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

CSE 702(FALL 2020)
Instructor: Dr. Russ Miller

Name: Mansi Shetty
Overview

• Terminologies and Definitions
• Problem Statement
• Applications
• Sequential Algorithm
• Parallel Implementation
• Results
• Observation
• References
Terminologies and Definitions

**Document**

A document is a collection 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, $\text{idf}(t) = \log(N/df)$

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

$$tf-idf_{t,d} = tf_{t,d} \times idf_t$$

$$Score(q, d) = \sum_{t \in q} tf-idf_{t,d}$$
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),
  
  $tf_{this,d2} = 1/7$ , $idf_{this} = \log(2/2) = 0$
  
  $tf_{is,d2} = 1/7$ , $idf_{is} = \log(2/2) = 0$
  
  $tf_{another,d2} = 2/7$ , $idf_{another} = \log(2/1) = 0.301$
  
  $tf_{example,d2} = 3/7$ , $idf_{example} = \log(2/1) = 0.301$

Therefore,

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

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

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

$tf-idf_{example,d2} = (3/7) \times 0.301 = 0.129$
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.
RESULTS
Total number of articles: 6144

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>56.568</td>
</tr>
<tr>
<td>4</td>
<td>28.841</td>
</tr>
<tr>
<td>8</td>
<td>14.67</td>
</tr>
<tr>
<td>16</td>
<td>8.252</td>
</tr>
<tr>
<td>32</td>
<td>4.34</td>
</tr>
</tbody>
</table>
Total number of articles: 12288

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>114.19</td>
</tr>
<tr>
<td>4</td>
<td>57.223</td>
</tr>
<tr>
<td>8</td>
<td>28.534</td>
</tr>
<tr>
<td>16</td>
<td>14.47</td>
</tr>
<tr>
<td>32</td>
<td>7.418</td>
</tr>
</tbody>
</table>

Number of articles: 12288
Total number of articles: 24576

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>229.6</td>
</tr>
<tr>
<td>4</td>
<td>115.02</td>
</tr>
<tr>
<td>8</td>
<td>57.612</td>
</tr>
<tr>
<td>16</td>
<td>29.12</td>
</tr>
<tr>
<td>32</td>
<td>14.99</td>
</tr>
</tbody>
</table>
Constant number of articles/thread (165 articles/thread)

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Articles</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>330</td>
<td>2.771</td>
</tr>
<tr>
<td>4</td>
<td>660</td>
<td>2.782</td>
</tr>
<tr>
<td>8</td>
<td>1320</td>
<td>2.806</td>
</tr>
<tr>
<td>16</td>
<td>2640</td>
<td>2.824</td>
</tr>
<tr>
<td>32</td>
<td>5280</td>
<td>2.864</td>
</tr>
</tbody>
</table>

![Graph showing time in seconds vs number of threads for constant number of articles/thread]
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.
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://github.com/classner/pymp

• https://www.kaggle.com/snapcrack/all-the-news
Thank You!