Lunr-MiniLM-L6 Information Retrieval Pipeline Using CHERCHE

Nouh Chelgham[1], Alexandra Villon[1]

[1]Université Toulouse III Paul Sabatier, 118 Route de Narbonne, 31000 Toulouse

Abstract

This report details the implementation and evaluation of an information retrieval (IR) pipeline using the CHERCHE library, which integrates traditional lexical retrieval methods with neural re-ranking models.The pipeline begins with document retrieval using the Lunr full-text search library, followed by re-ranking models, including all-mpnet-base-v2, LaBSE, paraphrase-albert-small-v2, and all-MiniLM-L6-v2. Key metrics such as nDCG@10, RecipRank, Recall@10, and Precision@10 are used to assess performance. Among the models tested, all-MiniLM-L6-v2 offered the best balance between efficiency and performance. Additionally, this work contributes to our participation in the ReNeuIR workshop.

Keywords

Neural Information Retrieval, Cherche library, Efficency, ReNeuIR Workshop

1. Introduction

Information retrieval (IR) systems are critical in various applications, such as search engines, recommendation systems, and document indexing. The efficiency and accuracy of these systems are paramount to delivering relevant results to users' queries. However, implementing effective IR pipelines can be complex, requiring seamless integration of different components, such as initial indexing, document retrieval and subsequent re-ranking.

CHERCHE [4] implements most common retrievers based on lexical matching between the queries and the documents, summarizing: Tfidf, BM25L and BM25Okapi, Elastic, Lunr, Flash, Encoder, DPR and Fuzz retrievers.

This report examines an IR pipeline implemented using the CHERCHE tool, which simplifies the creation of retrieval and ranking systems.

The provided Python code demonstrates a pipeline that integrates traditional retrieval techniques with modern neural re-ranking models using a pretrained transformer model. The pipeline's performance is then evaluated using pytrec_eval, a library designed to assess the effectiveness of IR systems using standard metrics.

2. Implementation overview

The implementation involves several key components:

2.1. Dataset Loading

The pipeline begins by loading a dataset using the ir_datasets library, which provides standardized access to various IR datasets. The documents and queries are iterated over and converted into a list format for further processing. In this shared task, we used dl-top-1000-docs-20240701-training dataset that was previously downloaded from Tira_client.

```python
from tira.third_party_integrations import ir_datasets
dataset = ir_datasets.load(dataset_path)
```

The approach was submitted to TIRA/TIREx [1, 2] and evaluated as part of the ReNeuIR [3] workshop.

noahchelgham@gmail.com (N. Chelgham); a.villon.h@gmail.com (A. Villon)

```
docs = list(dataset.docs_iter())
queries = list(dataset.queries_iter())
```

2.2. Initial Retrieval Using Lunr

The first retrieval step employs Lunr, a full-text search library that indexes documents and allows for rapid retrieval based on text similarity. Documents are indexed using their id and text fields, and the Lunr retriever is used to retrieve the top 100 documents for each query.

```
import cherche.retrieve as retrieve
retriever = retrieve.Lunr(key='id',
            on=['text'],
            documents=documents, k=100)
run = {query.query_id:
    {x['id']:float(x['similarity'])
    for x in retriever(query.text)}
    for query in dataset.queries_iter()}
```

2.3. Neural Re-Ranking with Sentence Transformers

Following the initial retrieval, a neural re-ranking model is applied using the SentenceTransformer library. The model re-ranks the top 50 documents based on semantic similarity to the query. In this step, reranker model measures semantic similarity between documents and queries unlike the lexical similarity in retrieval step.

```
import cherche.rank as rank
from sentence_transformers import SentenceTransformer
ranker = rank.Encoder(key='id',on=['text'],
    encoder=SentenceTransformer ('sentence-transformers/LaBSE').encode,
    k=50 )

search = retriever + ranker
search.add(documents=documents)
```

2.4. Writing Retrieval Results

The results of both the initial retrieval and the re-ranking are written to a file in the TREC format, which is a standard format for sharing IR results.

```
def write_trec_run(run, filename, run_name="run_name"):
with open(filename, 'w') as f:
    for query_id, doc_scores in run.items():
        for rank, (doc_id, score) in enumerate(sorted(doc_scores.items(),
        key=lambda x: x[1], reverse=True), start=1):
            f.write(f"{query_id} Q0 {doc_id} {rank} {score} {run_name}\n")
```

3. Evaluation and Results

3.1. Loading Qrels and Results

The evaluation process begins by loading the qrels (ground truth relevance judgments) and the results (retrieved documents' scores) from respective files. The qrels indicate the correct relevance of documents to queries, while the results file contains the retrieved documents and their scores.

```python
def load_qrels(qrels_file):
    qrels = {}
    with open(qrels_file, 'r') as f:
        for line in f:
            qid, _, doc_id, relevance = line.strip().split()
            if qid not in qrels:
                qrels[qid] = {}
            qrels[qid][doc_id] = int(relevance)
    return qrels
```

3.2. Evaluating the Retrieval and Ranking

The core evaluation is conducted using the pytrec_eval library, which computes metrics like nDCG@10, RecipRank, Recall@10, and Precision@10. These metrics provide a comprehensive assessment of the system's effectiveness in ranking relevant documents.

```python
import pytrec_eval
evaluator = pytrec_eval.RelevanceEvaluator(qrels,
            {'ndcg_cut_10', 'recip_rank', 'recall_10', 'P_10'})
metrics = evaluator.evaluate(results)
```

3.3. Experiences

We have evaluated several reranker models and compared the performance on mentionned key metrics. The models under consideration are:

- all-mpnet-base-v2
- LaBSE
- paraphrase-albert-small-v2
- all-MiniLM-L6-v2

The evaluation also considers the time taken by each model to complete the re-ranking process. We have used NVIDIA RTX 3050 GPU for pipeline execution.

Model	Time Taken	nDCG@10	RecipRank	Recall@10	Precision@10
all-mpnet-base-v2	1h 20min	0.6767	0.9180	0.2018	0.7443
LaBSE	26min 30sec	0.6564	0.9295	0.1990	0.7392
paraphrase-albert-small-v2	1h 10min	0.5280	0.8357	0.1435	0.5711
all-MiniLM-L6-v2	25min	0.6564	0.9295	0.1990	0.7392

The comparison might reveal the strengths and weaknesses of the IR pipelines. For instance, high nDCG@10 and RecipRank values indicate good ranking of relevant documents, while lower Recall@10 might suggest some relevant documents were missed in the top 10 results.

With a processing time of only 25 minutes, the all-MiniLM-L6-v2 model is the fastest and achieves the best balance between efficiency and effectiveness, with a strong nDCG@10 of 0.6564 and RecipRank of 0.9295

4. Conclusion

The IR pipeline implemented using CHERCHE showcases a robust combination of traditional retrieval and neural re-ranking techniques. The modularity and flexibility of CHERCHE facilitate rapid prototyping and experimentation, making it a valuable tool in the IR domain. Among the models tested, all-MiniLM-L6-v2 emerged as the best choice for the objectives of the ReNeuIR workshop, thanks to its efficient processing time and strong performance metrics. While the model effectively ranks relevant documents at the top, the relatively low Recall@10 score (0.1990) across all models suggests that there is still room for improvement in retrieving a more exhaustive set of relevant documents within the top 10 results. This could be a key area for future research to further optimize the pipeline.

References

[1] M. Fr”obe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous Integration for Reproducible Shared Tasks with TIRA.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. doi:10.1007/978-3-031-28241-6_20.

[2] M. Fr”obe, J. H. Reimer, S. MacAvaney, N. Deckers, S. Reich, J. Bevendorff, B. Stein, M. Hagen, M. Potthast, The Information Retrieval Experiment Platform, in: H.-H. Chen, W. Duh, H.-H. Huang, M. Kato, J. Mothe, B. Poblete (Eds.), 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023), ACM, 2023, pp. 2826–2836. doi:10.1145/3539618.3591888.

[3] M. Fröbe, J. Mackenzie, B. Mitra, F. M. Nardini, M. Potthast, ReNeuIR at SIGIR 2024: The Third Workshop on Reaching Efficiency in Neural Information Retrieval, in: 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024), ACM, 2024.

[4] R. Sourty, J. G. Moreno, L. Tamine, F.-P. Servant, Cherche: A new tool to rapidly implement pipelines in information retrieval, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’22, Association for Computing Machinery, New York, NY, USA, 2022, p. 3283–3288. URL: https://doi.org/10.1145/3477495.3531695. doi:10.1145/3477495.3531695.