MAPD-B distributed processing exam default projects

1 Analysis of Covid-19 papers

1.1 Introduction

Since this year our activities are determined by the COVID-19 pandemic, the distributed computing project will be focused on the analysis of 1000 papers about COVID-19, SARS-CoV-2, and related corona viruses. The dataset is a sub-sample of 1000 items taken from the original dataset that is composed of more than 75000 papers. This dataset is a part of real-world research on COVID-19 named COVID-19 Open Research Dataset Challenge (CORD-19). The research and related challenges are available on the dedicated page on Kaggle: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

1.2 Structure of the Data

All the documents presents in the *data* folder are structured in this way:

```
{
    "paper_id": <str>,      # 40-character sha1 of the PDF
    "metadata": {
        "title": <str>,
        "authors": [        # list of author dicts, in order
            {
                "first": <str>,
                "middle": <list of str>,
                "last": <str>,
                "suffix": <str>,
                "affiliation": <dict>,
                "email": <str>
            },
            ...
        ],
        "abstract": [          # list of paragraphs in the abstract
            {
                "text": <str>,
                "cite_spans": [ # list of character indices of inline
                   citations
                                # e.g. citation "[7]" occurs at positions
                                   151-154 in "text"
                                #linked to bibliography entry BIBREF3
                    {
                        "start": 151,
                        "end": 154,
                        "text": "[7]",
                        "ref_id": "BIBREF3"
                    },
                    ...
                ],
                "ref_spans": <list of dicts similar to cite_spans>,
                # e.g. inline reference to "Table 1"
                "section": "Abstract"
            },
            ...
        ],
        "body_text": [      # list of paragraphs in full body
                            # paragraph dicts look the same as above
            {
                "text": <str>,
                "cite_spans": [],
                "ref_spans": [],
                "eq_spans": [],
                "section": "Introduction"
            },
            ...
            {
                ...,
                "section": "Conclusion"
            }
        ],
        "bib_entries": {
            "BIBREF0": {
                "ref_id": <str>,
                "title": <str>,
                "authors": <list of dict>  # same structure as earlier,
                                           # but without ‘affiliation‘ or
                                              ‘email‘
                "year": <int>,
                "venue": <str>,
                "volume": <str>,
                "issn": <str>,
                "pages": <str>,
                "other_ids": {
                    "DOI": [
                        <str>
                    ]
                }
            },
            "BIBREF1": {},
            ...
            "BIBREF25": {}
        },
        "ref_entries":
            "FIGREF0": {
                "text": <str>,         # figure caption text
                "type": "figure"
            },
            ...
            "TABREF13": {
                "text": <str>,        # table caption text
                "type": "table"
            }
        },
        "back_matter": <list of dict>  # same structure as body_text
    }
}
```

Data may contain empty fields and malformed value, so pay attention.

1.3 Assignment

1.3.1 Word counter distributed algorithm

First of all, you have to implement the following distributed algorithm to count the occurrences of all the words inside a list of documents. In NLP (Natural Language Processing) a documents is a text, in this case, each paper is a document. The algorithm is defined as follow:

Map phase : For each document Di, produce the set of intermediate pairs $(w, cp(w))$, one for each word $w \in Di$, where $cp(w)$ is the number of occurrences of w in Di. E.g.: $('hello', 3)$

Reduce phase : For each word w, gather all the previous pairs $(w, cp(w))$ and return the final pair $(w, c(w))$ where $c(w)$ is the number of occurrences of w for all the Documents. In other words, $c(w)$ is equal to $\sum_{k=1}^{n} cp_k(w)$

The algorithm has be run on the full-text of the papers. To get the full text of the paper you have to transform the input data by concatenating the strings contained into the *body-text* fields of the JSONs.
To perform this transformation I strongly suggest you use the Bag data-structure of DASK. Anyway if you prefer to implement the algorithm by using the DataFrame structure feel free to do it.
The algorithm has be run several times by changing the number of workers and the number of partitions. For each run the execution time must be registered. Provide a comment on how change the computation time over the cluster by changing the partitions and the number of workers.
You have to try with at least 6 different partition numbers.
At the end of the algorithm analyze the top words and see how they are related to the viruses and the research (for example create a barplot of the top words)

1.3.2 Which are the worst and best represented countries in the research?

In this part you have to take the documents and to convert them in a usable DataFrame data structure in order to figure out the countries that are most and less active in the research. To do this you can use the country of the authors. Do the same for the universities (affiliations).
Even in this case do multiple runs by changing the number of partitions and workers and then describe the behaviour of the timings.

1.3.3 Get the embedding for the title of the papers

In NLP a common technique to perform analysis over a set of texts is to transform the text to a set of vectors each one representing a word inside a document. At the end of the pre-processing the document will be transformed into a list of vectors or a matrix of $n \times m$ where n is the number of words in the document and m is the size of the vector that represents the word n.
More information about word-embedding: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
What you can do is to transform the title of the paper into its embedding version by using the pre-trained model available on FastText page: https://fasttext.cc/docs/en/pretrained-vectors.html.
The pre-trained model that you have to download is the https://dl.fbaipublicfiles.com/fasttext/vectors
wiki/wiki.en.vec
Basically the pre-trained model is more or less a huge dictionary in the following format $key : vector$.
To load the model follow this snippet of code which is slightly different from what you can find at this page https://fasttext.cc/docs/en/english-vectors.html

```python
import io

def load_vectors(fname):
    fin = io.open(fname, 'r', encoding='utf-8', newline='\n', errors='ignore')
    n, d = map(int, fin.readline().split())
    data = {}
    for line in fin:
        tokens = line.rstrip().split(' ')
        data[tokens[0]] = list(map(float, tokens[1:]))
    return data

model = load_vectors('wiki.en.vec')

#to get the embedding of word 'hello':

model['hello']
```

Once you have loaded the model, use the map approach to create a DataFrame or a Bag that is composed by:

- paper-id
- title-embedding

The title embedding can be a list of vectors or can be flattened to a large vector.

1.3.4 Bonus point

Use the previously generated vectors to compute the cosine similarity between each paper and to figure out a couple of papers with the highest cosine similarity score.
This point is a bonus/optional point.

1.4 Hints

- The text of the documents must be sanitized before performing analysis (remove punctuation, stop-words).
- Not all words are contained in the pre-trained model. In this case, skip the word and go to one.
- Remember that if you use the distributed "read file" the files must be in the same path for each worker
- If you need help or have questions contact us via email at the addresses: stefano.campese@pd.infn.it, jacopo.pazzini@unipd.it