Information Retrieval: Assignment 2

Prof. Toon Calders, Ewoenam Tokpo

Deadline: 14/12/2020

For the second project, there are four proposed projects from which you can choose **one**. Besides these four projects, you can also contact us with your own project, related to the topics treated in the course so far, for approval. This project is to be executed in groups of 2 to 3 students. The project can be carried out with any programming language of choice. Python and Java will be recommended since they are more compatible with tools and frameworks like Apache Spark.

There will be a presentation session after completing the project where you will present your work to your colleagues. Hence, you should prepare a short power point presentation (maximum of 10 slides) that you can use to present the work you have done. This is a helpful exercise since you will be working on different projects related to different aspects of the course. This session will help you have an understanding and appreciation of the projects done by other students and the topics they relate to. More information regarding the presentation will be given later.

1 Page Rank

1.1 Introduction

The goal of this assignment is to implement a PageRank algorithm to rank webpages. The dataset for this project is a webgraph that mimics the structure of the web on a small scale.

The objective of this project is to help you understand the underlying structure, logic, and math behind some algorithms that rank web pages on the internet, particularly the PageRank algorithm. After completing the project, you should be able to implement a basic ranking system for a network of web pages.

1.2 Dataset

The dataset for this project is a webgraph consisting of 50 million nodes. This is extracted from a bigger webgraph, thus, some of the nodes will have outgoing links to nodes outside the 50 million nodes. In total, there are 428,136,613 modes if you include unique nodes for all the outgoing links for the 50 million nodes.

The first line in the file is the total number of nodes (428,136,613). Each line afterward represents each unique node ID; starting from 0 to 50 million. Since the webgraph primarily concerns the first 50 million nodes, the nodes outside the 50 million nodes will not have actual page content associated with it; there will be 0 outlink for those nodes.

The content of each line corresponding to a particular node ID are its outgoing links (target nodes), which are space-separated. If a node does not have any outlinks associated with it, the line will be blank. The data can be download directly from here or from the homesite, lemurproject.org → TREC Category B → ClueWeb09_WG_50m.graph-txt.gz. In addition to

this, the list of URLs corresponding to the 50 million node Ids can be downloaded here; this can also be found on the homesite. Given that the size of the URL list file is huge, it is not necessary to include it in the project; using just the node Ids is fine.

You can consider smaller webgraphs such as Google web graph in case of memory constraints, however, note that regard will be given to the size of the dataset during evaluation. You are also at liberty to use your own dataset for the project. The dataset should, however, be of a suitable size and structure. To be sure, you can contact us with the description of the dataset.

1.3 Assignment

The project should include the following basic steps:

1. Using a distributed scheme that employs map-reduce operations, generate the Page Rank score for each node (only the 50 million nodes). Let α be the teleportation probability (probability of jumping to a random node). Take α to be 0.15. Apache Spark can be used to implement the map-reduce operations for this task. You can find some useful materials and resources for Apache Spark/PySpark in the Appendix.

2. Rank the node Ids together with their scores in descending order of importance based on their PageRank values. Save this as pagerank_list.csv file which should be included in the GitHub repository.

3. Indicate in the report the number of iterations needed for convergence. Convergence is achieved when the change in value between two successive iterations for all nodes is less than ϵ. Take ϵ to be 10^{-6}

4. Experiment with $\alpha = 0.10$ and $\alpha = 0.50$. Discuss in the report the effects of the different α values ion the system

Please note that the scope of the project is not limited to the above steps; these are just the core aspects. For instance, you can also propose, discuss or develop an improved ranking algorithm that improves on the efficiency and performance of existing algorithms.

2 Latent Semantic Indexing

2.1 Introduction

The goal of this assignment is to implement an LSI system for indexing documents. You will be given a corpus of text documents for which you will index using LSI.

The objective of this project is to help you understand how LSI works, the math behind it, and how to effectively implement it for real-world use. After completing the project, you should be able to develop a good document indexing system with LSI.

2.2 Dataset

The dataset for this project is a text corpus of news items collected from some mainstream news outlets. The CSV file contains 141585 rows each representing a news article. This is an extract from the larger dataset from components.one. The dataset can be directly downloaded here. Alternatively, you can use the complete dataset consisting of 2.7 million news articles from components.

The first row of the CSV file contains the heading for each column. There are 9 columns in the extracted version and 10 columns in the original version. For the project, you basically need only the data from the **content** column which contains the main news content and the **id** column which contains the identifiers for each news article. You can use the remaining columns if you find them interesting to use for further tasks or explorations. Here again, you are at liberty to use your own dataset for the project.

2.3 Assignment

The project should fulfill the following basic steps:

1. Preprocessing of data. You are to try out different preprocessing techniques that will improve the efficiency of the algorithm. This could involve stopword removal, case-folding, lemmatization, etc.
2. Calculation of tf-idf scores and the creation of a term-document matrix using these scores.
3. Decomposition of the term-document matrix using SVD to obtain term matrix and document matrix. Take the value of k to be 200. Where k is the number of topics.
4. Experiment with $k = 100$ and $k = 300$ and discuss in the report the effects of the different k values on document retrieval.
5. Explore possible techniques to pick an optimal k value.
6. Use the titles of the last 20 news articles as queries and rank the top 100 results for each query (only for $k = 200$). Save this list as LSI_rank.csv file which should be included in the GitHub repository. Each column in the CSV file should represent a query. Hence, the first line/row of the CSV file would contain the respective query_number/id for each column (Make a list of the query ids with its respective query in the report; to help us know the exact query for each id). The remaining lines will contain the results in decreasing order for each query. If you use a dataset considerably different from the one given, you will have to generate your own queries for this part.

There are two options to implement this project since there are preexisting libraries that can implement many of the above tasks. You can implement the tasks from scratch which will be a huge plus to your work, or you can use existing libraries in which case you must carry out extra tasks such as cluster analysis of the documents from your implementation, analyzing word association in the text corpus, evaluating the effectiveness of the system in identifying term relations (polysemy and synonymy) etc.

3 Topic Modeling

3.1 Introduction

Due to the growth of data and information such as news articles on the internet, it has become useful to sort such articles according to topics such that readers can easily pick out articles related to their topics of interest. Topic models serve as important tools to achieve this. For this project, you will be using the dataset of news articles to implement a topic modeling system that can categorize news items according to topics.

The objective of this project is to help you understand how topic models like Latent Dirichlet Allocation work and how to effectively implement them for real-world scenarios. After completing the project, you should be equipped to implement a basic topic model for text articles.

3.2 Dataset

The dataset for this project is the same as that of LSI. Please refer to the Dataset section of the LSI project for details about the dataset. Again, you are at liberty to use your own dataset for the project.

3.3 Assignment

The project should fulfill the following basic steps:

1. Preprocessing of data. You are to try out different preprocessing techniques that will improve the efficiency of the algorithm. This could involve stopword removal, case-folding, lemmatization, etc.
2. Implementation of a topic model such as LDA. Take the number of topics k to be 20
3. Generate the list of topics from your implementation
4. Experiment with $k = 10$ and $k = 50$ and discuss in the report the effects of the different k values.
5. Explore and evaluate possible techniques to pick an optimal k value.
6. For each topic, find the top 100 document that best represents it; that have the highest score for the topic (only for $k = 20$). Save this list as topic_document_rank.csv file which should be included in the GitHub repository. Each column in the CSV file should represent a topic. Hence, the first line/row of the CSV file would contain the respective topic_ids for each column (Make a list of the topic ids with its respective topic in the report; to help us know the exact topics for each id). The remaining lines will contain the results in decreasing order for each topic.

You again have two options to implement this project since there are preexisting libraries that can implement many of the above tasks. You can implement the models from scratch which will be a huge plus to your work, or you can use existing libraries in which case you must carry out extra tasks such as analyzing the connection/co-occurrence between topics, finding the most popular news topics for each month, cluster analysis, etc. You can also propose, discuss, or develop an improved topic model.

4 Index Construction and Compression

4.1 Introduction

Another challenge that has arisen from the massive growth in data is the need to find efficient index construction and compression techniques that allow data to fit within the constraints of available hardware and to allow fast and effective information retrieval. In this project, your task will be to implement an effective Index Construction and Compression technique to efficiently index large sizes of data.

The objective here is to have hands-on experience with Index Construction and Compression techniques such that after completing the project, you should be equipped to implement an efficient topic model for text documents.

4.2 Dataset

The dataset for this project is the same as that of LSI. Please refer to the Dataset section of the LSI project for details about the dataset. Here again, you are at liberty to use your own dataset for the project.

4.3 Assignment

1. Preprocessing of data. You are to try out different preprocessing techniques that will improve the efficiency of the algorithm. This could involve stopword removal, case-folding, lemmatization, etc.

2. Implementation of an effective distributed index construction technique for the dataset. For this, you can use Apache Spark/PySpark to implement the map-reduce operations. See appendix for some useful resources on Spark.

3. Implementation of an effective dictionary and posting compression scheme for the data

4. Evaluate the space complexity of your implementation.

5 Deliverables

1. The code of your project. Please do not include bulky software libraries or large datasets in emails. The preferred way to share code is via a link to a publicly available GitHub repository. Include the link in your report. The files requested in the assignment description should be included in the GitHub repository.

2. The report in **PDF format**, to be submitted via BlackBoard. Do not submit zip-files, word documents etc., only the submission of a single PDF file will be accepted. The report should be approximately 10 pages in length.

   The report should include the following:

   - Description of selected project and background (concepts and techniques) information of its related topic.
   - Details of your implementation; algorithms, frameworks, environment, etc.
   - Analysis and evaluation of the work.

A Note on Plagiarism

There is absolutely nothing wrong with using existing materials, you will even be commended for not reinventing the wheel, as long as you are not violating the copyright of other authors. Nevertheless, it is expected from you to clearly indicate whenever you used material that was not created by yourself. Clearly indicate in your submissions which parts constitute original work, which parts are taken from other works, and which parts were adapted from external sources. These sources have to be properly acknowledged in all your submissions. Concretely, this means at least the following guidelines are observed:

- Papers, books, webpages, blogs, etc. that were inspected while making the assignment will be referenced in a separate section “References”. Citations to these materials are included in the text where appropriate.

- Text fragments exceeding one sentence that are copied from other sources are clearly marked as such. You could for instance include quoted text, definitions, etc. in italics, followed by a reference. An example of how to do this: Bela Gip (2014) defines plagiarism as *"The use of ideas, concepts, words, or structures without appropriately acknowledging the source to benefit in a setting where originality is expected"*

  References: (at the end of the document) Gipp, Bela. Citation-based plagiarism detection. Springer Vieweg, Wiesbaden, 2014. 57-88.

- When using code from other sources, indicate so in the report, and in the source code. This could for instance be done by adding a comment with a reference to the source of the function for each function that was copied from another source. It is recommended to include a separate folder "sources" in your GitHub repository with the original files from other authors that you used. Include source in the message of your commits.

6 Appendix

- Spark is an general-purpose open-source distributed data processing framework. It supports various programming languages such as Java, Scala, Python and R. You can install and run it locally on your laptops or use a cloud environment such as databricks community which is free to use. You can get more details about databricks community here.
- Official Spark documentation
- Spark/pyspark installation guide
- Some other useful Spark materials at tutorialspoint Spark and tutorialspoint PySpark
- A very useful book on spark: Mastering Large Datasets with Python: Parallelize and Distribute Your Python Code, by John T Wolohan