1 Text Representations for Ranking

1.1 BOW Encodings

The Bag of Words (BOW) encoding is a popular method for text representation in information retrieval. It is based on the frequency of terms in a document, where each term's contribution to the document score cannot exceed a saturation point, regardless of its frequency in the document. This is a valuable property of the BM25 weighting function, which has been shown to perform better than traditional term-weighting functions such as tf\*idf [REF0].

In the BOW model, different streams of text may be more predictive of relevance than others. For instance, a query match on the title might provide stronger evidence of possible relevance than an equivalent match on the body text. This is particularly true in the Web context, where matching on anchor text is a strong signal of relevance. A practical approach to this stream structure is to apply the ranking function separately to each stream, and then combine these in some linear combination for the final document score [REF1].

The maximum likelihood estimate of the probability of a term under the term distribution for a document is given by the raw term frequency of the term in the document divided by the total number of tokens in the document. However, this estimator has its limitations. For instance, it is not desirable to assign a probability of zero to a document that is missing one or more of the query terms [REF2].

The BOW model also makes certain independence assumptions. For instance, it assumes that for any non-query term, the ratio of conditional probabilities is constant and equal to one. This might seem a drastic assumption, but it is necessary for the model to work [REF4].

The BOW model also assumes a property called eliteness for each term. This can be interpreted as a form of aboutness: if the term is elite in the document, in some sense the document is about the concept denoted by the term. The actual occurrences of the term in the document depend on eliteness, and there may be an association between eliteness and relevance to the query [REF7].

In conclusion, the BOW model is a powerful tool for text representation in information retrieval. It is based on term frequency and makes certain assumptions about the independence and eliteness of terms. Despite its limitations, it has been shown to perform well in practice [REF8].

References sent to GTP:

REF0 - The Probabilistic Relevance Framework: BM25 and Beyond

REF1 - The Probabilistic Relevance Framework: BM25 and Beyond

REF2 - A Language Modeling Approach to Information Retrieval

REF3 - The Probabilistic Relevance Framework: BM25 and Beyond

REF4 - The Probabilistic Relevance Framework: BM25 and Beyond

REF5 - The Probabilistic Relevance Framework: BM25 and Beyond

REF6 - The Probabilistic Relevance Framework: BM25 and Beyond

REF7 - The Probabilistic Relevance Framework: BM25 and Beyond

REF8 - The Probabilistic Relevance Framework: BM25 and Beyond

REF9 - Scalability Challenges in Web Search Engines

1.2 LTR Features

Learning to rank (LTR) is a critical aspect of information retrieval, where the goal is to learn a ranking function that maps an input vector to a member of an ordered set of numerical ranks [REF1]. This process is often cast as ordinal regression, where ranks are modeled as intervals on the real line, and loss functions depend on pairs of examples and their target ranks [REF1]. The positions of the rank boundaries play a crucial role in the final ranking function [REF1].

One of the methods used in LTR is PRank, which maps a feature vector x ∈ R d to the reals with a learned w ∈ R d such that the output of the mapping function is just w • x [REF1]. PRank learns using one example at a time, which is considered an advantage over pair-based methods, as the latter must learn using O(m 2 ) pairs rather than m examples [REF1].

Another approach to LTR is LambdaRank and LambdaMART, which update their parameters differently [REF0]. LambdaRank updates all the weights after each query is examined, while LambdaMART updates only a few parameters at a time, but using all the data [REF0]. This allows LambdaMART to choose splits and leaf values that may decrease the utility for some queries, as long as the overall utility increases [REF0].

In contrast to other supervised tasks, in LTR, every example document in the training data is not independent and identically distributed (i.i.d) [REF3]. Instead, the documents associated with each query in the sample form a group - the groups are i.i.d., but the documents within a group are not i.i.d [REF3]. This necessitates the documents to be identified in a deterministic manner [REF3].

The training data in LTR is partitioned by query [REF6]. For a given query, each pair of URLs with differing labels is chosen, and each such pair is presented to the model, which computes the scores [REF6]. The two outputs of the model are mapped to a learned probability that one URL should be ranked higher than the other via a sigmoid function [REF6].

Sampling is another important aspect of LTR [REF2]. A sample is used during the application of a learned model to reduce the size of the number of documents for which features are calculated [REF2]. This provides efficiency advantages, particularly if some features are expensive to compute [REF2]. However, the learning time of many LTR techniques increases as the number of documents in the sample increases [REF3].

In conclusion, text representations for ranking in LTR involve various features and techniques, including ordinal regression, PRank, LambdaRank, LambdaMART, and sampling. Each of these has its own advantages and disadvantages, and their choice depends on the specific requirements of the task at hand.

References sent to GTP:

REF0 - From RankNet to LambdaRank to LabdaMART: An Overview

REF1 - Learning to Rank using Gradient Descent

REF2 - The Whens and Hows of Learning to Rank for Web Search

REF3 - The Whens and Hows of Learning to Rank for Web Search

REF4 - Learning to Rank using Gradient Descent

REF5 - Learning to Rank using Gradient Descent

REF6 - From RankNet to LambdaRank to LabdaMART: An Overview

REF7 - From RankNet to LambdaRank to LabdaMART: An Overview

REF8 - The Whens and Hows of Learning to Rank for Web Search

REF9 - Learning to Rank using Gradient Descent

1.3 Word Embeddings

Word embeddings are a type of text representation that can be used for ranking in information retrieval. They are distributed representations of words in a vector space that group similar words together, which can improve the performance of learning algorithms in natural language processing tasks [REF9]. One of the earliest uses of word representations was in 1986, and since then, they have been applied to a wide range of natural language processing tasks, including statistical language modeling, automatic speech recognition, and machine translation [REF9].

One of the most popular methods for generating word embeddings is the Skip-gram model, which is an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data [REF9]. The Skip-gram model benefits from observing the co-occurrences of words, but it benefits much less from observing the frequent co-occurrences of common words, as nearly every word co-occurs frequently within a sentence with these common words [REF2].

Another method for generating word embeddings is the masked language model (MLM), which enables the representation to fuse the left and the right context, allowing for the pretraining of a deep bidirectional Transformer [REF0]. Unlike unidirectional language models, which are used for pre-training, the MLM uses masked language models to enable pretrained deep bidirectional representations [REF0].

In addition to these methods, there are also matrix factorization methods for generating low-dimensional word representations, such as the Global Vectors (GloVe) model [REF5]. The GloVe model captures the global corpus statistics directly, making efficient use of statistics [REF5]. It operates on the matrix of word-word co-occurrence counts, tabulating the number of times a word occurs in the context of another word [REF8].

Despite the high accuracy of these models, there are significant differences across them. For example, the SWOW-RW and SWOW-PMI models outperformed all other models in predicting guesser responses, while the USF-RW and USF-PPMI models performed similarly but outperformed the USF-S model, as well as word2vec and GloVe [REF6].

In conclusion, word embeddings are a powerful tool for text representation in information retrieval, with various methods available for generating these embeddings. Each method has its strengths and weaknesses, and the choice of method will depend on the specific requirements of the task at hand.

References sent to GTP:

REF0 - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

REF1 - GloVe: Global Vectors for Word Representation

REF2 - Distributed Representations of Words and Phrases and their Compositionality

REF3 - GloVe: Global Vectors for Word Representation

REF4 - GloVe: Global Vectors for Word Representation

REF5 - GloVe: Global Vectors for Word Representation

REF6 - Semantic Memory Search and Retrieval in a Novel Cooperative Word Game: A Comparison of Associative and Distributional Semantic Models

REF7 - GloVe: Global Vectors for Word Representation

REF8 - GloVe: Global Vectors for Word Representation

REF9 - Distributed Representations of Words and Phrases and their Compositionality

2 Interaction-focused Systems

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been incorporated into interaction-focused systems to account for positional information, as seen in models such as MatchPyramid and local DUET [REF0]. These models, however, have struggled to significantly outperform simpler models like DRMM, indicating the complexity of utilizing positional information in deep neural IR models [REF0]. The interactions between a query and a document are sequential along both the query and document axes, making the problem multi-dimensional [REF0].

CNNs are crucial for modeling n-grams in typical IR approaches, which treat n-grams as discrete terms and use them the same as unigrams [REF1]. However, treating n-grams atomically in neural IR can lead to an explosion in the parameter space and data sparsity [REF1]. This issue can be avoided by learning a convolutional layer that forms n-grams from individual words' embeddings, projecting all n-grams into a unified embedding space [REF1]. This allows for matching n-grams of different lengths, providing partial evidence for queries [REF1].

In the class of interaction-focused models, models like Deep Match Tree and Match-SRNN have been proposed for short text matching and semantic matching problems respectively [REF5]. These models leverage deep neural networks for making matching decisions based on local interactions [REF5]. However, these models are designed for semantic matching, which is significantly different from the relevance matching problem in ad-hoc retrieval [REF5].

Conv-KNRM, a Convolutional Kernel-based Neural Ranking Model, embeds words in continuous vectors and employs CNNs to compose adjacent words' embeddings to n-gram embeddings [REF6]. This model has demonstrated its ability to cross-match n-grams with different lengths in a unified space [REF3]. Conv-KNRM overcomes the lexical mismatch and finds query-document connections that are difficult for exact-match-based approaches [REF4]. It also captures n-gram matches that are different from word matches, illustrating its generalizability [REF4].

The main challenge of translation models is the sparsity of word-pair translations [REF8]. This problem can be overcome by introducing word embeddings to calculate the translation scores [REF8]. CNNs have been used to combine the word-level translation scores to generate query-document ranking scores [REF8]. However, a later study found that the CNN filters tend to mix the match signals in the translation matrix at various levels and are suboptimal for ad hoc search [REF8].

In conclusion, while CNNs have been incorporated into interaction-focused systems, there are still challenges to be addressed. These include the complexity of utilizing positional information in deep neural IR models, the sparsity of word-pair translations, and the suboptimal performance of CNN filters in ad hoc search [REF0][REF8].

References sent to GTP:

REF0 - PACRR: A Position-Aware Neural IR Model for Relevance Matching

REF1 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

REF2 - A Deep Relevance Matching Model for Ad-hoc Retrieval

REF3 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

REF4 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

REF5 - A Deep Relevance Matching Model for Ad-hoc Retrieval

REF6 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

REF7 - A Deep Relevance Matching Model for Ad-hoc Retrieval

REF8 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

REF9 - Convolutional Neural Networks for Soft-Machting N-Grams in Ad-hoc Search

2.2 Pre-trained Language Models

Interaction-focused systems, particularly pre-trained language models, have seen significant advancements in recent years. One such model is BART, a denoising autoencoder built with a sequence-to-sequence model. It is designed to reconstruct original text from corrupted versions, making it applicable to a wide range of tasks. BART's architecture is a standard Transformer-based neural machine translation model, which generalizes BERT's bidirectional encoder and GPT's left-to-right decoder [REF0].

The pretraining process of BART involves two stages: text corruption using an arbitrary noising function and learning a sequence-to-sequence model to reconstruct the original text. The noising flexibility of BART allows for arbitrary transformations to be applied to the original text, including changing its length. This model has shown effectiveness when fine-tuned for text generation and comprehension tasks [REF0].

Another approach to pre-training involves multi-task learning, where a single model is trained on a mixture of tasks and evaluated using different parameter settings. This approach can be extended by pre-training the model on all tasks at once and then fine-tuning it on individual supervised tasks [REF1].

Noising the input is a crucial part of pre-training. Some of the transformations experimented with include token deletion, where random tokens are deleted from the input, and text infilling, where text spans are replaced with a single [MASK] token [REF2].

Training with large mini-batches has been shown to improve optimization speed and end-task performance when the learning rate is increased appropriately. This approach has been found to be effective with BERT [REF3].

DistilBERT, a smaller and faster version of BERT, retains 97% of the language understanding capabilities. It demonstrates that a general-purpose language model can be successfully trained with distillation [REF4].

BERT has shown significant performance improvements over previous state-of-the-art systems. BERT LARGE, in particular, has outperformed BERT BASE across all tasks, especially those with very little training data [REF5].

The Text-to-Text Transfer Transformer (T5) is another model that treats every problem as a text-to-text task. It is based on the Transformer architecture and has been used in a wide variety of NLP settings [REF7].

The Transformer architecture has been scaled up to explore performance with larger models. For instance, the "3B" variant has around 2.8 billion parameters, while the "11B" variant has about 11 billion parameters [REF8].

In conclusion, pre-trained language models have shown significant potential in improving performance across a wide range of tasks. The flexibility and scalability of these models make them a promising area for future research and development.

References sent to GTP:

REF0 - BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

REF1 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF2 - BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

REF3 - RoBERTa: A Robustly Optimized BERT Pretraining Approach

REF4 - DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

REF5 - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

REF6 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF7 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF8 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF9 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

2.3 Ranking with Encoder-only Models

Interaction-focused systems, particularly those that employ encoder-only models, have seen significant advancements in recent years. These systems have been greatly aided by the release of large-scale datasets such as MS MARCO and TREC CAR, which have been instrumental in training data-hungry neural models[REF0].

The interaction-focused systems typically follow a two-stage process. The first stage involves the retrieval of a set of documents, which are then re-ranked in the second stage using a more computationally-intensive method. The re-ranker's job is to estimate a score of how relevant a candidate passage is to a query. BERT has been widely used as a re-ranker, where the query is fed as sentence A and the passage text as sentence B[REF1].

For end-to-end document ranking, these models are arranged in a pipeline, balancing the size of the candidate set against the model's inherent complexity. This design allows the benefits of richer models to be obtained while controlling the increased inference latencies that come with these models. An example of this is the integration of monoBERT and duoBERT in a multistage ranking architecture[REF2].

The effectiveness of these models is often evaluated on large-scale document retrieval datasets such as the MS MARCO dataset and the Complex Answer Retrieval (CAR) Task at TREC. Both monoBERT and duoBERT contribute significantly to overall effectiveness, and pre-training on the corpus of the target task has been shown to improve effectiveness over pre-training on out-of-domain corpora[REF3].

The performance of these models can be further improved by incorporating contextualized language models into existing neural ranking architectures. This approach, known as CEDR (Contextualized Embeddings for Document Ranking), uses multiple similarity matrices - one for each layer of the language model. Despite the increased computation costs, this approach has been shown to considerably improve ranking performance[REF5].

The training of these models often involves a large number of example pairs. For instance, a model might be trained on 30M example pairs for 100k iterations. Despite the large training size, the models often do not show any gain on the dev set by training the model longer[REF6].

The final list of candidates is obtained by re-ranking the candidates according to their scores. Various methods such as SUM, BINARY, MIN, MAX, and SAMPLE can be used to measure the relevance of a candidate against other candidates[REF7].

Fine-tuning is often conducted on the test set, representing the maximum performance of the model when using static parameters over each dataset. A pretrained BERT model with a linear combination layer stacked atop the classifier token is often used for fine-tuning[REF8].

However, the improvement in MRR often comes at a cost in increased latency. For instance, duoBERT requires an additional 50 × 49 BERT inferences to compute the final ranking. Despite this, the BINARY method has been found to perform slightly better than SUM on the development set[REF9].

References sent to GTP:

REF0 - Multi-Stage Document Ranking with BERT

REF1 - Passage Re-Ranking with BERT

REF2 - Multi-Stage Document Ranking with BERT

REF3 - Multi-Stage Document Ranking with BERT

REF4 - Passage Re-Ranking with BERT

REF5 - CEDR: Contextualized Embeddings for Document Reranking

REF6 - Multi-Stage Document Ranking with BERT

REF7 - Multi-Stage Document Ranking with BERT

REF8 - CEDR: Contextualized Embeddings for Document Reranking

REF9 - Multi-Stage Document Ranking with BERT

2.4 Ranking with Encoder-decoder Models

Interaction-focused systems, particularly those that employ encoder-decoder models, have been instrumental in advancing the field of information retrieval. These models are designed to ingest a prefix before making predictions, where a span of text is sampled from an unlabeled dataset and split into prefix and target portions [REF0]. The choice of unsupervised objective is crucial as it provides the mechanism through which the model gains general-purpose knowledge to apply to downstream tasks [REF6].

The encoder-decoder architecture uses fully visible masking in the encoder and the encoder-decoder attention, with causal masking in the decoder [REF3]. This architecture has been used in a variety of models, including the decoder-only prefix language model (LM) which uses fully-visible self-attention over the input [REF0]. The LM is fed the concatenation of the input and target, using a causal mask throughout [REF3].

The use of unsupervised objectives has been a common practice in pre-training models, with the basic language modeling objective being a popular choice due to its natural fit for the language model architectures [REF0]. However, recent studies have shown that "denoising" objectives, where the model is trained to predict missing or corrupted tokens in the input, produce better performance [REF5].

In terms of data, the "Colossal Clean Crawled Corpus" (C4) has been introduced, which comprises heuristically-cleaned text from the Common Crawl web dump. The use of a large and diverse dataset like C4 has been motivated by the fact that performance can degrade when an unlabeled dataset is small enough that it is repeated many times over the course of pre-training [REF1].

The document ranking task has been approached using a novel generation-based method with pretrained sequence-to-sequence models [REF4]. This approach has shown promising results, even though there are challenges when the model is forced to produce tokens in an order and context that it has not encountered during pretraining [REF4].

Scaling these models can be achieved in a variety of ways, including using a bigger model, training the model for more steps, and ensembling [REF7]. For instance, to experiment with increased model size, the guidelines of "BERT LARGE" have been followed [REF7].

In conclusion, interaction-focused systems that employ encoder-decoder models have shown promising results in the field of information retrieval. However, there is still much to explore in terms of unsupervised objectives, data usage, and scaling methods.

References sent to GTP:

REF0 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF1 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF2 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF3 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF4 - Document Ranking with a Pretrained Sequence-to-Sequence Model

REF5 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF6 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF7 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

REF8 - Language Models as Knowledge Bases?

REF9 - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

2.5 Fine-tuning Interaction-focused Systems

Interaction-focused systems, such as those used in information retrieval, can be fine-tuned using various algorithms. Some of these include Multidimensional Scaling (MDS) and Locally Linear Embedding (LLE). However, these methods often suffer from the "no right answer" problem, where it is unclear how to systematically correct an embedding that fails to capture the structure important to a user [REF0].

Our approach differs in that we learn a full metric over the input space, which allows for easier generalization to previously unseen data [REF0]. This is achieved by posing metric learning as a convex optimization problem, which allows for the derivation of efficient, local-optima free algorithms [REF2].

One application of our methods is "clustering with side information," where we learn a distance metric using similarity information and cluster data using that metric [REF3]. This approach has shown to improve clustering performance, as demonstrated on artificial and UCI datasets [REF2].

In the case of clustering problems, both K-means and constrained K-means often fail to find good clusterings [REF4]. However, by first learning a distance metric and then clustering according to that metric, we can easily separate the true clusters from each other [REF4].

The optimization problem is convex, which enables us to derive efficient, local-minima-free algorithms to solve it [REF6]. We also note that, while one might consider various alternatives, some choices would not be good despite giving a simple linear constraint [REF6].

In most problems, using a learned metric with constrained K-means outperforms using constrained K-means alone, sometimes by a very large margin [REF8]. Not surprisingly, having more side-information typically leads to metrics giving better clusterings [REF8].

In the supervised learning setting, numerous attempts have been made to define or learn either local or global metrics for classification [REF9]. While these methods often learn good metrics for classification, it is less clear whether they can be used to learn good, general metrics for other algorithms such as K-means [REF9].

References sent to GTP:

REF0 - Distance metric learning, with application to clustering with side-information

REF1 - Distance metric learning, with application to clustering with side-information

REF2 - Distance metric learning, with application to clustering with side-information

REF3 - Distance metric learning, with application to clustering with side-information

REF4 - Distance metric learning, with application to clustering with side-information

REF5 - Distance metric learning, with application to clustering with side-information

REF6 - Distance metric learning, with application to clustering with side-information

REF7 - Distance metric learning, with application to clustering with side-information

REF8 - Distance metric learning, with application to clustering with side-information

REF9 - Distance metric learning, with application to clustering with side-information

2.6 Dealing with long texts

Interaction-focused systems dealing with long texts require a strategic approach to ensure effective information retrieval. One such strategy is the use of the PARADE end-to-end document reranking model, which incorporates diverse relevance signals from the full text into ad-hoc ranking, rather than basing it on a single passage [REF0]. This model performs better when the number of relevant passages per document is low, reflecting the reduced importance of aggregating relevance signals across passages [REF0].

However, the effectiveness of such models can be influenced by the nature of the queries. For instance, queries that result in a lower number of highly relevant passages per document may reduce the advantage of using more complex aggregation methods like PARADE-Transformer and PARADE-CNN [REF1]. This is supported by the fact that TREC DL shares queries and other similarities with MS MARCO, which only has 1-2 relevant passages per document by nature of its construction [REF1].

In dealing with long texts, the document score can be determined by the score of the first passage, the best passage, or the sum of all passage scores [REF2]. This approach considers all passages from a relevant document as relevant and vice versa [REF2]. It is also important to note that the document title, when available, is added to the beginning of every passage to provide context [REF2].

Aggregation over passage representations using architectures like CNNs and transformers has been found to outperform passage score aggregation [REF3]. However, the utilization of the full-text increases memory requirements, necessitating the use of knowledge distillation to create smaller, more efficient passage representation aggregation models that remain effective [REF3].

The Verbosity Hypothesis suggests that relevant excerpts can appear at different positions in a document, and it is not necessarily possible to account for all such excerpts by considering only the top passages [REF4]. The ordering of passages itself may affect a document's relevance; a document with relevant information at the beginning is intuitively more useful than a document with the information at the end [REF4].

In the context of interaction-focused systems, several neural ranking models have been proposed, such as DSSM, DRMM, (Co-)PACRR, (Conv-)KNRM, and TK [REF5]. These models, however, have their contextual capacity limited by relying on pre-trained unigram embeddings or using short n-gram windows [REF5].

The process of dealing with long texts often involves splitting a document into passages that can be handled individually [REF6]. A sliding window of 225 tokens is applied to the document with a stride of 200 tokens, and these passages are taken as input to the BERT model for relevance estimation [REF6].

In conclusion, dealing with long texts in interaction-focused systems requires a strategic approach that incorporates diverse relevance signals, considers the nature of the queries, and effectively aggregates passage representations. This approach ensures effective information retrieval and contributes to the overall performance of the system.

References sent to GTP:

REF0 - PARADE: Passage Representation Aggregation for Document Reranking

REF1 - PARADE: Passage Representation Aggregation for Document Reranking

REF2 - Deeper Text Understanding for IR with Contextual Neural Language Modeling

REF3 - PARADE: Passage Representation Aggregation for Document Reranking

REF4 - PARADE: Passage Representation Aggregation for Document Reranking

REF5 - PARADE: Passage Representation Aggregation for Document Reranking

REF6 - PARADE: Passage Representation Aggregation for Document Reranking

REF7 - PARADE: Passage Representation Aggregation for Document Reranking

REF8 - PARADE: Passage Representation Aggregation for Document Reranking

REF9 - PARADE: Passage Representation Aggregation for Document Reranking

3 Representation-focused Systems

3.1 Single Representations

Representation-focused systems, particularly single representation systems, are a crucial aspect of information retrieval. These systems are designed to match and replace each gold passage with the corresponding passage in the candidate pool, discarding questions when the matching fails due to different Wikipedia versions or pre-processing [REF0].

Dense retrieval, a type of representation-focused system, has been explored for open-domain question answering (QA). It retrieves relevant passages iteratively using reformulated question vectors [REF1]. Dense retrieval systems can map synonyms or paraphrases consisting of completely different tokens to vectors close to each other, providing a task-specific representation [REF2].

The Dense Passage Retrieval (DPR) model, for instance, is trained using the in-batch negative setting with a batch size of 128 and one additional BM25 negative passage per question [REF0]. This model is trained for up to 40 epochs for large datasets and 100 epochs for small datasets, with a learning rate of 10^-5 using Adam, linear scheduling with warm-up, and a dropout rate of 0.1 [REF0].

A recent advancement in dense retrieval, the Approximate Nearest Neighbor Negative Contrastive Estimation (ANCE), reduces the variance of the stochastic gradient estimation, leading to faster learning convergence [REF3]. ANCE uses an asynchronously updated ANN index of the corpus representation, which is refreshed once it finishes to keep up with the model training [REF3].

Another approach, ADORE, improves ranking performance by mapping the query closer to the relevant documents [REF5]. It uses the document encoder trained by BM25 Neg and further trains the query encoder [REF5]. ADORE can achieve better performance through end-to-end training, optimizing the ranking performance for different compression techniques [REF9].

However, it is generally believed that learning a good dense vector representation needs a large number of labeled pairs of question and contexts [REF2]. Despite this, dense retrieval methods have shown to outperform TF-IDF/BM25 for open-domain QA [REF2].

In conclusion, representation-focused systems, particularly single representation systems, are a crucial aspect of information retrieval. They provide a task-specific representation, improve ranking performance, and optimize the ranking performance for different compression techniques.

References sent to GTP:

REF0 - Dense Passage Retrieval for Open-Domain Question Answering

REF1 - Dense Passage Retrieval for Open-Domain Question Answering

REF2 - Dense Passage Retrieval for Open-Domain Question Answering

REF3 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF4 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF5 - Optimizing Dense Retrieval Model Training with Hard Negatives

REF6 - Optimizing Dense Retrieval Model Training with Hard Negatives

REF7 - Optimizing Dense Retrieval Model Training with Hard Negatives

REF8 - Optimizing Dense Retrieval Model Training with Hard Negatives

REF9 - Optimizing Dense Retrieval Model Training with Hard Negatives

3.2 Multiple Representations

Representation-focused systems in information retrieval often utilize multiple representations to enhance the accuracy and efficiency of the retrieval process. One such approach is the use of down-projected representations, such as the ME-BERT-k model, which uses a feed-forward layer with dimension 768 × k [REF0]. This model is a hybrid that balances the fidelity of sparse representations and the generalization of learned dense ones by linearly combining a sparse and dense system's scores [REF0].

However, the first-stage retrieval model in this pipeline can impose a strict upper bound due to recall errors [REF1]. An alternative approach is to use learned dense low-dimensional encodings of documents and queries for the first-stage retrieval [REF1]. This dual encoder model scores each document by the inner product between its encoding and that of the query, allowing it to be easily applied to large document collections [REF1].

Classic lexical retrieval systems rely on overlapping query document terms under morphological generalization like stemming, to score query document pair [REF2]. However, these systems suffer from vocabulary mismatch problems, where the query and the relevant documents use different terms for the same concept [REF3]. To address this, recent studies in neural information retrieval have shifted to soft matching between all query and document terms [REF3].

The ColBERT model proposes a novel late interaction paradigm for estimating relevance between a query and a document [REF4]. Under this paradigm, the query and document are separately encoded into two sets of contextual embeddings, and relevance is evaluated using cheap and pruning-friendly computations between both sets [REF4].

In the standard two-stage retrieval and ranking system, the first-stage retrieval from a large document collection is followed by reranking with a cross-attention model [REF5]. The ColBERT model, which employs late interaction over BERT base, performs competitively in effectiveness and is orders of magnitude cheaper than BERT base [REF6].

For retrieval from large document collections with scalable models, efficient approximate nearest neighbor search libraries like ScaNN are used [REF7]. Pretrained transformers, such as BERT-base, are also used in addition to these models [REF8]. These models employ embedding-based representations of queries and documents and directly model local interactions between their contents [REF9].

In conclusion, multiple representations in representation-focused systems enhance the efficiency and accuracy of information retrieval by addressing vocabulary mismatch problems, enabling the application to large document collections, and reducing computational costs.

References sent to GTP:

REF0 - Sparse, Dense, and Attentional Representations for Text Retrieval

REF1 - Sparse, Dense, and Attentional Representations for Text Retrieval

REF2 - COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List

REF3 - COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List

REF4 - ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

REF5 - Sparse, Dense, and Attentional Representations for Text Retrieval

REF6 - ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

REF7 - Sparse, Dense, and Attentional Representations for Text Retrieval

REF8 - Real-time Inference in Multi-sentence with Deep Pretrained Transformers

REF9 - ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

3.3 Fine-tuning Representation-focused Systems

Representation-focused systems, particularly those used in information retrieval, have seen significant advancements in recent years. One of the key areas of focus has been the fine-tuning of these systems to improve their performance and accuracy. Dense Passage Retriever (DPR), for instance, uses a dense encoder to map any text passage to a d-dimensional real-valued vector, building an index for all the passages that will be used for retrieval [REF2].

The selection of negative examples is a crucial aspect of learning a high-quality encoder. Three types of negatives are considered: random passages from the corpus, top passages returned by BM25 that don't contain the answer but match most question tokens, and positive passages paired with other questions that appear in the training set [REF3]. The best model uses gold passages from the same mini-batch and one BM25 negative passage [REF3].

The training of DPR models involves different schemes. For instance, the DPR model used in experiments was trained using the in-batch negative setting with a batch size of 128 and one additional BM25 negative passage per question [REF0]. The question and passage encoders were trained for up to 40 epochs for large datasets and 100 epochs for small datasets, with a learning rate of 10 −5 using Adam, linear scheduling with warm-up, and a dropout rate of 0.1 [REF0].

The use of dense vector representations for retrieval has a long history, with applications to cross-lingual document retrieval, ad relevance prediction, Web search, and entity retrieval [REF6]. Dense retrieval methods have been shown to outperform TF-IDF/BM25 for open-domain QA [REF5]. However, learning a good dense vector representation requires a large number of labeled pairs of question and contexts [REF5].

The performance of a dense passage retriever can be improved by training it with a small number of question-passage pairs. For instance, a DPR trained using only 1,000 examples already outperforms BM25 [REF1]. Adding more training examples consistently improves the retrieval accuracy [REF1].

In conclusion, the fine-tuning of representation-focused systems involves a careful selection of negative examples, the use of appropriate training schemes, and the use of a sufficient number of training examples. These factors contribute to the performance and accuracy of the system in retrieving relevant information.

References sent to GTP:

REF0 - Dense Passage Retrieval for Open-Domain Question Answering

REF1 - Dense Passage Retrieval for Open-Domain Question Answering

REF2 - Dense Passage Retrieval for Open-Domain Question Answering

REF3 - Dense Passage Retrieval for Open-Domain Question Answering

REF4 - Dense Passage Retrieval for Open-Domain Question Answering

REF5 - Dense Passage Retrieval for Open-Domain Question Answering

REF6 - Dense Passage Retrieval for Open-Domain Question Answering

REF7 - Dense Passage Retrieval for Open-Domain Question Answering

REF8 - Dense Passage Retrieval for Open-Domain Question Answering

REF9 - Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

4 Retrieval Architectures and Vector Search

4.1 MIP and NN Search Problems

Locality Sensitive Hashing (LSH) is a widely used tool for approximate nearest neighbor search, where a random mapping from objects to a small, possibly binary, alphabet is used. The collision probabilities of this mapping are related to the desired notion of similarity between objects [REF0]. Recent studies have explored the power of asymmetry in LSH and binary hashing, where two different mappings are used to approximate similarity [REF1].

However, the exact regimes under which LSH-based methods are superior to tree-based methods and vice versa are not fully established yet [REF1]. In the context of Maximum Inner Product Search (MIPS) problems, the relevant threshold is the maximal inner product, which typically varies with the query. Therefore, it is desirable to have a single hash that works for all thresholds [REF2].

An asymmetric hash for a pair of spaces X and Y is a joint distribution over pairs of mappings. The asymmetric hashes we consider will be specified by a pair of deterministic mappings P: X → Z and Q: Y → Z and a single random mapping h: Z → Γ, where f(x) = h(P(x)) and g(y) = h(Q(y)) [REF3].

However, it has been suggested that asymmetry might not be required nor helpful for MIPS. When queries are normalized and data is bounded, a symmetric LSH is possible and there is no need for asymmetry. But when queries and data vectors are bounded and queries are not normalized, the power of asymmetry is observed: here, a symmetric LSH is not possible, but an asymmetric LSH exists [REF4].

In the MIPS setting, an asymmetric hash, as defined here, is not needed, but an asymmetric view of the problem is required. To use a symmetric hash, one must normalize the queries but not the database vectors, which can legitimately be viewed as an asymmetric operation which is part of the hash [REF7].

Shrivastava and Li (2014a) established the existence of an asymmetric LSH, referred to as L2-ALSH(SL), over this pair of database and query spaces. However, there does exist a simple, parameter-free, universal, symmetric LSH, referred to as SIMPLE-LSH, over X • , Y • . This suggests that we do need to consider the hashing property asymmetrically, but the same hash function can be used for both the database and the queries and there is no need for two different hash functions or two different mappings P (•) and Q(•) [REF9].

References sent to GTP:

REF0 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF1 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF2 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF3 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF4 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF5 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF6 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF7 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF8 - On Symmetric and Asymmetric LSHs for Inner Product Search

REF9 - On Symmetric and Asymmetric LSHs for Inner Product Search

4.2 Locality sensitive hashing approaches

Locality sensitive hashing (LSH) is a powerful tool for high-dimensional similarity search, but it often requires a large number of hash tables to achieve desired search accuracy and query time. The multi-probe LSH method, however, offers a more space-efficient solution. This approach uses carefully derived probing sequences to systematically probe multiple hash buckets, reducing the number of hash tables required by a significant factor [REF0].

The multi-probe LSH method is based on the concept of a probing sequence. The sequence is designed to check multiple buckets that are likely to contain the nearest neighbors of a query object [REF3]. Two types of probing sequences are commonly used: the step-wise probing sequence and the query-directed probing sequence. The step-wise probing method probes all the 1-step buckets first, then all the 2-step buckets, and so on [REF2]. The query-directed probing sequence, on the other hand, requires significantly fewer hash tables and probes to achieve the same recall precisions [REF1].

The efficiency of the multi-probe LSH method can be further improved by generating perturbation vectors in a more efficient way. Instead of generating all perturbation vectors, which can be wasteful, the method generates only those with low scores, represented by the non-zero coordinates as a set of (i, δi) pairs [REF6].

However, the multi-probe LSH method is not without its drawbacks. For instance, the sampling process can be inefficient due to the slow computation of perturbing points and their hash values. It also tends to generate duplicate buckets, leading to wasteful computation [REF8].

Despite these challenges, the multi-probe LSH method has been proven to work well on data that is extremely high-dimensional but sparse. This property is not shared by other known spatial data structures, making the multi-probe LSH method particularly useful for applications such as fast color-based image similarity search [REF4].

In conclusion, the multi-probe LSH method offers a more efficient approach to high-dimensional similarity search. It reduces the number of hash tables and probes required, and works well on sparse, high-dimensional data. However, further research is needed to address its drawbacks and improve its efficiency [REF9].

References sent to GTP:

REF0 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF1 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF2 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF3 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF4 - Locality-Sensitive Hashing Scheme Based on p-Stable Distributions

REF5 - Locality-Sensitive Hashing Scheme Based on p-Stable Distributions

REF6 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF7 - Locality-Sensitive Hashing Scheme Based on p-Stable Distributions

REF8 - Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search

REF9 - Locality-Sensitive Hashing Scheme Based on p-Stable Distributions

4.3 Vector quantisation approaches

Vector quantisation is a critical aspect of retrieval architectures and vector search. It is a process that operates on vectors instead of scalars, an extension of the simple scalar quantizers [REF9]. The primary goal of vector quantisation is to minimise the quantisation error to obtain precise distances, which necessitates a large number of centroids, for instance, k = 2^64 for 64-bit codes [REF0]. However, this requirement poses several challenges, including the need for a large number of samples to learn the quantizer, the complexity of the algorithm, and the prohibitive memory requirements for storing the centroids [REF0].

To address these challenges, various strategies have been proposed. One such strategy is the hierarchical k-means (HKM), which enhances the efficiency of the learning stage and the corresponding assignment procedure [REF0]. Another approach involves operating on the signal before quantisation to make the process more efficient. This pre-processing aims to remove redundancy in the signal, reduce signal variance, or concentrate the signal energy, all of which can enhance performance for a given bit rate and complexity [REF1].

The use of Hamming space is another strategy aimed at computing distances efficiently [REF2]. This approach involves using table lookups, which results in comparable efficiency. However, comparing the query vector with all codes is prohibitive for very large datasets. Therefore, a modified inverted file structure is introduced to rapidly access the most relevant vectors [REF2]. This approach is significantly more efficient than SDC and ADC on large datasets, as it only compares the query to a small fraction of the database vectors [REF3].

The quality of a quantizer is usually measured by the mean squared error between the input vector x and its reproduction value q(x) [REF4]. For the quantizer to be optimal, it has to satisfy two properties known as the Lloyd optimality conditions. First, a vector x must be quantized to its nearest codebook centroid, in terms of the Euclidean distance [REF4].

The product quantizer is another approach that has shown promising results. It creates the subvectors by splitting the input vector according to the order of the components [REF3]. This approach has been found to be more efficient than SDC and ADC on large datasets [REF3]. However, the choice of parameters for the product quantizer is crucial for optimal results [REF6].

In conclusion, vector quantisation is a complex but essential aspect of retrieval architectures and vector search. Various strategies and approaches have been proposed to address the challenges associated with this process, each with its strengths and limitations. The choice of the most appropriate approach depends on the specific requirements of the dataset and the constraints of the system.

References sent to GTP:

REF0 - Product Quantization for Nearest Neighbor Search

REF1 - Vector Quantization and Signal Compression

REF2 - Product Quantization for Nearest Neighbor Search

REF3 - Product Quantization for Nearest Neighbor Search

REF4 - Product Quantization for Nearest Neighbor Search

REF5 - Product Quantization for Nearest Neighbor Search

REF6 - Product Quantization for Nearest Neighbor Search

REF7 - Product Quantization for Nearest Neighbor Search

REF8 - Vector Quantization and Signal Compression

REF9 - Vector Quantization and Signal Compression

4.4 Graph approaches

The process of information retrieval can be significantly improved by employing graph-based approaches. One such approach is the use of a navigable small world network topology, which is based on a greedy search algorithm for the approximate k-nearest neighbor search problem [REF3]. This algorithm iteratively explores the neighborhood of the closest elements in a greedy manner, improving the known k closest elements at each step [REF0].

The graph contains an approximation of the Delaunay graph and has long-range links together with the small-world navigation property. This allows the algorithm to choose the accuracy of search without modification of the structure [REF3]. The algorithm's performance can be further enhanced by gradually shrinking the search radius through iterations, a concept similar to decentralized search of small-world networks [REF4].

However, the navigability criterion, inspired by Kleinberg's study of random Watts-Strogatz networks, can be extended for more general spaces, but it requires knowledge of the data distribution beforehand [REF1]. Moreover, greedy routing in Kleinberg's graphs suffers from polylogarithmic complexity scalability at best [REF1].

An alternative approach is the use of the NSW algorithm, which allows decentralized graph construction and is suitable for data in arbitrary spaces [REF5]. This algorithm uses a simpler, previously unknown model of navigable networks. However, it also suffers from the polylogarithmic search complexity of the routing process [REF5].

To optimize the performance of the controllable hierarchy, the overlap between neighbors on different layers should be minimal. This can be achieved by decreasing the m L parameter, which, however, increases the average hop number during a greedy search on each layer, negatively affecting the performance [REF6].

In terms of practical implementation, our method can iteratively improve the graph in place, demonstrating high recall rates with each point comparing only to several percent of the whole dataset on average [REF2]. Moreover, it is easy to implement, with a single-node implementation taking less than 200 lines of C++ code [REF2].

In conclusion, graph-based approaches to information retrieval offer significant advantages in terms of performance and scalability. However, they also present challenges, such as the need for knowledge of the data distribution and the complexity of the routing process. Despite these challenges, these approaches represent a promising direction for future research and development in the field of information retrieval.

References sent to GTP:

REF0 - Approximate Nearest Neighbor Algorithm based on Navigable Small World Graphs

REF1 - Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs

REF2 - Efficient K-Nearest Neighbor Graph Construction for Generic Similarity Measures

REF3 - Approximate Nearest Neighbor Algorithm based on Navigable Small World Graphs

REF4 - Efficient K-Nearest Neighbor Graph Construction for Generic Similarity Measures

REF5 - Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs

REF6 - Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs

REF7 - Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs

REF8 - Approximate Nearest Neighbor Algorithm based on Navigable Small World Graphs

REF9 - Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs

4.5 Optimisations

Retrieval architectures and vector search optimisations are crucial in the field of information retrieval. The approaches to these can generally be divided into two categories: tree-based and product quantization (PQ) based ones [REF0]. Tree-based approaches often require special approximate training techniques, which can slow down their adoption due to their complexity. On the other hand, existing PQ based approaches are typically designed for smaller computer vision tasks and may not be applicable to large scale information retrieval tasks with millions of items [REF0].

Dense retrieval methods often differ significantly from sparse retrievals and typically retrieve many new documents [REF1]. These methods have a low overlap with traditional retrieval methods such as BM25, suggesting that dense retrieval methods might benefit more from optimisations [REF1].

In terms of vector search, the use of cosine as a vector similarity function and L2-based retrieval has been found to be effective [REF2]. The number of partitions and the nearest to each query embedding are also important considerations in optimising retrieval [REF2].

The Dense Passage Retrieval (DPR) model is another approach that has been used in retrieval architectures [REF3]. This model is trained using the in-batch negative setting and has been found to work well across different datasets [REF3]. The Approximate Nearest Neighbor Negative Contrastive Estimation (ANCE) is another method that focuses on representation learning for dense retrieval [REF4]. This method constructs training negatives globally from the entire corpus, which has been shown to improve retrieval accuracy [REF4].

The use of contextualized token embeddings and late-interaction operations in retrieval architectures such as ColBERT has also been found to be effective [REF6]. However, these models often have a large index size due to the storage of token-level representations [REF6].

In terms of optimisations, the use of the LAMB optimizer and linear warm-up and decay after a certain number of steps has been found to be effective [REF7]. The ANCE method has also been found to converge in about 10 epochs, similar to other dense retrieval baselines [REF7].

However, there are still challenges in retrieval architectures and vector search optimisations. For instance, dense retrieval methods can sometimes retrieve documents that are related but not exactly relevant to the query [REF8]. This can be due to the lack of domain knowledge in the pretrained language model [REF8].

In conclusion, while there have been significant advancements in retrieval architectures and vector search optimisations, there are still challenges to be addressed. These include the complexity of tree-based approaches, the limitations of PQ based approaches for large scale tasks, and the retrieval of irrelevant documents by dense retrieval methods.

References sent to GTP:

REF0 - Joint Learning Deep Retrieval Model and Product Quantization based Embedding Index

REF1 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF2 - ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

REF3 - Dense Passage Retrieval for Open-Domain Question Answering

REF4 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF5 - Billion-Scale Similarity Search with GPUs

REF6 - Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance

REF7 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF8 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

REF9 - Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval

5 Learned Sparse Retrieval

5.1 Document expansion learning

The concept of Learned Sparse Retrieval is a significant development in the field of information retrieval. It focuses on the expansion of documents to improve the retrieval process. One of the key aspects of this process is the diagnosis of term mismatch, which is a common problem in search queries. This diagnosis guides the interactive query expansion, creating Boolean conjunctive normal form (CNF) structured queries that selectively expand 'problem' query terms while leaving the rest of the query untouched [REF0][REF2].

The term mismatch probability can be estimated reliably prior to retrieval and is typically used in probabilistic retrieval models to provide query dependent term weights [REF2]. This approach has been shown to reduce user effort by 33% and produce simple and effective structured queries that surpass their bag of word counterparts [REF0][REF2].

However, the process of document expansion can be time-consuming and resource-intensive, especially for large-scale information retrieval applications such as web search. For instance, the docT5query, a T5-based sequence-to-sequence generative model, requires significant computational resources to expand the whole MS MARCO passage collection [REF1].

To address this issue, a new method of passage expansion has been introduced, replacing docT5query with the original TILDE model. This method has been shown to be 45 times faster than docT5query, with less than 1% effectiveness loss [REF1].

The term mismatch problem and the P (t|R) prediction problem are both long-standing and central problems in retrieval. This research takes an entirely new approach to the problem, and successfully demonstrates several uses of this new tool in improving retrieval [REF6].

The diagnosis based on predicted P (t|R) selects the query terms with the lowest P (t|R) probabilities first. This dissertation develops the best known method to predict P (t|R), being the first to use query dependent features for prediction [REF9].

In conclusion, the Learned Sparse Retrieval - Document expansion learning approach provides a promising solution to the term mismatch problem in information retrieval. It offers a more efficient and effective method for document expansion, which can significantly improve the retrieval process [REF1][REF6].

References sent to GTP:

REF0 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF1 - Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion

REF2 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF3 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF4 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF5 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF6 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF7 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF8 - Modeling and Solving Term Mismatch for Full-Text Retrieval

REF9 - Modeling and Solving Term Mismatch for Full-Text Retrieval

5.2 Impact score learning

Learned sparse retrieval has been a significant focus in recent years, with models such as uniCOIL and DeepCT leading the way. The uniCOIL model, for instance, has demonstrated its effectiveness in sparse retrieval using learned impact weights, outperforming other models like DeepImpact[REF0]. This model is directly compatible with inverted indexes, and its effectiveness is enhanced with the addition of doc2query-T5, which negates the need for specialized retrieval infrastructure[REF0].

The concept of learned sparse retrieval is based on the idea of projecting the learned knowledge back into the sparse vocabulary space, which allows for the reuse of decades of innovation in inverted indexes and efficient query evaluation algorithms[REF1]. This approach has been shown to be effective, but the trade-offs between output quality, time, and space are not yet fully understood[REF1].

DeepCT, another prominent example of learned sparse retrieval, uses a transformer to learn term weights based on a regression model[REF2]. However, it only assigns weights to terms that are already present in the document, limiting retrieval to exact match[REF2]. This limitation is addressed by the use of dense representations, which are capable of capturing semantic matches[REF2].

The use of BM25 ranking over a DocT5Query expanded index has also been proposed, which uses the DeepImpact ranking impacts to compute document scores[REF3]. This approach has shown promise in improving the retrieval effectiveness of learned sparse models[REF3].

Contextualized neural language models have been proposed to address the issue of capturing context in word embedding based approaches[REF4]. These models, such as BERT, have been used to predict query-document relevance scores[REF4]. DeepCT, for instance, has been trained using a BERT model, which has shown to improve first-stage retrieval accuracy[REF5].

The process of learned sparse retrieval can be likened to a conceptual cascade, where each layer aims to filter or rank documents before passing onto the next layer[REF6]. The bottom layer may filter a collection of documents down to the ones that should be scored, and the top layer may use additional ranking features to re-rank the documents[REF6].

DeepCT is also capable of identifying terms that are central to the topic of the text, emphasizing central terms and suppressing non-central terms[REF7]. This ability to recognize the relative term importance in a passage is a significant advantage of learned sparse retrieval[REF7].

Other models like TILDEv2 have also been employed as effective first-stage rankers[REF8]. The use of knowledge distillation has been proposed to increase model accuracy without affecting retrieval latency[REF8]. However, the effectiveness-efficiency trade-off of these models, like that of the BERT re-ranker, is still a subject of ongoing research[REF9].

References sent to GTP:

REF0 - A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques

REF1 - A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques

REF2 - A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques

REF3 - Fast Learned Sparse Retrieval with Guided Traversal

REF4 - Context-Aware Sentence/Passage Term Importance Estimation for First Stage Retrieval

REF5 - Context-Aware Sentence/Passage Term Importance Estimation for First Stage Retrieval

REF6 - Efficient Query Processing for Scalable Web Search

REF7 - Context-Aware Sentence/Passage Term Importance Estimation for First Stage Retrieval

REF8 - Fast Learned Sparse Retrieval with Guided Traversal

REF9 - Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion

5.3 Sparse representation learning

Learned Sparse Retrieval - Sparse representation learning

Sparse representation learning is a fundamental aspect of learned sparse retrieval techniques in information retrieval systems. These techniques aim to efficiently represent and retrieve relevant information from large datasets while minimizing storage requirements and query-time latency. Various approaches have been proposed in the literature, leveraging different algorithms and data structures to achieve sparse representation learning.

One common approach is based on random projections [REF0]. This technique involves projecting high-dimensional data onto lower-dimensional spaces using random matrices. This approach has been shown to be effective in reducing the dimensionality of the data while preserving its discriminative properties.

Another approach is based on constructing efficient search graphs by finding clusters in the data [REF0]. Navigable small world graphs (NSW) and hierarchical NSW (HNSW) are examples of such techniques. These methods aim to organize the data into clusters, allowing for efficient retrieval by navigating through the graph structure.

Product quantization (PQ) is another popular approach in sparse representation learning [REF0]. PQ decomposes the original space into a cartesian product of low-dimensional subspaces and quantizes each of them separately. This technique has been shown to be effective in reducing storage requirements while maintaining retrieval accuracy.

Spectral hashing is a technique that involves computing an optimal binary hash for data points [REF0]. However, this problem is NP-hard, and thus, a relaxation to continuous valued hashes is often used. Spectral hashing leverages the spectrum of the similarity matrix to find an approximate solution to the optimization problem.

To enhance the efficiency of sparse representation learning, various carefully chosen data structures are employed [REF0]. These include hashes, locality-sensitive hashes, inverted file structures, trees, clustering, and quantization sketches. These data structures enable faster query times and compact representations of the data.

In recent years, there has been a growing interest in learning the structure of approximate nearest neighbor (ANN) graphs end-to-end [REF0]. This approach involves training models to learn binary representations or hash codes that optimize the similarity metric. Margin-based triplet loss and imposing independence and orthogonality conditions on hash bits are examples of techniques used in this context [REF0].

Overall, learned sparse retrieval techniques leverage various algorithms and data structures to achieve efficient and effective information retrieval. These techniques aim to reduce storage requirements, minimize query-time latency, and improve ranking effectiveness. By combining sparse representation learning with end-to-end learning of ANN structures, researchers have made significant advancements in the field of information retrieval.

References sent to GTP:

REF0 - Minimizing FLOPS to Learn Efficient Sparse Representations

REF1 - Expansion via Prediction of Importance with Contextualization

REF2 - Minimizing FLOPS to Learn Efficient Sparse Representations

REF3 - Expansion via Prediction of Importance with Contextualization

REF4 - SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking

REF5 - SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking

REF6 - SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking

REF7 - Minimizing FLOPS to Learn Efficient Sparse Representations

REF8 - Minimizing FLOPS to Learn Efficient Sparse Representations

REF9 - Minimizing FLOPS to Learn Efficient Sparse Representations