COL 764 Assignment 2: Document Reranking Task

Rohan Chaturvedi
2022MT11262

September 25, 2024

Abstract

This report presents the implementation and evaluation of three reranking models: a Local Word2Vec model, a Generic Word2Vec model, and a Generic GloVe model for the document reranking task. The objective is to improve precision and recall by expanding queries using pseudo-relevance feedback and word embeddings. We evaluate the models using nDCG metrics and explore hyperparameter tuning to achieve optimal performance.

1 Introduction

This report details the reranking task implemented as part of Assignment 2 for COL 764. The goal is to improve the ranking of top-100 documents retrieved for a query using pseudo-relevance feedback and word embeddings for query expansion. The task involves implementing three models:

- Local Word2Vec Model
- Generic Word2Vec Model
- Generic GloVe Model

We evaluate these models based on their performance using the TREC dataset and calculate nDCG@5, nDCG@10, and nDCG@50.

2 Data Description

The dataset consists of 3,213,835 unique documents and 24 queries from the TREC benchmark. Each query has a corresponding set of top-100 documents that need to be reranked. The relevance judgments for each query are provided in the qrels file, containing graded relevance levels.

3 Task 0: Relevance Model for Retrieval

We first implemented a language modeling-based retrieval model with Dirichlet smoothing as the base ranking model. The Dirichlet smoothing parameter (μ) was tuned to balance document length normalization and query likelihood.

3.1 Implementation Details

This describes the implementation of a language model that utilizes Dirichlet smoothing for document retrieval tasks. The model has been tested with various tokenization strategies, including **lemmatization, tokenization, and stopword removal.** Based on empirical evaluation, only stopword removal has been used in the final implementation, as it resulted in overall better retrieval performance. The corpus model is built on 100 top-retrieved documents combined with another randomly chosen set of 100 documents, providing a balanced dataset for training.

3.1.1 Corpus Language Model

The Corpus Language Model named as *Global Language Model* estimates the global probabilities of terms across the entire document collection. It tracks the frequency of each term in the collection and the total number of tokens. The global term probability is computed as the relative frequency of a term w in the entire corpus C:

$$P(w|C) = \frac{\text{count}(w, C)}{\sum_{w' \in C} \text{count}(w', C)}$$

where $\text{count}(w, C)$ represents the count of term w in the corpus C, and the denominator is the total number of terms in C.

3.1.2 Local Language Model with Dirichlet Smoothing

The *Local Language Model* is used to estimate term probabilities within individual documents. Dirichlet smoothing is applied to combine local term probabilities with global probabilities, balancing between the specific document statistics and the general distribution of terms across the corpus.

For a query term w, the probability of w given a document d is smoothed using the following formula:

$$P(w|d) = \frac{c(w, d) + \mu P(w|C)}{|d| + \mu}$$

where:

- $c(w, d)$ is the count of term w in document d,

- μ is the Dirichlet smoothing parameter (set to 500 in our implementation),
- $P(w|C)$ is the global probability of term w in the corpus C,
- $|d|$ is the total number of terms in document d.

3.1.3 Data Preparation

The corpus used for training consists of 100 documents from the top-100 retrieved set along with another 100 randomly chosen documents. This selection is designed to leverage pseudo-relevance feedback while ensuring that the model is trained on a diverse set of documents.

During preprocessing, stopword removal was found to significantly enhance the model's performance. Consequently, common stopwords were excluded from the tokenization process, and the focus was placed on more informative terms that are likely to impact document ranking.

3.1.4 Probabilistic Ranking

The language model assigns a smoothed probability to each query-document pair by combining the local document term frequency with the global term distribution. Given a query composed of several terms, the log-probabilities of individual terms are summed to compute the document's relevance score:

$$\log P(Q|d) = \sum_{w \in Q} \log P(w|d)$$

The query terms are weighted based on their frequency in the query, and only terms with non-zero probabilities contribute to the final score. The final document ranking is determined by the combined smoothed probabilities for all query terms.

3.2 Parameter Choices

The following values for μ were tested, along with their corresponding nDCG scores at different cutoff points (nDCG@5, nDCG@10, nDCG@50, nDCG@100):

μ	**nDCG@5**	**nDCG@10**	**nDCG@50**	**nDCG@100**
50	0.4152	0.4487	0.5709	0.6596
100	0.4241	0.4393	0.5738	0.6591
200	0.4371	0.4530	0.5719	0.6657
300	0.4374	0.4649	0.5747	0.6706
400	0.4467	0.4631	0.5734	0.6697
700	0.4586	0.4779	0.5741	0.6703
1000	0.4568	0.4797	0.5813	0.6754
1500	0.4553	0.4810	0.5817	0.6747
2000	0.4495	0.4634	0.5758	0.6674

Table 1: nDCG Scores for Different Values of μ

The optimal value of μ was found to be around 500, providing the highest average nDCG scores.

3.3 Effect of Stopword Removal, Stemming, and Lemmatization

The following table shows the nDCG scores for different combinations of stopword removal, stemming, and lemmatization, giving the reason for the choice:

Stopwords	**Stemming**	**Lemmatize**	**nDCG@5**	**nDCG@10**	**nDCG@50**	**nDCG@100**
False	False	False	0.3670	0.3884	0.5076	0.6273
True	False	False	0.4464	0.4652	0.5741	0.6691
True	True	False	0.4119	0.4506	0.5550	0.6597
True	False	True	0.4257	0.4592	0.5729	0.6694

Table 2: nDCG Scores for Different Combinations of Stopwords, Stemming, and Lemmatization

4 Task 1: Local Word2Vec Reranking

This task involved using locally trained word embeddings on the top-k documents retrieved for each query. The Word2Vec model was trained using the SkipGram architecture on the pseudo-relevance set.

In this document, we explain the implementation of a query expansion and document reranking process using a Word2Vec model for local word embeddings and KL divergence for document reranking. The process is designed to improve retrieval performance by expanding queries with semantically related terms and re-ranking documents based on the expanded queries.

4.1 Algorithm

The following steps outline the key stages of the query expansion and reranking system:

1. Extract the top-k documents for each query based on initial retrieval results.
2. Train a Word2Vec model on the extracted document set.
3. Find the top-20 nearest words to the query terms using the trained Word2Vec model.
4. Expand the query using these nearest words.
5. Rerank the top-100 documents using the expanded query and KL divergence for scoring.

4.2 Training the Word2Vec Model

The training of the Word2Vec model is performed on a subset of the document collection, specifically the top-k documents retrieved for each query. The following steps summarize the training process based on the code:

1. **Document Processing:** The model first processes each document by extracting the relevant sections (typically, sections of interest related to the query) and then passes them through the text processor, which performs tokenization and filtering (e.g., stopword removal).
2. **Sentence Formation:** The processed sections from the document are collected as sentences, with each sentence consisting of a list of words.
3. **Training the Model:** Once the sentences are collected, a Word2Vec model is trained on these sentences. The Word2Vec model learns distributed representations (embeddings) for each word in the document set. The key hyperparameters used in this process include:
    - `vector_size`: Dimensionality of the word vectors (set to 50 in this case due to much smaller corpus and best performance on it).
    - `window`: The context window size for words (set to 10 to capture large difference between models).
    - `min_count`: Minimum frequency for a word to be included in the vocabulary (set to 60 to avoid training words which favour only a particular document).
    - `workers`: Number of CPU threads to use for parallelization.

4.3 Query Expansion

Once the Word2Vec model is trained, it is used to expand the original query by finding semantically related terms. The expansion process is as follows:

1. **Query Processing:** The original query is passed through the same text processor to extract the terms.
2. **Finding Nearest Neighbors:** For each query term that exists in the vocabulary of the Word2Vec model, the top-10 most similar words are retrieved using the `most_similar` function. These similar words are ranked by their cosine similarity to the query terms in the embedding space.
3. **Query Expansion:** The top-20 similar words are selected and added to the original query. A weight is assigned to each term based on its similarity score. Only terms that appear in the language model's vocabulary and are not already part of the original query are included in the expansion.

4.4 KL Divergence-Based Reranking

After query expansion, the system uses Kullback-Leibler (KL) divergence to rerank the documents. KL divergence measures the difference between the query's language model and the document's language model. The goal is to rank documents that minimize the divergence from the query model.

The reranking process works as follows:

1. **Document Scoring:** For each document, its language model is compared to the expanded query model. The probability distribution of terms in the query and document is computed using Dirichlet smoothing.
2. **KL Divergence Calculation:** KL divergence is calculated for each document, and documents with smaller divergence scores are considered more relevant.

$$\sum_{w} \mathrm{P}(w|R) \log \mathrm{P}(w|D)$$

Here, $P(w|Q)$ represents the probability of word w in the expanded query, and $P(w|D)$ represents the probability of word w in the document D. Documents are then ranked based on the total divergence score.

4.5 Implementation

The implementation is encapsulated within the `QueryExpander` class, which handles both training and query expansion. Key methods include:

- `train()`: This method trains a Word2Vec model on a set of documents. It processes each document using the text processor, which removes stopwords and tokenizes the text, and then collects sentences to train the model.
- `query_expansion()`: This method expands the original query by finding similar terms using the trained Word2Vec model. The expanded query is then passed to the language model for document reranking.
- `generate_results()`: This method calculates document scores based on KL divergence and ranks the documents accordingly.

5 Task 2: Generic Word2Vec and GloVe Reranking

This task implements query expansion using pre-trained word embeddings from Word2Vec and GloVe models. The main goal is to enhance the query's semantic representation and improve document reranking by expanding the initial query with terms similar to those in the pre-trained embedding spaces.

5.1 Model Loading

Unlike the Local Word2Vec model, where word embeddings are trained on a subset of the corpus, this task uses pre-trained embeddings from external sources:

- **Word2Vec:** Pre-trained embeddings trained on the Google News corpus.
- **GloVe:** Pre-trained embeddings trained on the Wikipedia + Gigaword corpus.

The pre-trained models are loaded into the system, allowing access to a large vocabulary of terms, including many that may not appear frequently in the local document collection.

5.2 Query Expansion

Once the pre-trained models are loaded, query expansion is performed using the following steps:

- **Query Processing:** Similar to the Local Word2Vec model, the initial query is processed using tokenization and stopword removal. The processed query terms are then mapped to their corresponding word vectors in the embedding space.
- **Nearest Neighbor Search:** For each query term found in the vocabulary of the pre-trained embeddings, the top-10 most similar terms are identified using cosine similarity. These terms represent semantically related words that are likely to help refine the query.
- **Query Expansion:** The top-20 terms across all query terms are selected for expansion. These terms are weighted based on their cosine similarity score and are appended to the original query. Expansion terms that do not appear in the corpus language model are discarded to ensure that only relevant terms are included in the reranking process.

5.3 Reranking Using KL Divergence

As with the Local Word2Vec model, the reranking is performed using Kullback-Leibler (KL) divergence. After the query is expanded, the system ranks the documents based on how closely their language models match the expanded query's language model. KL divergence is computed between the query and document models, with lower divergence indicating a higher degree of relevance.

$$D_{KL}(P||Q) = \sum_{w} \mathrm{P}(w|R) \log \mathrm{P}(w|D)$$

where $P(w|Q)$ and $P(w|D)$ are the probabilities of term w in the expanded query and the document, respectively. This method ensures that documents with term distributions closer to the expanded query are ranked higher.

5.4 Implementation Details

The query expansion and reranking process for the pre-trained embeddings is encapsulated within the same `QueryExpander` class used for the Local Word2Vec model. The main difference lies in the method for loading the models and performing expansion based on pre-trained embeddings. The rest of the steps, including document scoring and reranking, remain the same as in Task 1.

5.5 Parameter Tuning

We experimented with different values of k (the number of top documents used for pseudo-relevance feedback) and found that setting $k = 50$ produced

the best performance. Additionally, we evaluated different expansion sizes (i.e., the number of added terms) and observed that expanding the query with 20 additional terms provided optimal results.

5.6 Evaluation and Results

Both models were evaluated using nDCG metrics. The GloVe model performed slightly better in terms of nDCG@5, while Word2Vec showed better results for nDCG@10 and nDCG@50.

6 Results

6.1 Localy Trained Word2Vec

The following results were obtained using locally trained Word2Vec on the first five queries:

- Query 1: 0.7171, 0.7171, 0.7171
- Query 2: 0.4966, 0.5044, 0.6125
- Query 3: 0.0, 0.0800, 0.3753
- Query 4: 0.1083, 0.1591, 0.5202
- Query 5: 0.5363, 0.4171, 0.5776
- Average: 0.7227, 0.5869, 0.7174

The expanded terms for each query are as follows:

- Query 1: salaries, averaged, bls, hourly, assistants, percentile, hygienists, earn, part-time, employment
- Query 2: alterations, brides, seamstresses, hems, bustle, gown, seamstress, bridal, dresses, renting
- Query 3: stillbirth, progesterone, heartbeat, experiencing, urination, fetal, cramping, miscarriages, contractions, chromosomal
- Query 4: constructive, contexts, context, affirmation, logical, definitions, distinction, meritorious, discourse, sentences
- Query 5: dropbox, sync, app, autosave, chrome, downloading, sheets, files, disable, android
- Query 6: taxes, deductions, taxed, arkansas, rollovers, income-tax, taxpayers, retirees, exempt, retirement

6.2 Word2Vec: Pre-Trained for First 5 Queries

The following results were obtained using pre-trained Word2Vec on the first five queries:

- Query 1: 0.2774, 0.2774, 0.2774
- Query 2: 0.3486, 0.3992, 0.5523
- Query 3: 0.3072, 0.3993, 0.4989
- Query 4: 0.6438, 0.6366, 0.7720
- Query 5: 0.1792, 0.2236, 0.4644
- Average: 0.2591, 0.3764, 0.5187

The expanded terms for each query are as follows:

- Query 1: hygienists, dentists, dentist, dentistry, salaries, medical, orthodontics, cleanings, maxillofacial, orthodontic, employer-sponsored, physicians, doctors, healthcare, hygiene, health-care, nurses, patient
- Query 2: dresses, pre-wedding, clothes, attire, floor-length, bride, gown, sequins, expensive, costume, gowns, wearing, groom, nuptials, actual
- Query 3: abruption, eclampsia, pre-eclampsia, preeclampsia, jaundice, stillbirth, toxoplasmosis, hypovolemic, amenorrhea, prematurity, stillbirths, sepsis, auto-immune, antiphospholipid, coagulopathy, hydronephrosis, infection, anencephaly
- Query 4: definitions, defining, praiseworthy, define, sense, regard, notion, insofar, admirable, defines, defined, commendable, terminology, redefinition, normative, notions, context, principle, endeavour, interpretation
- Query 5: uploader, web, one-click, upload, gif, log-in, Gmail, Picasa, inbox, apps, config, user, resize, app, users, add-ins, xml
- Query 6: taxes, taxable, taxation, surtax, taxed, incomes, income-tax, taxpayers, after-tax, levy, revenues, Tax, VAT, Income, revenue, untaxed, deductions, profits, levied

6.3 Pretrained GloVe for First 5 Queries

The following results were obtained using GloVe on the first five queries:

- Query 1: 0.7171, 0.7171, 0.7171
- Query 2: 0.4991, 0.4847, 0.6486
- Query 3: 0.0969, 0.0778, 0.3744
- Query 4: 0.5310, 0.5424, 0.7286
- Query 5: 0.1465, 0.2447, 0.4761
- Average: 0.4125, 0.4619, 0.6321

The expanded terms for each query are as follows:

- Query 1: dentist, insurance, salaries, per, tuition, averages, pay, percentage, earning, lowest, benefits, earn, minimum, care, monthly, income, employee, job, nursing, adjusted
- Query 2: less, typical, actual, mean, actually, costs, amount, every, even, expensive, much, change, required, pay, size, instance, changes, example, price, wear
- Query 3: causes, illness, complications, infection, caused, symptoms, disease, severe, birth, result, suffering, pregnant, pregnancies, cancer, onset, diagnosis, causing, fatal, sudden, childbirth
- Query 4: definitions, defining, defines, objectives, objective, defined, define, implies, achievable, attainable, insofar, admirable, incompatible, unambiguous, worthwhile, necessarily, understandable, regard, accomplishing, accomplish
- Query 5: yahoo, internet, software, web, online, microsoft, users, apple, companies, computer, automotive, consumers, download, technology, gm, pc, aol, search, industry, automobile
- Query 6: taxes, revenue, incomes, pay, revenues, taxpayers, payments, savings, paying, benefit, spending, benefits, money, increase, profits, insurance, businesses, payroll, tuition, taxation

7 Conclusion

The Local Word2Vec model achieved the best overall performance for the task, particularly for nDCG@50. The GloVe model provided better results for nDCG@5, likely due to the more diverse training corpus. Future improvements could focus on optimizing the selection of expansion terms and experimenting with different neural embedding models.