# Coursework : Document Indexing in MapReduce


- author: Eu-Bin KIM (10327741)

- date: 24th of November 2020
- word count : 998

## Functionality

**Stop-words filtering** is implemented in line 86 and 161 of the code. Stop-words are words that occur in abundance, such as "my", "is" and "the". They are filtered out as they rarely hold (Nenadic, 2020) semantic significance. While this might help denoise search queries, there is a domain-specific downside to this. When people search for a particular episode, one might search for a memorable quote from the episode, say, "I did mean it". However, the three terms would not be indexed as they are all listed as Stop-words in `lib/stopwords.txt`. If any of Simpson episode Wiki's contained the quote, searching the quote would return nothing.

**Domain specific filtering** - filtering out dates, urls, ISBN's, numbers and page numbers is implemented in line 79-83 of the code. Wikipedia pages include in-line citations and "References" section, both of which are not worthy of indexing simply because people would rarely use them to search for an episode. However, they might be important if we were indexing for a book search engine, as then people might want to find books by their ISBNs, or jump into a specific page in a book. This well illustrates that tokenisation process depends on (Nenadic, 2020) the domain of data.

```
allow|2   [Bart_the_Fink.txt.gz|3|[423, 548, 603], ...]
alter|2   [Bart_the_Fink.txt.gz|2|[782, 967], ...]
air|6 [Bart_the_Fink.txt.gz|3|[147, 205, 220]
```

*Figure 1* : Parts of the Inverted Index for the terms `allow` , `alter` and `american`. DF (next to the term), TF (next to the document) and positional indices (next to TF) are stored.

$$tf * idf_{t,d} = (1 + log_{10} tf_{t,d}) * log_{10}(N/df_t)$$

*Figure 2* : The formula for TF-IDF weight of a term (Nenadic, 2020).

Computation of **Term Frequency (TF) and Document Frequency (DF)** is implemented in line 281 and 320 of the code, respectively. They are needed to compute TF-IDF weights for each term according to the formula in *Figure 2*. TF-IDF weights help us measure the significance of the terms in relation to the documents where they appear. Take the index shown in `Figure 1` for example; `air` has higher DF than the other two terms while its TF being nearly identical to those of the others. Since TF-IDF is inversely proportional to DF (DF is the

denominator of the input to a monotonically increasing logarithm), we can see that `air` holds less significance in `Bart_the_Fink.txt.gz` than `allow` or `alter` does.

**Positional Indexing** is implemented in line 205-221 of the code, which utilises an instance of `Counter` to keep track of term positions. A Positional Index specifies (Manning, et al., 2008) the positions at which terms appear in their postings. For example, *Figure 1* above shows the positional index for `allow`; the term appears at 423rd, 548th and 603rd positions in the document `Bart_the_Fink.txt.gz`. Such Explicit specification of term positions is useful for an efficient proximity search. For instance, if we were to search for "allow to alter" on the positional indices above (*Figure 1*), we can use them to efficiently work out that the two terms appear closer in `Bart_the_Fink.txt.gz` than in `Bart_the_Lover.txt.gz` (782 - 603 = 179 < 581 - 329 = 252), and that the former should be more relevant to the query than the latter.

**Normalising case** of all the terms to lowercase is implemented in line 85 of the code. Some words are in uppercase only because they are used to starting a sentence (e.g. "In" as in "In the episode, ... "), and setting all of them to lowercase allows us to normalise them to match with those used in the middle of a sentence (e.g. "in" as in "... partially in response to"), thus building a much dense index. One evident disadvantage of this is that we can no longer disambiguate Proper Nouns from Nouns (e.g. "apple(fruit)" from "Apple(company")). Despite this, it should be noted that such drawbacks could be compensated with the context provided in a query (e.g. "eat apple", "apple's stock price").

**Stemming** all the terms after they are cleansed is implemented in line 87 of the code. As with lemmatization and case folding, stemming is (Nenadic, 2020) a way of normalising tokens. It works by dropping affixes of words, only leaving out stems of the words (e.g. amazing -> amaz). Although this largely helps make the index more dense, thus reducing its size as well, there is a risk of completely changing the meaning when stemming them. (e.g. clueless -> clue).

## Performance

| in-mapper aggregation | time (seconds) |
|---|---|
| NO | 21.782 |
| YES | 22.549 |

*Figure 3* : The time it took to run jobs, with and without In-mapper Aggregation

**In-mapper Aggregation pattern**, where aggregation of values takes place (Paton et al., 2020) in Mappers, is used in the code. As *Figure 3* illustrates, adopting the pattern has led to an increase in the performance. This is because

aggregating intermediate TF's and Positional Indices inside Mappers alleviated the bottleneck caused by fewer Reducers being available than they are needed. However, it should be noted that this would not have been the case if there existed numerous duplicated terms in each file split. Consider an extreme case where all the file splits contain duplicates of a token and that token only; Mappers would iterate over `tokens` twice (first for aggregation, then for emitting) only to produce the same key-value pairs as what simple Mappers would produce by iterating `tokens` only once. In this case, mappers with In-mapper Aggregation pattern would in fact take twice the time the simper one would take.

## References


- Manning, D Christopher et al., 2008, *Introduction to Information Retrieval*, viewed 23rd November 2020, https://nlp.stanford.edu/IR-book/html/htmledition/positional-indexes-1.html
- Nenadic, Goran, 2020, *Preparing to Index*, lecture notes, COMP38211, UOM
- Nenadic, Goran, 2020, *Querying and ranking: Ranked retrieval*, COMP38211, UOM
- Nenadic, Goran, 2020 *Principles of IR – Indexing*, COMP38211, UOM
- Paton, Norman et al., 2020, *Map Reduce Design Patterns*, COMP382111, UOM