NLP for Search

Rishabh Ranjan (Intern)
Jongjin Bae (Mentor)

Samsung Internship Project - Summer 2021

Enterprise Search Engines

Lucene, Solr and ElasticSearch

Popular search engines for production

- **Apache Lucene** is a Java library for full-text search
- **Apache Solr** and **ElasticSearch** are enterprise platforms using Lucene
- **Scalable** for big data applications
- **RESTful** HTTP API and additional clients

Search Methodology

Traditional Information Retrieval (IR)

- **Documents:** collections of fields
- **Preprocessing:** tokenisation, stemming, stop-word removal, normalisation
- **Indexing:** inverted index of postings lists
- **Retrieval:** matched documents - boolean, fuzzy search
- **Ranking:** ordering by relevance
- Default: BM25 (SOTA for tf.idf based ranking functions)
- Cosine similarity for vector space models

Semantic Search

Understanding the query

- Search with **meaning** to address searcher intent
- Term based matching is not sufficient
- Requires **natural language understanding**
- **No direct support** in Lucene / Solr / ElasticSearch
- Some examples:
  - "zipping up files" returns "Compressing / Decompressing Folders & Files"
  - "determine if something is an IP" returns "How do you tell whether a string is an IP or a hostname"
  - "translate bytes to doubles" returns "Convert Bytes to Floating Point Numbers in Python"

Enabling Semantic Search

Topic Modelling

A classical ML technique

- **Statistical model** to discover **topics**
- Methods include: LSA, LDA, HDP, NNMF
- Low-dimensional vector representations
- **Gensim** library
- Useful with low resources
- Inferior to deep language models
  - omitted in evaluation

Language Modelling

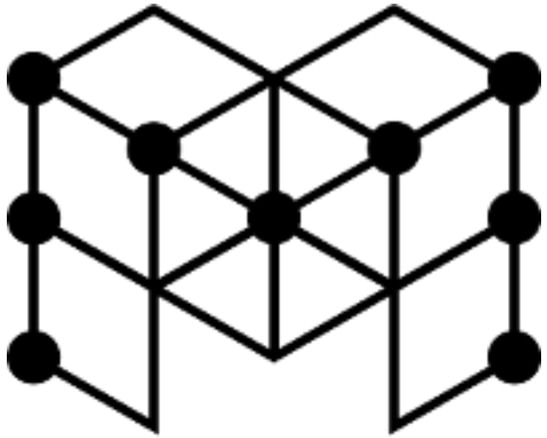
Deep learning for NLP

- **Language Models** to summarise docs / queries into vectors
- Transformer based models
- Pre-training on huge text corpora
- Eg. **BERT** and variants

Selecting the Best Techniques

The MS-MARCO document ranking dataset and leaderboard

- Large scale text IR dataset
- **Source:** real Bing searches
- Top-100 document (re-)ranking task
- Leaderboard ranked using MRR metric
- Top entries:
  - Ensembles
  - Combinations of methods



Shortlisted Techniques

From the MS-MARCO leaderboard

1. **ANCE**: Approximate nearest neighbour Negative Contrastive Estimation
   - training mechanism for BERT-Siamese Dense Retrieval model
2. **PROP**: Pre-training with Representative wOrds Prediction
   - pre-training BERT-like models for IR-oriented objectives
3. **HDCT**: Context-aware Hierarchical Document Term-weighting
   - estimates semantic importance of terms for bag-of-words model
4. **BERT**: Bi-directional Encoder Representations from Transformers
   - baseline model with original training scheme

Search Engine + NLP

Combination Strategies

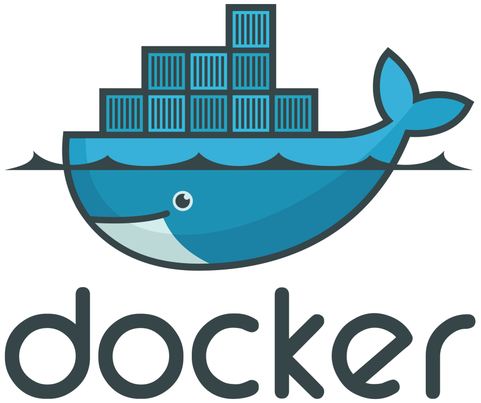
Possibilities and limitations

- **Enhance** dataset - eg. doc2query, docT5query
- **Fine-tune** model on dataset
- **Embed** documents
- Direct rank vs retrieve and re-rank
- dense_vector field in ElasticSearch
- Fast approximate nearest neighbour (ANN) not yet supported

Architecture

ElasticSearch and Huggingface

- Docker container for **ElasticSearch**
  - separate index for each technique
  - communicate via localhost:9200
- **Huggingface transformers** library:
  - pre-trained Pytorch models
  - patch for models not in library
  - mean-pool output of second-last layer
  - embedding-head output when available
- Scripts to embed, index and evaluate



Infrastructure Limitations

Working from home

- Samsung SmartTV **dataset** unavailable
  - Use MS-MARCO documents as corpus
  - Use MS-MARCO dev set for evaluation (eval set is not public)
- No **GPU**
  - Skip fine-tuning step: use pre-trained models
  - CPU based inference
- Further simplifications
  - No enhancements or ensembles

Implementation Details

Index settings and scripts

- **Settings:** number_of_shards: 1, number_of_replicas: 0
- **Mappings:** docid: keyword, body: text, emb: dense_vector
- **Scripts:**
  - make_docs.py: MS-MARCO docs (*tsv*) to ElasticSearch docs (*jsonl*)
  - make_index.py: make new ElasticSearch index from config (*json*)
  - index_docs.py: add ElasticSearch docs (*jsonl*) to existing index

Evaluation

TREC Metrics

Standard evaluation for the "Text REtrieval Conference"

- Relevant Documents Returned
- Mean Average Precision (**MAP**)
- R Precision (**R-prec**)
- Mean Reciprocal Rank (**MRR**)
- MS-MARCO has 1 relevant doc per query:
  - **MAP == MRR**
  - **R-prec** == relevant docs returned at rank 1

Evaluation Setup

2-stage search

- Stage-1: **Retrieve**
  - top-100 documents from 3.2M documents
  - full-length documents
  - BM25 and BM25 + HDCT
- Stage-2: **Rerank**
  - reorder the retrieved top-100
  - documents trimmed to 50 words
  - tinyBERT, BERT-Base, PROP, ANCE-FirstP, ANCE-MaxP

Implementation Details

Scripts and Optimisations

- Query:
  - 'script_score' with cosineSimilarity / 'match' for BM25 / 'filter'+'script_score' for reranking
- Scripts:
  - make_queries.py: MS-MARCO queries (*tsv*) to ElasticSearch queries (*jsonl*)
  - search_queries.py: search index for queries
  - make_trec.py: convert search results (*jsonl*) to TREC format
- Optimisations:
  - make '*url*', '*title*' and '*body*' fields non-indexable
  - trim documents to 50 words for reranking
  - parallelise on all 4 cores wherever possible
  - use volume instead of bind-mount in docker

Results

Method Details

All models were fine-tuned on MS-MARCO

- **BM25:** traditional text IR method, default in ElasticSearch
- **HDCT:** pre-process docs for better results with BM25
- **tinyBERT:** BERT architecture with 6 layers and 312 dimension embedding
- **BERT-Base:** BERT architecture with 12 layers and 768 dimension embedding
- **PROP:** BERT-Base pre-trained using PROP objective
- **ANCE:** fine-tuned RoBERTa with 12 layers and 768 dimension embedding
  - **ANCE-FirstP:** first chunk pooling strategy
  - **ANCE-MaxP:** max pooling strategy

Top-100 Docs →	BM25 [Relevant Docs (%) = 80]		BM25+HDCT [Relevant Docs (%) = 85]	
Rerank ↓ \ Metric → (Words = 50)	MRR / MAP (%)	R-Precision (%)	MRR / MAP (%)	R-Precision (%)
TinyBERT	13.26	10.00	12.99	10.00
BERT-Base	18.32	10.00	18.22	10.00
PROP	12.45	10.00	14.00	10.00
ANCE-FirstP	34.47	20.00	35.65	20.00
ANCE-MaxP	32.66	20.00	**37.99**	**25.00**
None (Words = ∞)	15.94	5.00	35.94	20.00

Thank you!