Information Retrieval: Assignment 2

Prof. Toon Calders, Ewoenam Tokpo
{toon.calders, ewoenamkwaku.tokpo}@uantwerpen.be

Deadline: 13/12/2021

This project is to be executed in groups of 2 students. For further inquiries about the project, please email ewoenamkwaku.tokpo@uantwerpen.be. Each group will present their project to the other groups. More information such as the schedule and whether presentations will be online or on campus will be communicated later.

1 Introduction

ArchiveIR is a company that runs an online archive with search and retrieval functionalities that users can use to quickly retrieve documents. Over the years, Lucene has been the core of their system. This system has proven effective over the years, but, due to customer dissatisfaction and complaints about the quality of search results, ArchiveIR wants to try out new technologies to further filter the initial search results. One key problem users complain about is the system's ineffectiveness in handling synonymy (different words with the same meaning), and polysemy (one word with multiple meanings). Words such as *car* and *vehicle* are considered to be very different by the system. On the other hand, words such like *bank* in *river bank* and *investment bank* are considered to be the same by the system, although due to the context in which they appear, they might refer to vastly different concepts.

Your team has been tasked to use **one** of two approaches described below to build a model that can be used to refine the initial search results retrieved by Lucene. To do this you have been provided with a record of past search queries and results. For each result (returned text documents), users have ranked the retrieved documents from most relevant to least relevant. In addition to this, users were asked to explicitly label documents they found to be relevant and irrelevant to their search with a binary annotation: 1 for relevant, 0 for irrelevant. For instance, in Fig 1, each *doc_number* corresponds to a retrieved document for query 1089071, all ranked in descending order (doc 432658 is the most relevant). Additional label information has been provided in the *label* column to indicate relevant documents .

2 Project 1: Dimensionality reduction

2.1 Introduction

The first approach is to consider dimensionality techniques. In addition to reducing the size of document representations, dimensionality reduction techniques are also able to handle issues such as synonymy and uncovering hidden topics in a text corpus.

Here, you are to explore the use of a dimensionality reduction technique to rank documents retrieved by Lucene. That is: in order to avoid the complexity of having to do a full nearest

Query_number	doc_number	Query	doc_text	label
1089071	432658	va death benefits contact number	Veteran Benefits in Nebraska Ads You May Be In...	1
1089071	417115	va death benefits contact number	Home › Benefit Library › State / Territory Ben...	1
1089071	429474	va death benefits contact number	VA Burial Benefits Burial Allowances Burial, H...	1
1089071	122086	va death benefits contact number	Home › Benefit Library › Federal Benefits › Bu...	1
1089071	60461	va death benefits contact number	Call Today1-844-VET-LAWS (838-5297)REFER A FRI...	1
1089071	476602	va death benefits contact number	Death after Retirement Non Service-Connected D...	1
1089071	362869	va death benefits contact number	Veteran Benefits In Arkansas Ads You May Be In...	1
1089071	128382	va death benefits contact number	V. A. Burial Benefits Find Out What Burial Cos...	1
1089071	346632	va death benefits contact number	What GAO Found The number of older veterans re...	0
1089071	29215	va death benefits contact number	TRICAREPublished January 05, 2016PRINT | E-MAI...	0
1089071	467667	va death benefits contact number	Home > Student Services > Admissions & Records...	0
1089071	501419	va death benefits contact number	VA Mortgage Loan Checklist Posted on: June 18,...	0
1089071	429264	va death benefits contact number	Sleep Apnea in Veterans – VA Benefits Are Avai...	0

Figure 1: dataset

neighbor search between the reduced query vector with each and every reduced document vector, we will only re-rank the top-10 results returned by Lucene using the dimensionality reduction technique. You can choose to use one of the techniques for dimensionality reduction covered in the course: LSI or word embeddings, or compare the performance of both.

Evaluate the performance gain (if any) when applying dimensionality reduction.

2.2 Dataset

The full corpus of documents in the archive is available to generate the embeddings or transformation matrix.

The *dev datasets* are available to evaluate your implementation during development and to run your experiments. They consist of the following:

- dev queries: Queries to probe the system.
- dev Lucene retreivals: This is the ranked (in descending order of relevance) output of documents using Lucene.
- dev ground truth rankings: This is the desired rankings of the documents for each query (also in descending order of relevance)

Finally, the test dataset will be used to produce the output for your final submission. The test dataset contains queries and associated texts that are in no order of relevance. And the task will de to re-rank them.

2.3 Project specifications

1. The project should among other things fulfill the following basic goals:

(a) An implementation of a dimensionality reduction method. The goal is not to implement the entire system from scratch. Existing libraries can be used for the implementation. The full corpus of documents in the archive is available to generate the embeddings or transformation matrix.

(b) Using the dev datasets (*dev queries, dev Lucene retreivals, dev ground truth rankings*):

i. Evaluate the performance gain (if any) when applying dimensionality reduction: Using your implementation, reorder the retrieved documents from dev Lucene retreivals in descending order of relevance. Use dev ground truth rankings as the ground truth ranking to see if your implementation improves on Lucene's retrieval ranking.

ii. Discuss how preprocessing techniques such as stop-word removal, lemmatization, etc. affect the performance of the system. For LSI, you can also examine how the choice of term-document matrix representation (one-hot encoding or TF-IDF) affects the retrieval performance.

iii. Discuss how different values of k affect the performance of the system (where k is the size of the dimension from the decomposed matrix). How will you determine the right value of k for your implementation to ensure the best performance? What will be the optimum k value based on the training dataset?

(c) Using the optimum value of k from the previous step, rank the top 10 results for each query in the test data.csv. Save this list as *dimensionality_reduction_rank.csv* file which should be included in the GitHub repository. This file should be in the same format as format example.

2. A report describing the details of the project. The report should cover the following aspects of the project:

(a) Background details on the dimensionality reduction approach used.

(b) The main or essential components of the technique.

(c) Implementation details of the various components.

(d) Experimental analysis done. E.g., experiments that were done to select parameter values for your implementation. Experiments and discussions including visual plots are very much encouraged

(e) Analysis and discussion of the final results of your implementation.

(f) The link to the GitHub repository of your implementation should be included in the report.

3. A presentation of the project. The level of technicality and difficulty of your presentation should be such that it is suitable for the other students in the course; that is: you can assume the audience knows concepts like TFIDF, precision, recall, ... but they do not necessarily have a good idea how the data needed to be prepared for the dimensionality reduction, and how exactly it was applied. The presentation should cover the following:

(a) The conceptual idea behind your approach.

(b) The implementation steps of the approach.

(c) Experimental and final results from the implementation.

2.4 Useful resources

Below are some useful resources to get started:

1. Tutorial on Latent Semantic Analysis using Python: datacamp - discovering-hidden-topics-python
2. Information Retrieval using word2vec; https://www.analyticsvidhya.com/blog/2020/08/information-retrieval-using-word2vec-based-vector-space-model/

3 Project 2: Neural retrieval model - Document ranking

3.1 Introduction

Neural networks, although relatively new to the field of information retrieval, have exhibited impressive performance in information retrieval tasks. Transformer-based models in particular have stood out for such tasks. BERT Devlin et al. [2018], has grown in popularity in Natural Language Processing and has been successfully applied for information retrieval.

For this assignment, you are required to build a document relevance detection and ranking system for information retrieval using any neural architecture. Since BERT has been a preferred choice for such tasks, it is highly recommended to consider retrieval models based on BERT.

Apart from the type of technique applied, the setting and both the dev and test data are the same as for the dimensionality reduction approach.

3.2 Dataset

The Training dataset is a collection of past queries that were retrieved with Lucene. For each query, users were asked to label the relevance of the document (1 for relevant, 0 for irrelevant) and rank the documents in descending order of relevance. Each row is made up of a query-document pair. The file consists of 5 columns: Query_number, Query, Text, doc_number, label.

- Query_number: Identifier for each query
- Query: Text query
- doc_number: Identifier for each text
- Text: Text content for the document.
- label: Relevance label of a text with respect to the corresponding query: {0:irrelevant, 1:relevant}.

The dev datasets are available to evaluate your implementation during development and to run your experiments. They consist of the following:

- all documents: Entire list of documents.
- dev queries: Queries to probe the system.
- dev Lucene retreivals: This is the ranked (in descending order of relevance) output of documents using Lucene.
- dev ground truth: This is the desired rankings and labels of the documents for each query (also in descending order of relevance)

To grade the performance of the system, the test dataset will be used to produce the output for your final submission. The test dataset contains queries and associated texts (the texts are in no order of relevance) and no labels.

3.3 Project specifications

1. Build and train a neural model that predicts whether a particular document is relevant for a particular query. Given a set of retrieved documents by Lucene, the model should be able to filter out False positives from the initial retrieval.
2. The model should be able to generate scores that can be used to rank the retrieved documents in order of relevance.
3. Using the dev datasets (*dev queries, dev Lucene retrievals, dev ground truth*):
   (a) Evaluate the performance gain (if any)
   (b) Find an effective strategy to select the best hyper-parameter values and how these values impact the performance of the retrieval system.
4. Run the model on the test dataset:
   (a) For each query-document pair in the *test dataset*, we first want to predict whether the document is relevant to the query. Output label should be 1 for relevant and 0 for irrelevant.
   (b) For each query, rank its retrieved documents in descending order of relevance. You can use the relevance probability score (from the prediction task in the previous step) for this.
   (c) Save this file as *neural_model_rank.csv* which should be included in the GitHub repository. This file should be in the format used for the *dev ground truth* dataset.
5. A report to describe details of the project. The report should cover the following aspects of the project:
   (a) Background details on the neural model or approach used.
   (b) The key conceptual components of the project.
   (c) Implementation details of the various components.
   (d) Experimental analysis done. eg. Experiments done to select hyperparameter values for your implementation.
   (e) Analysis and discussion of the final results of your implementation.
   (f) The link to the GitHub repository of your implementation should be included in the blog.

   A presentation of the project. The level of technicality and difficulty of your presentation should be such that it is suitable for the other students in the course; that is: you can assume the audience knows concepts like TFIDF, precision, recall, ... but they do not necessarily have a good idea of how the neural model was trained, and how exactly it was applied. The presentation should cover the following:
   (a) The conceptual idea behind your approach.
   (b) The implementation steps of the approach.
   (c) Experimental and final results from the implementation.

3.4 Useful resources

To help get started, here are some useful resources to possible techniques to explore:

1. Tutorial on semantic search with S-BERT: https://medium.com/mlearning-ai/semantic-search-with-s-bert-is-all-you-need-951bc710e160
2. Tutorial on text entailment with BERT: https://towardsdatascience.com/fine-tuning-pre-trained-transformer-models-for-sentence-entailment-d87caf9ec9db

4 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 blog. The files requested in the assignment description should be included in the GitHub repository.
2. A presentation. Please upload the slides to Blackboard before your presentation.
3. 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.

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.

References

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.