces results as it is.

-Making automated dictionaries other than abbreviation dictionary using docanalysis:

ORGANISATION dictionary:

```docanalysis --project_name wiki_hackathon --spacy_model spacy --entities ORG --output org.csv```

```docanalysis --project_name wiki_hackathon --spacy_model spacy --entities ORG --output org_aut_aff.csvv --make_ami_dict org```

-To upload this dictionary in Peter's github page, place the results in semanticClimate path

"/user/…./ipcc/ar6/wg3/Chapter02/abb.xml"

and upload on github using pycharm or other IDE (we use pycharm).

One can create other dictionaries as well, as described in the github page for docanalysis. Please refer to the docanalysis page
```

https://github.com/petermr/docanalysis/blob/main/README.md

- **Validating the created dictionary:**
  We do this using pyami from the repository

  - Enter into the cloned pyami repository folder

  ```cd /user/.../peter_github_repository/pyami/```

  - Then, do ```ls``` to see if the different directories are showing

  - Then, use the following command:

  ```python -m py4ami.pyamix DICT --dict /Users/pm286/projects/semanticClimate/ipcc//ar6/wg3/Chapter02/dict/emissions.xml –valid```

  where,
  --dict – the path of the dictionary to be validated, with the name of the dictionary as well.

- **Annotating the created dictionaries:**
-

  This is done by using pyami directly by installing using pip install.

  - Download and install the py4ami using following command:

  ```git install py4ami```

  - Then, use the following command:

```py4ami HTML --annotate \
--dict /Users/pm286/projects/semanticClimate/ipcc/ar6/wg3/Chapter02/dict/emissions.xml \
--inpath /Users/pm286/projects/semanticClimate/ipcc/ar6/wg3/Chapter02/fulltext.html \
--outpath  /Users/pm286/projects/semanticClimate/ipcc/ar6/wg3/Chapter02/annotated/fulltext_emissions.html --color YELLOW```

  where,
  --dict – the path of the dictionary to be annotated with the dictionary name as well
  --inpath – the path of the full text html with the name of the html file as well
  --outpath – the path of the output, with the name of the output html file
  --color - the colour in which the annotated words should be highlighted in the annotated html

- **Literature Search: Downloading any paper from the web using pygetpapers**

  **- Install the tool by using the following command:**

  ```pip install pygetpapers```
  ```

- Test if the tool has been installed:

```pygetpapers --help```

- Then run the following command to download papers:

```pygetpapers -q "METHOD: invasive plant species" -k 10 -o "invasive_plant_species_test" -c --makehtml -x –save_query```

where,

-q “METHOD” – the place to enter the query keyword search. It must be inside the “” as shown above

- -k 10 – number of results. The number can be changed to get more or less results

- -o – directory where the result will be saved

--makehtml – the result will also be in html. If this not entered, result will be in xml

-c - saves per-paper metadata into a single csv

--save_query – saves the result in the output directory