New York Times Article Abstract Analysis using Hadoop and NLTK

By: Lucas Nunno (lnunno@cs.unm.edu)

Part 1: Data Acquisition

Data acquisition

- Used the python **requests** module.
  - Used the offset parameter to load new pages of abstracts and slept 1/8th of a second between each request to abide by the NYT API terms of use.
- Loaded JSON response into python dictionary and then exported as a single large JSON file containing all the articles and all metadata. (**~40,000**)
- In a separate script, I export this JSON data to a CSV file with the docIDs, URLs, and abstracts.
  - This is also where I check for **duplicates**. I have a set of URLs that the exporter has seen, if this URL is in this set the program prints a warning and does not export it.

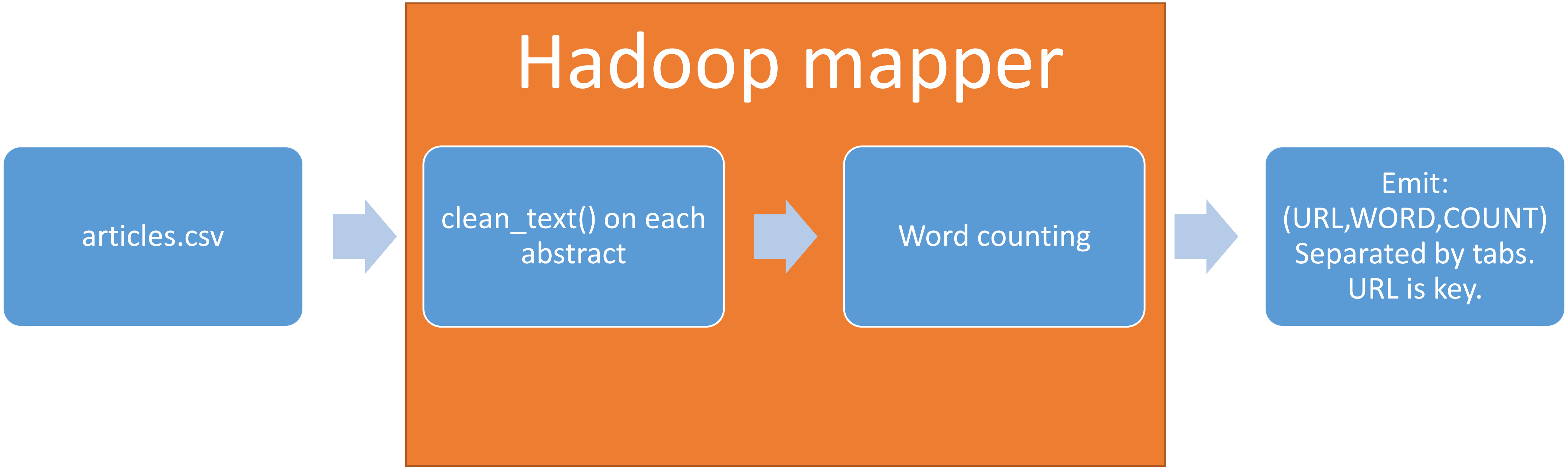
Part 2: Preprocessing and tf-idf

Preprocessing

- Used the python natural language toolkit (NLTK) module for most of the preprocessing tasks. The algorithm is as follows:
  1. Convert text to lowercase.
  2. Remove punctuation and numbers.
     - Simple regex substitution:

     ```
     remove_pattern = re.compile(r'[^a-z\s]')
     ```
  3. Remove stopwords.
     - See: `nltk.corpus.stopwords`
  4. Stem all the remaining words.
     1. See: http://www.nltk.org/api/nltk.stem.html#module-nltk.stem.porter
  5. Output the cleaned abstract.

How is this parallelized in Hadoop?



Hadoop mapper

articles.csv

clean_text() on each abstract

Word counting

Emit:
(URL,WORD,COUNT)
Separated by tabs.
URL is key.

tf-idf

- Dictionary is constructed of each document, the words it contains, and the frequencies of these words – this is all provided from the mapper.
- Frequency matrix is constructed for all documents.
  - Items without entries are filled with zeroes.
- Augmented (normalized) frequency matrix is calculated.
  - This removes the bias for longer documents.

$$\boxed{\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}}$$

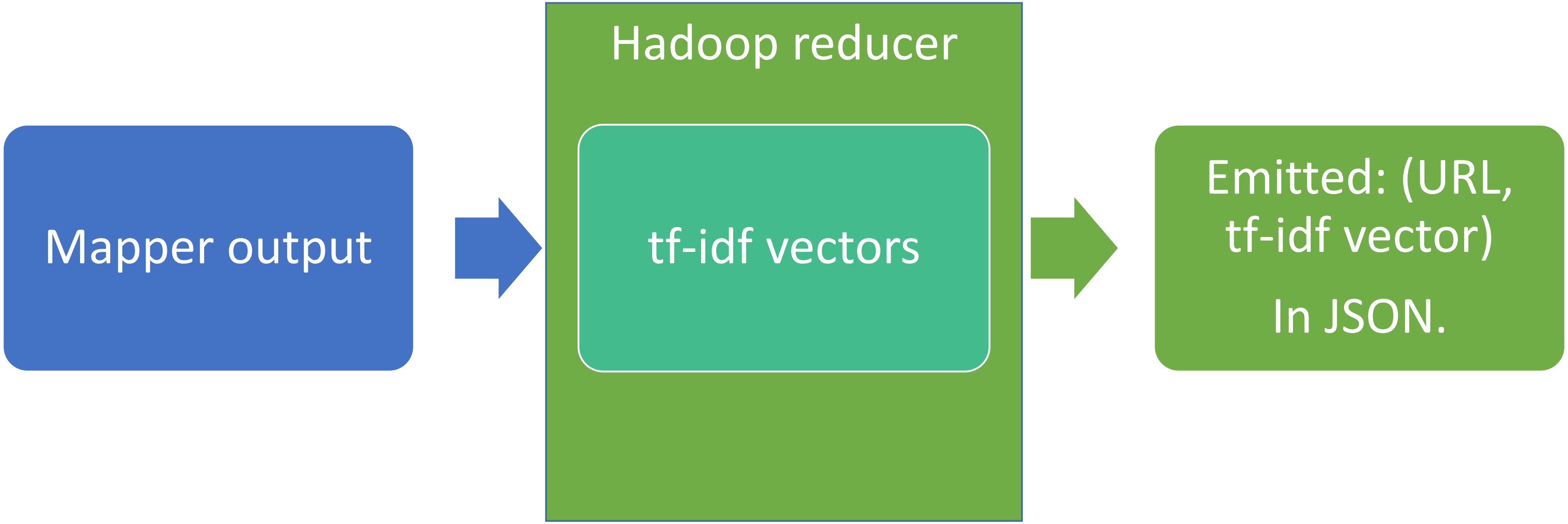
tf-idf (contd.)

- Document frequency vector is calculated by counting the number of non-zero items on each row.
- From these data structures we can calculate the inverse document frequency (idf) vector.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- From this, we multiply the normalized frequency matrix with the idf vector to calculate the tf-idf matrix.

How is this parallelized in Hadoop?



Mapper output

Hadoop reducer

tf-idf vectors

Emitted: (URL, tf-idf vector)

In JSON.

Part 3: Clustering and Visualization

Clustering

Visualization

Thank you.

Check out the source on Github: https://github.com/lnunno/big-data-nyt-tf-idf