Document/Page ranking using tf-idf Weighting Scheme with OpenMP

CSE 702(FALL 2020)

Instructor: Dr. Russ Miller

Name: Mansi Shetty

University at Buffalo The State University of New York

Overview

University at Buffalo The State University of New York

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

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

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

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,

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

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

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

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

Applications

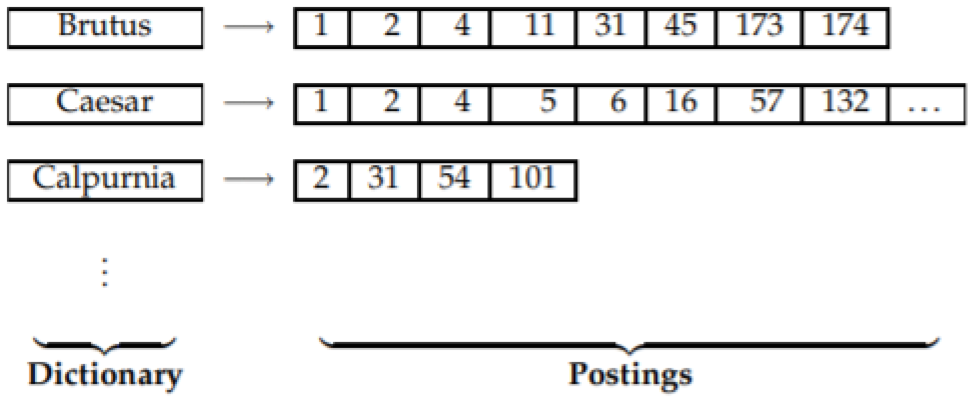
- Information retrieval
- Web search
- Keyword Extraction
- Stop words elimination

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 cores (1 thread per core).
- Assign N/P documents to each core.
- Each core creates the inverted index for its N/P documents independently.
- The file with the queries is read into shared memory.
- Each core calculates the document frequency for each of the query terms and then saves it to the shared memory.
- Summation of the document frequencies is calculated in the shared memory and then read by each core.
- Each core independently forms the resultant set from their inverted index.
- Tf-idf score is calculated by the cores for their respective resultant documents.
- Each core sends their tf-idf scores to the shared memory.
- These documents are then sorted by their tf-idf scores.

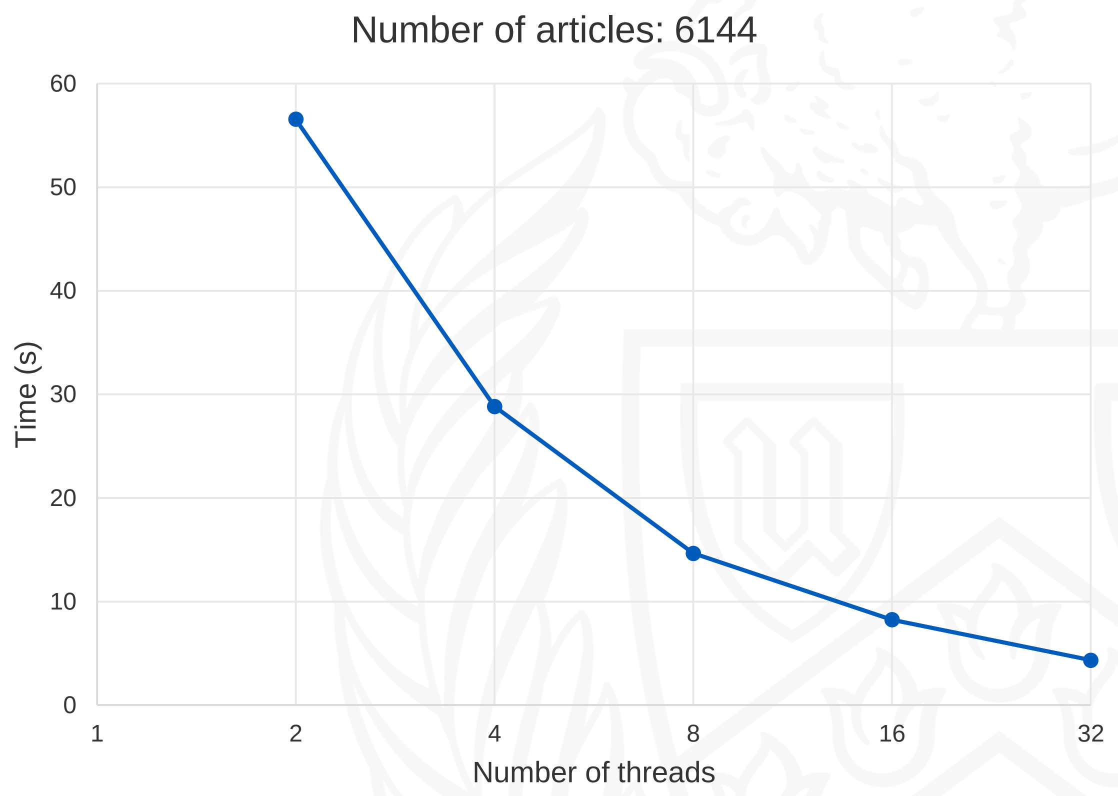
University at Buffalo The State University of New York

RESULTS

University at Buffalo The State University of New York

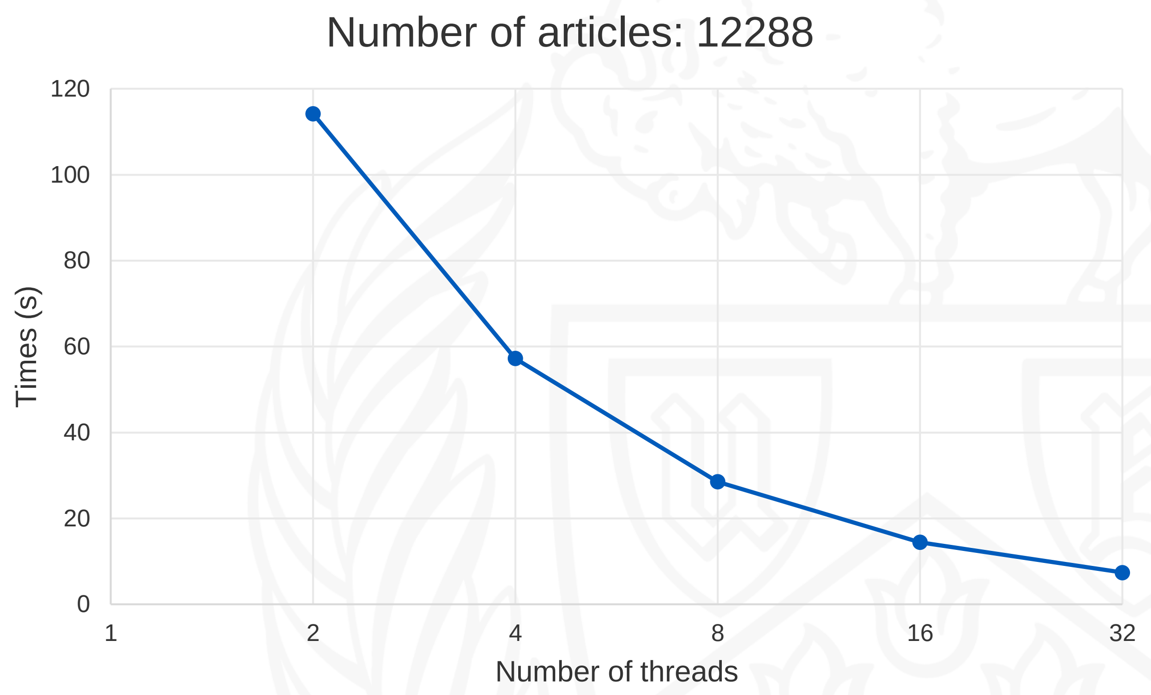
Total number of articles: 6144

Number of threads	Time (s)
2	56.568
4	28.841
8	14.67
16	8.252
32	4.34



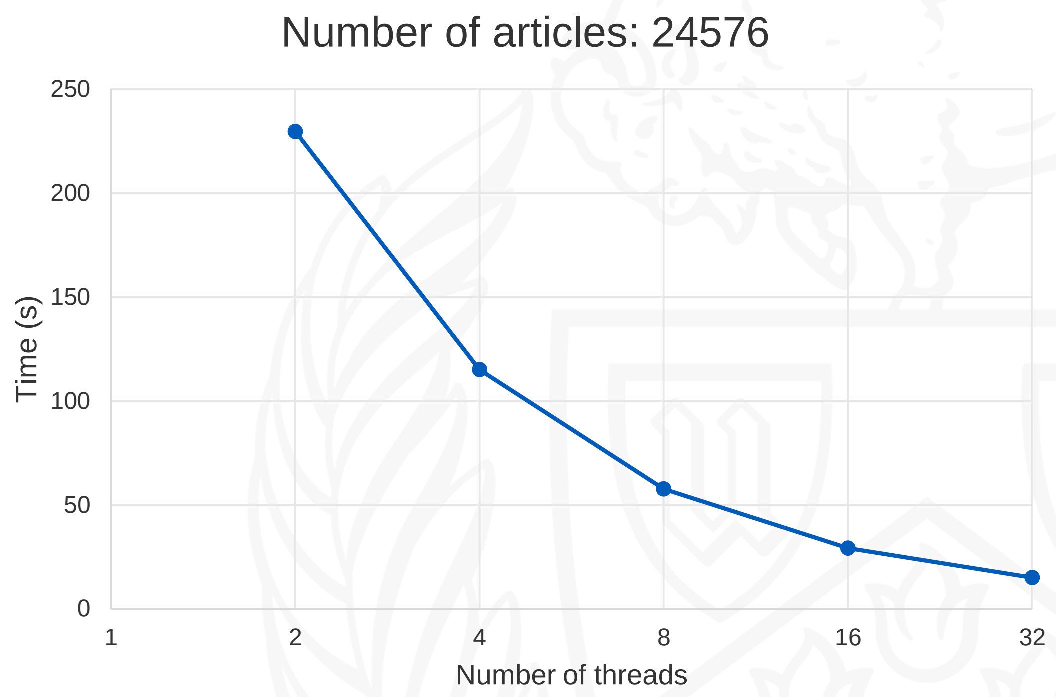
Total number of articles: 12288

Number of threads	Time (s)
2	114.19
4	57.223
8	28.534
16	14.47
32	7.418



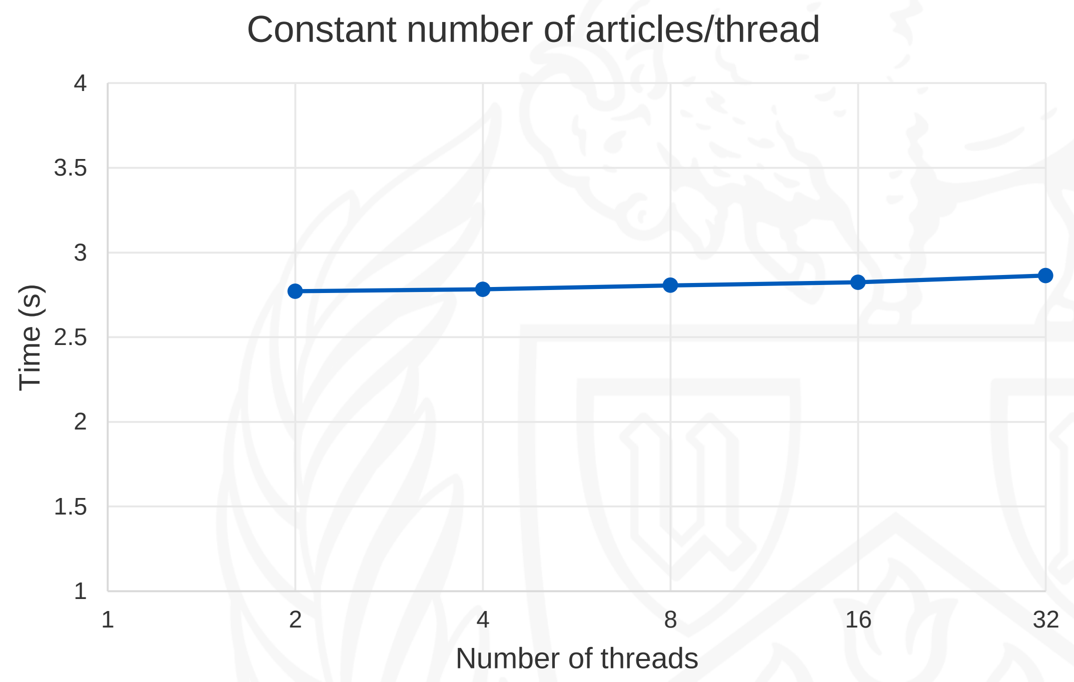
Total number of articles: 24576

Number of threads	Time (s)
2	229.6
4	115.02
8	57.612
16	29.12
32	14.99



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

Number of threads	Articles	Time(s)
2	330	2.771
4	660	2.782
8	1320	2.806
16	2640	2.824
32	5280	2.864



Observations

- For a constant number of articles, the computation time decreases as the number of threads increase.
- The speed up decreases as the number of threads increase, due to overhead caused by thread scheduling.
- When we maintain a constant number of articles/cores, the computation time remains comparable with slight increase as we increase the number of cores due to overhead introduced by thread scheduling.

University at Buffalo The State University of New York

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://github.com/classner/pymp
- https://www.kaggle.com/snapcrack/all-the-news

University at Buffalo The State University of New York

Thank You!