Document/Page ranking using tf-idf Weighting Scheme

CSE 633(SPRING 2020)

Instructor: Dr. Russ Miller

Name: Mansi Shetty

University at Buffalo The State University of New York

Overview

- Terminologies and Definitions
- Problem Statement
- Applications
- Sequential Algorithm
- Parallel Implementation
- Results
- Observations
- References

Terminologies and Definitions

Document

A document is a collections of terms/words. Examples: Web page, tweet

Corpus

It is a collection of documents

Term Frequency (TF)

It is a document weighting scheme that takes into account the number of occurrences of the term t in document d.

Inverse Document Frequency (IDF)

It is defined as, $idf(t) = \log(N/df)$

where, N is the total number of documents in the corpus and df is the document frequency which indicates the number of documents in the corpus that contain the term t.

Problem Statement

Rank documents/search results based on Term Frequency – Inverse Document Frequency, a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

$$\text{Score}(q, d) = \sum_{t \epsilon q} \text{tf-idf}_{t,d}$$

Example

- Lets consider a corpus with two documents i.e. N=2
- For document 1 (d1),

$tf_{this,d1} = 1/5$, $idf_{this} = \log(2/2) = 0$

$tf_{is,d1} = 1/5$, $idf_{is} = \log(2/2) = 0$

$tf_{a,d1} = 2/5$, $idf_{a} = \log(2/1) = 0.301$

$tf_{sample,d1} = 1/5$, $idf_{sample} = \log(2/1) = 0.301$

- For document 2 (d2),

$tf_{this,d2} = 1/7$, $idf_{this} = \log(2/2) = 0$

$tf_{is,d2} = 1/7$, $idf_{is} = \log(2/2) = 0$

$tf_{another,d2} = 2/7$, $idf_{another} = \log(2/1) = 0.301$

$tf_{example,d2} = 3/7$, $idf_{example} = \log(2/1) = 0.301$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Therefore,

$tf\text{-}idf_{sample,d1} = (1/5) * 0.301 = 0.0602$

$tf\text{-}idf_{sample,d2} = 0 * 0.301 = 0$

$tf\text{-}idf_{example,d1} = 0 * 0.301 = 0$

$tf\text{-}idf_{example,d2} = (3/7) * 0.301 = 0.129$

Applications

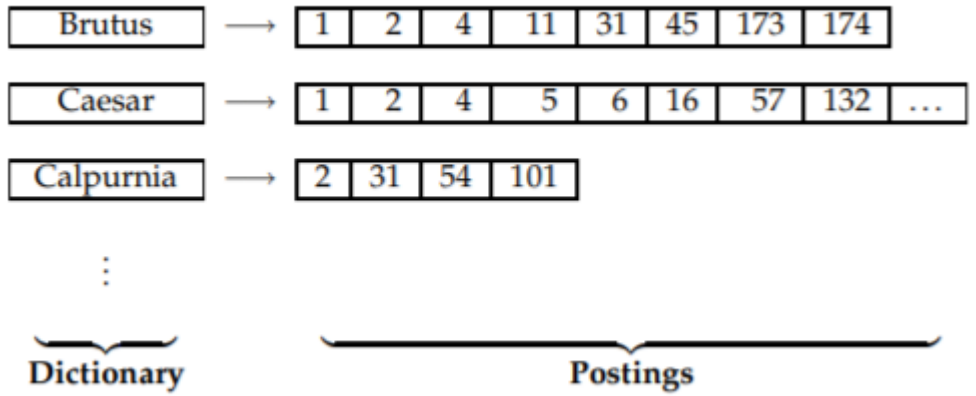
- Information retrieval
- Web search
- Keyword Extraction
- Stop words elimination

In applications like above, the time taken to return the results becomes the most important factor and given the amount of data that the internet has to offer today, computations takes a significantly longer time.

Sequential Algorithm

- Create an inverted index

  {term: (document frequency, [document list])}
- Read the query terms and create a list of resultant documents
- Calculate term frequency $tf_{t,d}$
- Calculate inverse document frequency idf_t
- Calculate tf-idf score for each document
- Sort the documents based on tf-idf score



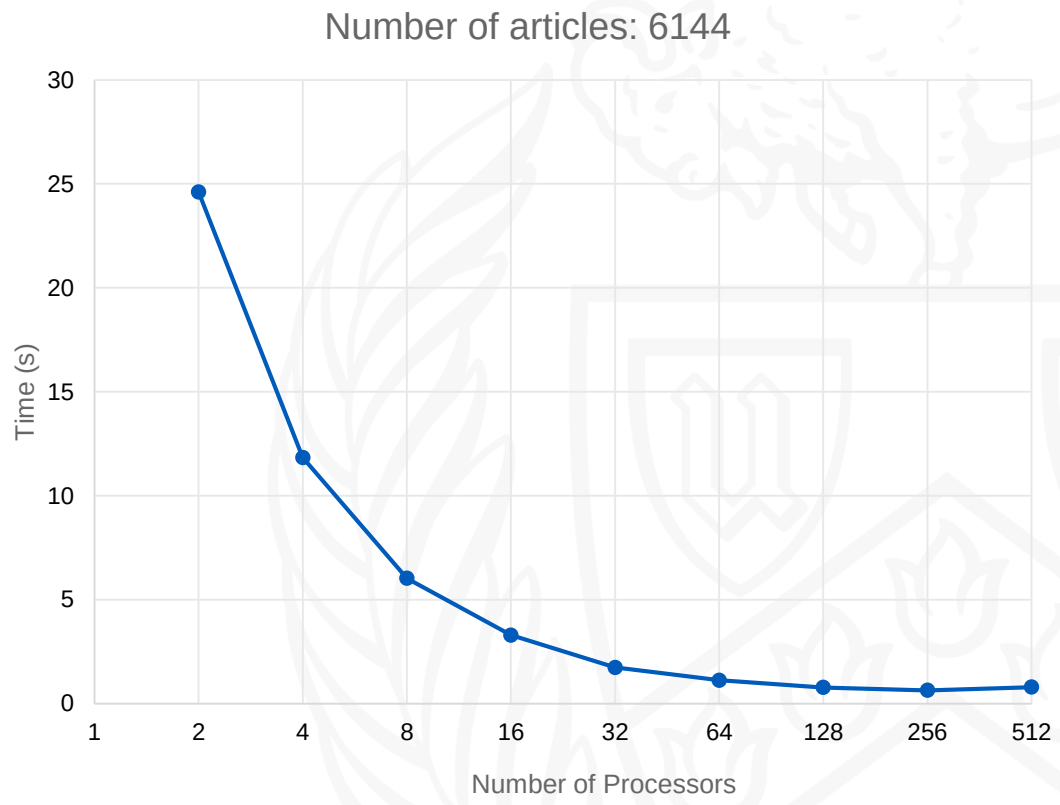
Parallel Implementation

- Consider N documents and P processors.
- Assign N/P documents to each processor.
- Each processor creates the inverted index for its N/P documents independently.
- The file with the queries is read by each processor.
- Document frequencies of all the terms of the query are consolidated by recursive halving and broadcasting.
- Processors independently form the resultant set from their inverted index.
- Results in each of the processors are ranked according to tf-idf weighting scheme.
- Final result for each query is consolidated by recursive halving where the task is to merge two sorted lists.

RESULTS

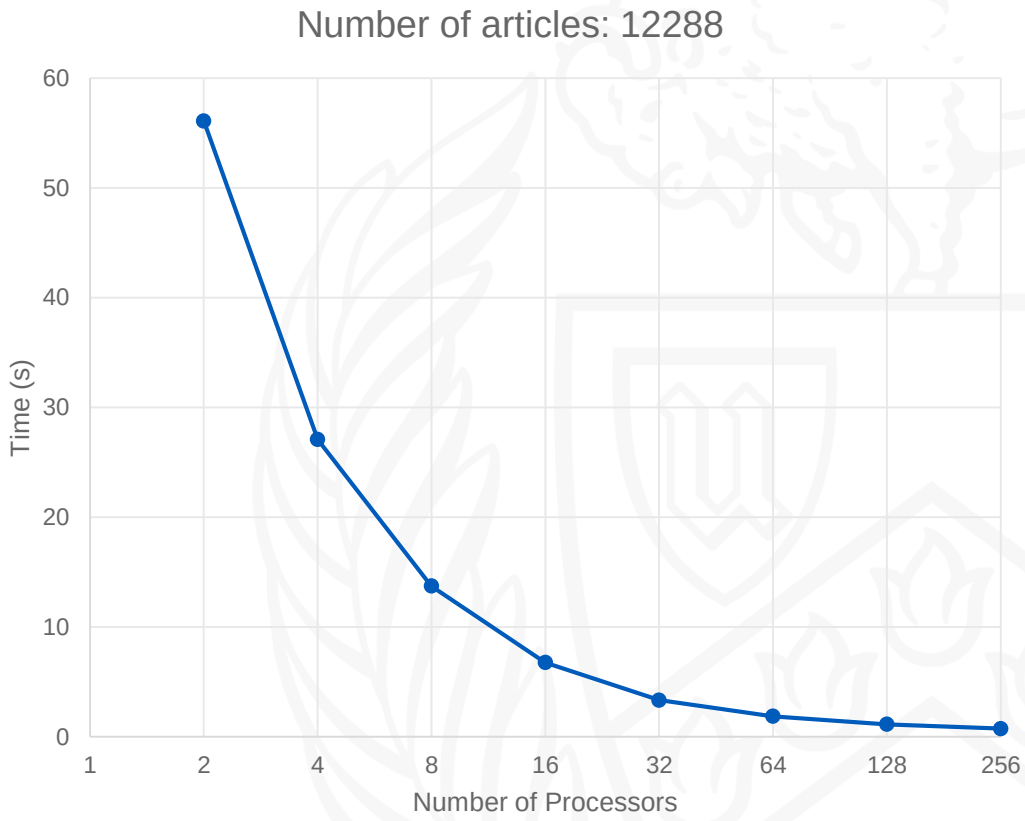
Total number of articles: 6144

Number of Processors	Time (s)
2	24.598
4	11.837
8	6.024
16	3.288
32	1.735
64	1.120
128	0.771
256	0.632
512	0.786



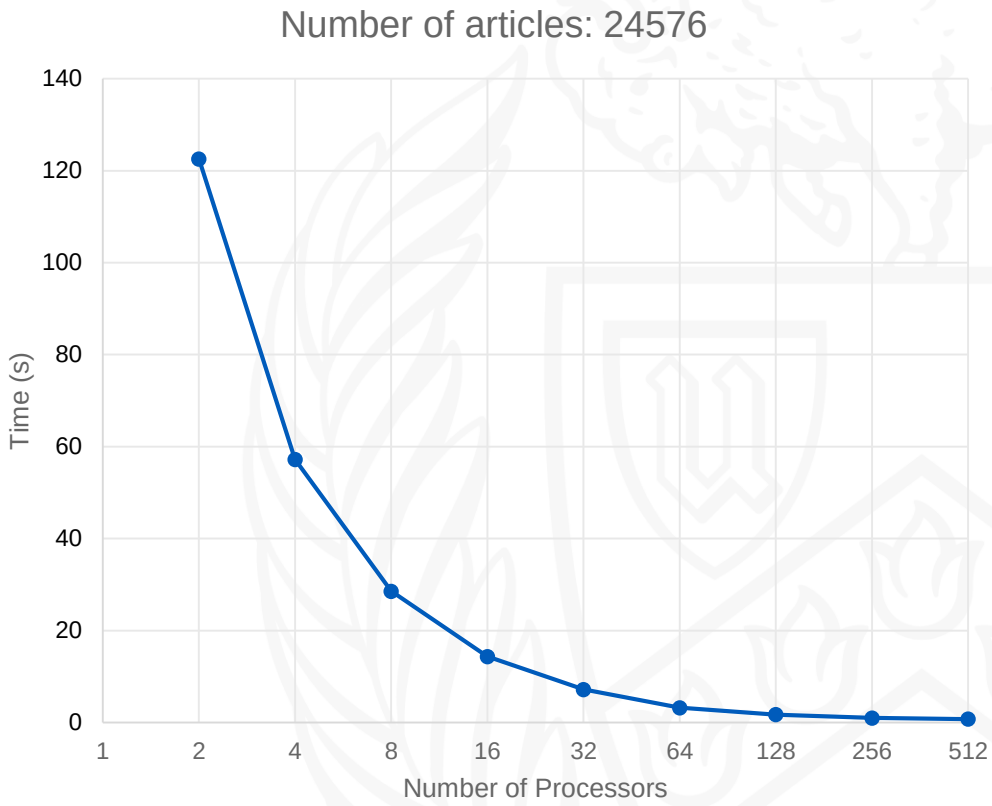
Total number of articles: 12288

Number of Processors	Time (s)
2	56.093
4	27.075
8	13.719
16	6.757
32	3.330
64	1.852
128	1.126
256	0.739



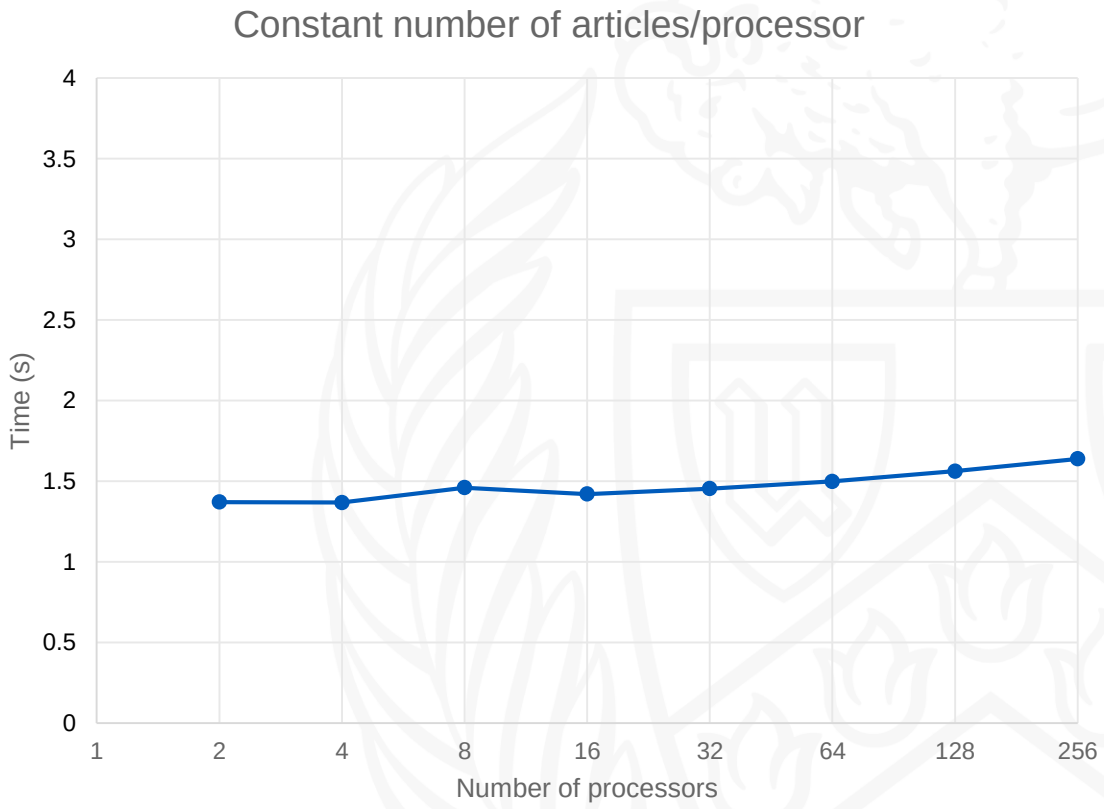
Total number of articles: 24576

Number of Processors	Time (s)
2	122.559
4	57.218
8	28.548
16	14.354
32	7.184
64	3.202
128	1.725
256	0.995
512	0.759



Constant number of articles/processor (165 articles/processor)

Number of processors	Articles	Time(s)
2	330	1.370
4	660	1.368
8	1320	1.459
16	2640	1.420
32	5280	1.454
64	10560	1.498
128	21120	1.562
256	42240	1.638



Observations

- For 6144 total articles, as we increase processors beyond 256, the total time increases as the communication overhead overpowers reduction in computation time.

- For 24576 total articles, there is not a significant decrease in total time when we increase the processors from 256 to 512.

- When we maintain a constant number of articles/processors, the computation time remains comparable with slight increase as we increase the number of processors.

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer)
- https://nlp.stanford.edu/IR-book/
- https://en.wikipedia.org/wiki/Tf%E2%80%93idf
- https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089
- https://mpi4py.readthedocs.io/en/stable/
- https://www.kaggle.com/snapcrack/all-the-news

Thank You!