Bilkent University

CS433/533: Information Retrieval Systems
Assignment No. 1: Implementation of an IR System Using Word Embeddings
October 21, 2024
Due date/time: November 22, 2024

Notes: 1. You will upload your programs a report in pdf form (prepared by latex or Word). Handwritten submissions get 0 (no excuses). On Moodle course website necessary submission interface will be available. **2**. Submission deadline is sharp and November 22, 23:59. **3**. You can do late submission but for each late day 10 points will be subtracted. You can be at the most three days late. For example, if you submit on November 23 at midnight (one minute late) your assignment will be graded out of 90. The same applies to late days. **4**. It is not a group project. **5**. In your report and programs give **a**. Student name. **b**. Course number that applies to individual students. **c**. Email address of student. **d**. Assignment no. **e**. Your report should have proper title and good presentation. **f.** Number your pages and likewise number any table or figure you include. **5**. A plagiarism check will be performed on the code and the report.

If you have questions you can directly write the TA Pouya Ghahramanian (Ghahramanian@bilkent.edu.tr).

Overview

In this assignment, you will implement an information retrieval system using a combination of traditional retrieval methods and word embeddings. Specifically, you will be integrating TF-IDF and BM25 ranking methods with FastText embeddings. Your task involves three different cases for using FastText: (1) training a FastText model from scratch on the CISI dataset, (2) using a pre-trained FastText model, and (3) fine-tuning a pre-trained FastText model on the CISI dataset. You will rank documents for the given queries based on a combination of their traditional retrieval scores and the similarity between their embedding vectors. The performance of your retrieval system will be evaluated using the Mean Average Precision (MAP) metric.

Concepts

- **FastText:** FastText is a word embedding method that extends Word2Vec by representing words as bags of character n-grams, allowing it to capture subword information. It generates word embeddings by considering the internal structure of words, which makes it effective at handling rare words and misspellings. FastText can be trained from scratch or fine-tuned on specific datasets, and pre-trained models are also available for various languages.
- **TF-IDF (Term Frequency-Inverse Document Frequency):** TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. This helps to give weight to more discriminative terms in a document.
- **BM25:** BM25 is a modern ranking function used by search engines to rank matching documents according to their relevance to a given search query. It builds upon the basic TF-IDF principle but introduces two additional parameters that control the term frequency saturation and the document length normalization. This makes BM25 more adaptable and often more effective than plain TF-IDF in ranking documents.
- **CISI Dataset:** CISI is an information retrieval dataset that contains 1,460 abstracts and 112 queries. You can access the dataset in CSV format at the following link: https://github.com/pouyaghahramanian/CISI

Requirements

You will need the following packages in Python to complete this assignment.

- pandas: For data manipulation
- fasttext: For downloading and using pretrained Fasttext models
- gensim: For training and fine-tuning Fasttext models
- numpy: For numerical computing
- sklearn: For machine learning tasks
- rank_bm25: For calculating BM25 scores

Part 1: Dataset Preprocessing

1. Load the CISI dataset from the provided CSV files. The dataset consists of:
    - Documents: Represented by document id, title, and text.
    - Queries: List of natural language questions represented by query ids and texts.
    - Ground Truth: Contains relevance judgments for query-document pairs.

2. Preprocess the dataset. This may include tokenizing the text, removing stopwords and punctuation marks, and lowercasing the text. Explain your preprocessing steps.

Part 2: Word Embedding

1. **Training FastText:** Use the gensim library to train a FastText model on the CISI dataset. Ensure that both the text from the queries and the documents are used for the training process. This approach will help the model learn the specific vocabulary and context present in the dataset.
2. **Using a Pre-trained FastText Model:** Instead of starting from scratch, you can leverage pre-existing knowledge by using a pre-trained FastText model. Download a pre-trained FastText model, such as “cc.en.300.bin” (English), from the FastText website or using the fasttext library. The pre-trained model can be loaded using the gensim library for direct use in your retrieval system.
3. **Fine-tuning a Pre-trained FastText Model:** Further improve a pre-trained FastText model by fine-tuning it on the CISI dataset. This involves continuing the training process using the text from both the queries and the documents, allowing the model to adapt and better capture the characteristics and domain-specific vocabulary of the dataset.

Part 3: Retrieval Task

In this part, you will focus on the core of information retrieval: fetching relevant documents based on a user's query. The challenge lies in effectively combining traditional retrieval methods with modern word embeddings to enhance the accuracy of the results.

1. **Embedding Computation:** Implement a method to derive the embedding for an input (a query or a document) by averaging the embeddings of its constituent words.
2. **Retrieval System:** Construct a retrieval system that:

- Retrieves the top 10 documents for a given query.
- Combines scores from TF-IDF, BM25, and word embeddings to rank documents. You can use TfidfVectorizer from sklearn.feature_extraction.text, and BM25Okapi from rank_bm25 to obtain the TF-IDF and BM25 scores. Use cosine similarity to calculate the embedding scores. The overall score for a query-document pair is calculated as the weighted average of the TF-IDF, BM25, and embedding scores.
- Allows for the utilization of three FastText models: (1) the FastText model you trained from scratch on the CISI dataset, (2) a pre-trained FastText model, and (3) the fine-tuned version of the pre-trained model using the CISI dataset.

- Provides the flexibility to set weights for TF-IDF, BM25, and embedding scores to compute a combined score.

Part 4: Evaluation

Evaluation is crucial in information retrieval to ensure that the system meets user expectations. In this part, you will assess the performance of your retrieval system under various configurations. This includes experimenting with different possibilities of combining FastText embeddings with traditional TF-IDF and BM25 methods, as well as using each of them in isolation.

- **Performance Metrics:** Evaluate the efficacy of your retrieval system using the Mean Average Precision (MAP) metric.
- **Comparative Analysis:** Contrast the performance metrics when employing the three FastText models: (1) the FastText model trained from scratch on the CISI dataset, (2) the pre-trained FastText model, and (3) the fine-tuned FastText model. Analyze the potential reasons for observed differences or similarities in the results.
- **Experimentation:** Experiment with different combinations of weights for TF-IDF, BM25, and embedding scores. Also, test the performance of each method in isolation. Report your observations and the MAP scores for each scenario.

Report

In your report, detail your observations from the experiments. Analyze the results from different retrieval configurations, including the impact of varying weights for TF-IDF, BM25, and FastText embeddings. Discuss the performance differences among the three FastText models (trained from scratch, pre-trained, and fine-tuned). Provide insights into the observed trends and their implications in information retrieval.