Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring

Osama Haggag and Samhaa El-Beltagy
Center for Informatics Science,
Nile University,
Egypt
Outline

- Introduction
- Problem Description
- Task Description
- Implementation
- Results
- Conclusion
- Future Work
Task Description
Task Description

- We are given a plagiarized dataset
  - Plagiarized from the ClueWeb09 corpus
  - There's little to no obfuscation
  - Some passages and headlines are not plagiarized
  - Documents are well written, and punctuated
  - Documents are organized into paragraphs focusing on certain subtopics related to the larger topic at hand
The goal is to:
- Maximize and maintain a good balance in the retrieval performance
- Minimize workload and runtime

**The plan is to broaden the searching scope through topical segmentation**

**While introducing some form of search control in utilizing the queries**
- It would be favorable to score queries that haven’t been used yet against already downloaded documents

**The core of the problem is document downloads**
- Downloading irrelevant documents leads to more irrelevance
- Downloading relevant documents minimizes the search effort and sharpens precision
Implementation
The slight obfuscation was disregarded due its insignificance

ChatNoir is the search engine of choice

The system is made up of a number of phases
- Data preparation
- Query formulation
- Searching
- Tuning the parameters
Implementation

- Data Preparation:

  - Preprocessing
  - Tokenization
  - Frequency Distribution
  - Keyphrase Extraction
  - Sentence Tokenization
  - TextTiling

  - Document
    - Sent 1, sent 2, sent 3, sent 4, sent 5, sent 6, ...
    - Sent 1, sent 4, ...
    - Sent 3, sent 6, ...
    - Sent 2, sent 6, ...

  - Keyphrase 1, Keyphrase 2, Keyphrase 3, ...

  - Query Generation
    - “obama”: 23
      - “clan”: 1
    - “barack obama”
      - “michelle obama”
    - [s1, s4, s11, s13]
      - [s16, s19, s22, s25]
      - 4 sentence chunks
Query formulation:

- For each segment we have:
  - For each 4-sentence chunk:

Queries are stored as a list of strings per document

Query has to be < 10 keywords
Searching:
- Given a list of queries per document:

- Query_1
- Query_2
- Query_3
- ... 
- Query_n

Snippet

> 50%?

Skip to next Query

> 60%?

Consider document a source

Download

Add to list
Tuning the parameters:

- The system has a number of parameters that need tuning.
- Due to the time cost of an experiment over the dataset, difficult to optimize by iteration over combinations.
- We use human intuition, common sense, and a small number of experiments to determine values that are good enough, but not necessarily optimal.
Tuning the parameters (in processing):

- **TextTiling parameters:**
  - Control over size of subdocuments
  - Tuning for a large number of segments of small size gives higher recall
  - Tuning for a small number of large topics is best for both precision and recall
Tuning the parameters (in processing):

- Sentence chunk size selection:
  - A chunk size of 1, gives better recall at loss of precision
  - A chunk size of 4 is determined to do best

- Frequency threshold:
  - Identifies the “unique” words in the query
  - The threshold of 1 is chosen after running experiments
Implementation

- Tuning the parameters (for search):
  - Number of results returned:
    - First result is often the most relevant one
  - Query vs. Snippet score:
    - A score of 50% filtered search results nicely
    - Less meant higher recall, more meant less recall without equivalent improvement in precision
Implementation

- Tuning the parameters (for search):
  - Query vs. Candidate Document score:
    - Same rationale as scoring against snippets
    - 60% a relatively good filter
    - Higher values are better for recall
  - Refer to Tables 1, 2, 3 on page 6 in the paper for details
Results
Our system was evaluated using the measures set by PAN’13.
The system is determined to be one of the top three systems at PAN’13.

Results

<table>
<thead>
<tr>
<th>Retrieval Performance</th>
<th>Workload</th>
<th>1st Detection</th>
<th>No Detection</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Prec</td>
<td>Recall</td>
<td>Qrs</td>
<td>Dlds</td>
</tr>
<tr>
<td>Haggag</td>
<td>0.44</td>
<td>0.63</td>
<td>0.38</td>
<td>32.04</td>
</tr>
<tr>
<td>Williams</td>
<td>0.47</td>
<td>0.55</td>
<td>0.50</td>
<td>116.4</td>
</tr>
<tr>
<td>Lee</td>
<td>0.35</td>
<td>0.50</td>
<td>0.33</td>
<td>44.04</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

- We have a system that can retrieve possible plagiarism sources with competitive performance at minimal workload.
- This is done through careful formulation, and discriminative elimination of queries.
- The system employs two algorithms:
  - TextTiling: topical segmentation – Marti A. Hearst
  - KPMiner: keyphrase extraction – Samhaa R. El-Beltagy
Future Work

- There is room for improvement on the current system
  - Optimize the parameters
  - Make use of ChatNoir’s advanced search functions
- Investigate more about obfuscation
- More intelligence in the scoring functions
- The code to our implementation available on git-hub, under the MIT license