A multi-stage pipeline to automatically generate surveys

Leandro Carísio Fernandes and Gustavo Bartz Guedes

June 2023

Abstract

We propose an architecture and a reference implementation of a multi-stage pipeline for survey generation. TODO TODO TODO

1 Introduction

In the scientific community knowledge dissemination is performed by the publication of articles, commonly referred as papers. With the advent of Internet, papers are now available in online repositories.

TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

This article presents two contributions: it proposes the architecture of a three-stage pipeline for automatic survey generation and provides a reference implementation for this architecture.

2 Proposed Pipeline

Figure 1 shows the general architecture of our three-stage pipeline to automatically generate surveys. The main idea is that the user wants a survey on a specific topic (SURVEY_TOPIC_QUERY) considering only papers that matches some criteria (SURVEY_FILTERS) (e.g. restricted to some field of study, years, or venue). With these inputs we propose a pipeline to retrieve papers, to aggregate them, and to suggest sections titles and text contents.

These input are sent to the "Retriever", which accesses a scientific article search engine and returns a set of candidate articles (all_papers) to be used in the survey.

The output of the Retriever is sent to the second stage, "Sections discover", which also receive the desired structure of the survey (number of sections, subsections per section, and so on). The second stage organizes the papers returned by the Retriever in this structure and outputs a list of related papers for each section and a suggestion of the title for the section. The following algorithm is used:

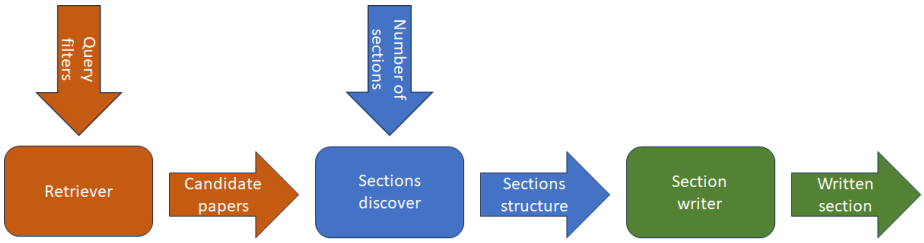

Figure 1: Architecture of a multi-stage pipeline to automatically generate surveys. TODO: Lembrar de uniformizar os modelos das figuras - esquema de cores etc

1. The number of sections is represented by the variable $n\_sections$, a list where the first element is the number of sections, the second is the number of subsections per section and so on.

2. The structure of the survey is represented by the variable $sections$, a list of sections, which is initialized according to $n\_sections$. Each of its elements has at least three attributes: $section\_title$ (the title of the section), $papers$ (a list of related papers of the section), and $subsections$ (a list of subsections of the section, also defined by $n\_sections$).

3. To populate $sections$, a recursive function is used. This is done using three inputs: the number of sections in a certain level ($n\_sec\_in\_level$, which is extracted from $n\_sections$), the related papers in this level ($papers$), and some text ($query$) representing a main query that is related to $papers$. Using a clustering method, the $papers$ are split into $n\_sec\_in\_level$ groups of papers ($papers\_in\_section$). Each cluster represents a $section$ in $sections$. To populate this section we use the algorithm:

   (a) Select the top_{k1} closest papers to the center of the cluster and send as input to a generative LLM to get the title of the section.

   (b) Generate the embeddings of $papers\_in\_section$ and store them in a vector database.

   (c) Search for the full name of the section in the vector database.

   (d) If this section does not have subsections, consider that the top_{k2} papers retrieved will be used as related papers of the section.

   (e) If this section has subsections, select the top_{k3} papers retrieved and recursively use them to finde the subsections.

4. In the first level, all the papers received by the Retriever are used by the clustering method. In the next levels, only the top_{k3} papers retrieved by the vector database are used.

The last stage if the "Section writer". It receives the previous structure and writes the contents of each section using the following algorithm for each section:

1. For each paper of the related papers of the section, the "Section writer" gets its contents, splits into chunks of text, and generate their embeddings.
2. The embeddings of the chunks of text are stored in a vector database, which is queried by the section title.
3. Using a reranker, the most relevant bunch of chunks are selected and sent to a generative LLM to write the section.

3 Reference implementation

In section 2 we showed the architecture of the proposed pipeline. In this section, we describe in details a reference implementation for each stage.

3.1 Retriever

In this reference implementation, the Retriever is an HTTP call to the Semantic Scholar API [5]. The initial query and filter criteria must sufficiently specify the survey to bring a significant and relevant number of papers. The entire pipeline relies on this initial output. If the term is too broad, it will result in the retrieval of papers from general areas, leading to sections that possibly is irrelevant for the user's intention.

3.2 Sections discover

The "Sections discover" stage generates the embeddings of the candidates papers using Specter 2.0 [2], a method to generate document-level embedding of scientific documents using their title and abstract.

The embeddings are stored in a vector database using Faiss [4] and split into clusters using the K-means algorithm. The top_{k1} (user-defined parameter) closest papers to the center of the cluster are selected and their title and abstract are used by a generative LLM to get a suggested section title. The GPT [1] chat model is used with the following prompt:

> [System]
>
> You are a renowned scientist who is writing a survey on '{SURVEY_TOPIC_QUERY}'. You are currently writing a section about '{section_title}'

[Human]

I will send you a list of title and abstract of scientific articles. Most of them cover a specific topic about the section '{query_section}'.

Your task is to find out what this topic is and suggest a good title for a section in a scientific survey that addresses it.

You should also explain your reasoning.

Your answer should be a valid RFC8259 compliant JSON object with three properties:

The first property, called "topic", describes the main topic that is a subset of '{query_section}' that are present in most abstracts.

The second property, called "title", is the title of the section that will cover this topic and must be clearly related to the property "topic".

The third property is called "reasoning" and should contains your reasoning to choose this topic as an answer.

Use this format:

```
{
  "topic": {TOPIC},
  "title": {TITLE},
  "reasoning": {REASONING}
}
```

Do you understand?

[AI]

Sure, send me the list and I will give you what you need.

[Human]

Title: {TITLE OF PAPER 1}
Abstract: {ABSTRACT OF PAPER 1}

...

Title: {TITLE OF PAPER TOP_K1}
Abstract: {ABSTRACT OF PAPER TOP_K1}

Note that the maximum value of top_{k1} depends on the context window length of the LLM used.

Using the generated title of the section, if the section doesn't have subsections, the "Sections discover" stage selects the top_{k2} papers for the next stage (text generator). Keep in mind that this filtering is based only on the title

and abstract of the papers. top_{k2} is the maximum number of related papers in a section, i.e. is the maximum number of papers that can be referenced in a section.

If the section has subsections, the top_{k3} papers are used to create clusters for the subsections. This value should be greater than top_{k2} and smaller than the total of papers retrieved in the first stage).

Both top_{k2} and top_{k3} depends on the number of papers returned by the Retriever. If the Retriever returns 1,000 papers, top_{k3} can be set to 100. But if the Retriever returns only 100 papers, top_{k3} should be much smaller (and so do top_{k2}).

3.3 Section writer

The "Section writer" stage receives, for each section, a title and a list of papers related to the section. The first step is to download the available papers and extract its contents. Each paper is split into chunks of text using containing 7 sentences (stride of 2 sentences).

The embeddings of each chunk are generated using Specter, which are stored in a vector database [4] and queried by the title of the section. The top 20 (configurable parameter) chunks of texts are submitted to GPT [1] to assign a score from 0 to 5 to the chunk of text (point-wise score). We used the following prompt:

> [System]
>
> You are a renowned scientist who is writing the section '{title_section}' of a survey entitled '{title_survey}'.
>
> [Human]
>
> I've found a text excerpt from a scientifc article that might be useful for the section '{section_title}' of your survey about '{survey_topic}'.
> Your task is to generate a score for it ranging from 0 to 5 indicating its importance to the section that you are writing.
> The score of a text written in a language other than English must be 0.
> If the text contains only metadata of an article (title, authors, venue, etc), i.e. if the text belongs to the references session, the score should be 0.
> You should also explain why you choose this score.
> Your answer MUST be enclosed in a RFC8259 compliant JSON object with two properties, "score" and "reasoning", containing the score and the reasoning for it. Remember to enclose the value of the reasoning property in quotes.
> Do not answer with anything besides the JSON object. Do not insert any text before or after the RFC8259 compliant JSON object.

Use the following format to answer (write the reasoning property first and than write the score):

```
{
  "score": {SCORE},
  "reasoning": {REASONING}
}
```

[AI]

Sure, send me the text I will give you what you need. I will answer with only a JSON object containg the score and the reasoning.

[Human]

{chunk_of_text}

Note that this prompt might also filter chunks of text with undesired characteristc. These chunks are reordered based on the new score and the first 10 chunks (configurable parameter) are sent again to GPT, who writes the text of the section using the prompt:

[System]

You are a renowned scientist who is writing a survey entitled '{survey_title}'.

[Human]

Your task is to write the text of the section '{section_title}' of the survey. To complete this task, I will give you a list of documents that should be used as references. Each document has a text and an alphanumeric ID.

When writing the section, you MUST follow this rules:

- be aware of plagiarism, i.e., you should not copy the text, but use them as inspiration.
- when using some reference, you must cite it right after its use. You should use the IEEE citing style (write the id of the text between square brackets).
- you are writing the paragraphs of the section. You MUST write only this section.
- you MUST NOT split the section in subsections, nor create introduction and conclusion for it.
- DO NOT write any conclusion in any form for the subsection.

- DO NOT write a references section.
- DO NOT begin the text writing that the context is 'Neural Information Retrieval', as this is obvious from the title of the survey.
Do you understand your task?

[AI]

Sure, send me a list of text and I will write a section about '{section_title}' using them as references. I am aware that I should use the IEEE citing style.

[Human]

ID: {REF0}
Text: {CHUNK_0}

...

ID: {REF9}
Text: {CHUNK_9}

Depending on the size of the context window of the generative LLM used to write the text, the amount of chunks to rerank and to be sent to the LLM might change.

Figure 2 summarizes the reference implementation.

4 Experiments

Two experiments were conducted. The first one is tests the third stage (text generator), comparing the generated text with a published survey about neural information retrieval - "Lecture Notes on Neural Information Retrieval" [6]. The second experiment tests the complete pipeline, generating all the text on the same topic from scratch.

The second experiment tests the generation of titles for sections..

4.1 Experiment 1 - Testing the generation of the contents of sections

To compare the text generation of the sections, let's disregard the article search and section title generation. Instead, we will use the same papers in each section as the reference survey. That is necessary for comparison because the generated text is based on chunks of text related to the articles in the section. Different

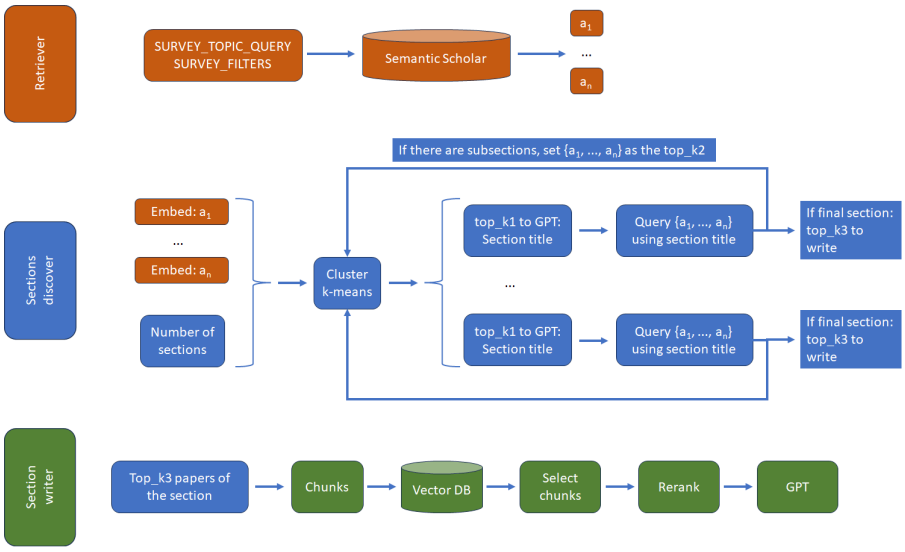

Figure 2: Reference implementation.

queries and filters will yield different sets of articles, which will inevitably generate texts addressing different topics. This approach allows us to focus on testing the steps of extracting chunks from the text, generating embeddings, performing searches using a vector database, re-ranking the results, and generating the section text.

Appendix A shows the generated survey using the gpt-3.5-turbo-0613 model. Comparing the generated text with the text from the reference survey reveals several similarities and differences.

Regarding the similarities, since the same articles were used in the text generation, it is evident that in many sections, the generated text addressed topics that were also covered in the reference survey. For example, consider the section "Interaction-focused Systems / Pre-trained Language Models." The reference survey began by discussing the transformer model and the self-attention mechanism. It then delved into various tasks where these models can be applied and, within the context of neural information retrieval, mentioned encoder-only and encoder-decoder models, providing examples of BERT and BART. The section concluded by discussing fine-tuning. Similarly, the generated text (A.2.2) also covered BERT and BART, although without categorizing them as encoder-only and encoder-decoder models. It presented applications and commented on fine-tuning as well.

One difference that occurred in some sections is that the generated text addressed different issues from those covered in the reference survey. This is likely because the search in the vector database for text chunks is performed using the section title. Since the reference survey has very generic titles, it is

natural that the vector database retrieval may also include unrelated questions, approaching the subject from a different angle.

An example of this occurred in the first section of the article, "Text Representations for Ranking / BOW Encodings." Both the reference survey and the generated text provided the definition of bag-of-words, but while the former focused on a mathematical definition, the generated text (A.1.1) focused on a textual definition and its characteristics (advantages and limitations).

Overall, there is a convergence of ideas in several of the generated sections. Using BERTScore [7] to compare the generated text with the reference survey we obtained precision, recall, and F1 of, respectively, 78.2%, 81.4%, and 79.8%.

However, there are also errors and hallucinations. For example, the section "Retrieval Architectures and Vector Search / MIP and NN Search Problems" covers two topics: maximum inner product (MIP) search and nearest neighbor (NN) search. The generated text (A.4.2) treated MIP as "Multi-Index Hashing" (a hallucination, as there is only one article linked to this section, and there is no chunk of text with that term). There are still hallucinations in the generation of fictitious references (e.g. A.1.1).

4.2 Experiment 2 - Generating sections titles

In the previous section, we evaluated the generated text while keeping the section title and the related articles fixed. This was done to isolate the effect of the received articles and the section title. There is no single rule regarding what should or should not be included in a survey, as it is a choice made by the authors.

Therefore, in this section, we will discuss the effect of initial search criteria on the generation of section titles.

Based on the topic of "neural information retrieval," let's generate some section title suggestions. The goal is to address this topic in the context of textual information retrieval, similar to the reference survey [6].

Considering an initial search for SURVEY_TOPIC_QUERY="neural information retrieval" in the field of "Computer Science" from the years 2020-2023, Semantic Scholar returns nearly 28,000 open access papers, allowing to access the metadata of the first 9,999 papers.

Assuming a structure of 3 sections with 2 subsections per section and $top_{k1} = 10$ and $top_{k3} = 100$, the first two stages of the pipeline suggest the following titles for the sections:

1. Advancements in Training and Encoding Techniques for Neural Information Retrieval
    (a) Advancements in Deep Learning-based Approaches for Information Retrieval
    (b) Neural Information Retrieval for Image Processing: Techniques and Applications
2. Advancements in Self-Supervised 6D Object Pose Estimation

   (a) Advancements in Adversarial Edge-Aware Image Colorization
   (b) Advancements in Attention-based Self-Supervised 6D Object Pose Estimation

3. Robust Normalized Softmax Loss for Deep Metric Learning-Based Characterization of Remote Sensing Images With Label Noise
   (a) Deep Learning-Based Characterization of Remote Sensing Images: A Robust Approach
   (b) Deep Learning-Based Classification of Remote Sensing Images with Label Noise

Most of the suggested titles do not have any relation to the subject. This occurs because the output of the retrieval was too broad, returning papers that are far from the desired topic. This shows that the initial research should reduce the quantity of articles, aiming to limit it to the desired subject. The query cannot be too generic.

Now let's consider another query: "large language models for textual information retrieval". This is more specific and returns less than 600 papers. Trying to group the articles in 5 sections with 2 subsections each yield:

1. Advancements in Large Language Models for Textual Information Retrieval
   (a) Advancements in Multimodal Information Retrieval for Text and Image
   (b) Advancements in Large Language Models for Domain-specific Textual Information Retrieval

2. Advancements in Large Language Models for Textual Information Retrieval
   (a) Advancements in Multimodal Information Retrieval for Text and Image
   (b) Advancements in Large Language Models for Domain-specific Textual Information Retrieval

3. UniMS: A Unified Framework for Multimodal Summarization with Knowledge Distillation
   (a) UniMS: A Unified Framework for Multimodal Summarization with Knowledge Distillation
   (b) UniMS: A Unified Framework for Multimodal Summarization with Knowledge Distillation

4. Leveraging Multimodal Information for Enhanced Information Retrieval

   (a) Enhancing Keyword Extraction with Statistical and Graph-Based Term Weighting Schemes
   (b) Leveraging Multimodal Data for Enhanced Information Retrieval
5. Document Clustering: Efficient Tool for Organizing Textual Documents
   (a) Enhancing Textual Document Retrieval through Document Clustering
   (b) Compression Techniques for Deep Learning Models in Textual Information Retrieval

Note that sections 1 and 2 are identical, which suggest that the articles are not sufficient to be grouped into five distinct clusters. Similarly, section 3 produced two identical subsections, indicating the same. Recalculating for a structure of 2 sections and 2 subsections:

1. Advancements in Large Language Models for Textual Information Retrieval
   (a) Advancements in Multimodal Information Retrieval for Text and Image
   (b) Advancements in Large Language Models for Domain-specific Textual Information Retrieval
2. Leveraging External Knowledge for Enhanced Textual Information Retrieval
   (a) Enhancing Fake News Detection with Multimodal Information
   (b) Leveraging External Knowledge for Enhanced Textual Information Retrieval

It is interesting to note that in the new grouping, the first section continues to address the same topic. The other sections have been grouped under a new title.

5 Conclusion

In this paper, we present a three-stage architecture for automatic survey generation and develop a reference implementation.

The architecture is flexible enough to be used for generating complete sections for a survey or in a sort of copilot mode, where researchers can use specific parts of the pipeline. For example, if a researcher has a basic structure in mind for the sections of their survey and the articles to be included in each section, they can use the third stage of the pipeline. Alternatively, if they already have a selection of articles on a specific topic and wants suggestions for grouping these articles with (or without) section name suggestions, the second stage can be employed.

Two experiments were conducted. The first experiment compared the text generation stage based on the sections and articles used in a reference survey. The results demonstrated that the pipeline generated text of good overall quality, covering various topics in common with the reference survey used (precision, recall, and F1 of, respectively, 78.2%, 81.4%, and 79.8%). This occurred because the same set of articles was used. The first experiment suggests that receiving a set of high-quality excerpts during the text writing stage is essential.

The second experiment tested the generation of section titles and the selection of articles to be used for text generation. As we often lack control over the search mechanism, which is often a black box (as was the case in the reference implementation), this task is challenging and requires careful consideration when selecting search criteria.

Indeed, clusterization for generating sections is based solely on the titles and abstracts of the articles. Returning a broad range of articles will inevitably result in section titles that are outside the desired topic. On the other hand, returning too few articles often hinders the clusterization process. It is a delicate balance to strike when selecting the number and relevance of articles to ensure meaningful and coherent section titles.

Therefore, we understand that although the proposed pipeline can be used for the complete generation of a survey on some topic, better results will be achieved if it is used with interventions from the researcher. For example, when writing a survey on a broader topic, better outcomes can be obtained by feeding the pipeline section by section, narrowing down the scope to handle.

6 Future Work

Despite the good results extracted from the reference implementation, specially when focusing on the "Section writer" stage, it is possible to replace various elements in the reference implementation (Figure 2) to test potential improvements in the result. For example:

- If a large number of articles are returned by the Retriever, it is possible to extract their content and re-rank them, selecting a reduced number of papers.
- The embeddings extracted for clustering were generated using Specter 2.0, which only uses the title and abstract of the articles. Other models can be tested for generating embeddings, potentially incorporating the full content of the articles as well.
- The k-means algorithm was used to generate the clusters. It is possible to test other clustering algorithms.
- It is possible to test alternative prompts to generate section titles, to rerank and to generate the contents of the sections. Considering that the prompt used has a very big impact on the pipeline, this looks like a great opportunity for improvement.

- In the "Section writer" the chunks of text contains 7 sentences. Different numbers of sentences can be tested.
- The embeddings for the chunks of text were generated using Specter 2.0. Alternative methods (e.g. Splade [3]) can be tested.
- The results shown in this paper used the gpt-3.5-turbo-0613 model. Alternative models and/or versions can also be tested.

However, there is still a need to generate the text for the introduction, conclusion, and abstract of the survey. Besides, complex elements such as equations, tables, algorithms, listings, and figures were not addressed.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *ACL*, 2020.

[3] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. Splade v2: Sparse lexical and expansion model for information retrieval, 2021.

[4] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[5] Semantic Scholar. https://www.semanticscholar.org/.

[6] Nicola Tonellotto. Lecture notes on neural information retrieval, 2022.

[7] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

A Appendix - Survey generated using sections titles and papers defined by the user

This appendix shows the text generated for the sections of a survey about neural information retrievel as described in Experiment 1. The titles of the sections

are the same as the title in the survey "Lecture Notes on Neural Information Retrieval", used as a reference [6].

In the text, REF# refers to the chunk of text used. The text of the chunk was omitted to reduce the length of this paper, but it is available in the project repository.

A.1 Text Representations for Ranking

A.1.1 BOW Encodings

Bag-of-Words (BOW) encodings are a popular approach for representing text documents in information retrieval systems. In this encoding scheme, each document is represented as a vector of term frequencies, where each dimension corresponds to a unique term in the document collection. BOW encodings have been widely used in traditional retrieval models, and their application in neural information retrieval has also gained attention [REF0].

The BOW encoding scheme assumes that the order of terms in a document does not convey important information for retrieval purposes. Instead, it focuses on the frequency of occurrence of terms within a document. This approach has been motivated by the observation that the presence and frequency of certain terms can be indicative of the document's relevance to a given query [REF1].

One advantage of BOW encodings is their simplicity and efficiency in computation. The encoding process involves tokenizing the text into individual terms, removing stop words, and counting the occurrences of each term. This results in a high-dimensional vector representation of the document, where each dimension corresponds to a unique term in the collection. These vectors can then be used for various ranking algorithms, such as cosine similarity or BM25 [REF2].

The BOW encoding scheme has some limitations. One of the main challenges is the sparsity of the resulting vectors, especially when dealing with large document collections. As the number of unique terms increases, the majority of dimensions in the vector will be zero, leading to computational inefficiency and potentially affecting the retrieval performance. Various techniques, such as dimensionality reduction or term weighting, have been proposed to address this issue [REF3].

Another limitation of BOW encodings is their inability to capture the semantic relationships between terms. Since BOW representations treat each term independently, they do not consider the contextual information or the meaning of the terms within the document. This can result in a loss of important information during the retrieval process [REF4].

Despite these limitations, BOW encodings have been widely used and have shown promising results in information retrieval tasks. They provide a simple and efficient representation of text documents, allowing for fast retrieval and ranking. Moreover, BOW encodings can serve as a baseline for more advanced neural models that aim to capture the semantic relationships between terms and documents [REF5].

In conclusion, BOW encodings have been a popular choice for representing text documents in information retrieval systems. They offer simplicity and efficiency in computation, making them suitable for large-scale retrieval tasks. However, their limitations in capturing semantic relationships and handling sparsity should be taken into consideration when designing retrieval systems. Future research in neural information retrieval can explore ways to enhance BOW encodings or combine them with more advanced techniques to improve retrieval performance [REF6] [REF7] [REF8] [REF9].

A.1.2 LTR Features

Text representations play a crucial role in neural information retrieval systems, particularly in the context of ranking. The goal of ranking is to match processed queries with indexed documents effectively and efficiently [REF1]. Learning to rank (LTR) techniques have emerged as a promising approach to improve ranking models by leveraging machine learning technologies [REF1]. In this section, we discuss the use of LTR features for text representations in ranking.

One important aspect of LTR features is the learning time, which can increase with the sample size [REF0]. To address this, researchers have investigated the number of relevant documents identified for query sets when the number of retrieved documents is unconstrained [REF0]. It has been observed that precision tends to fall below a certain threshold after a certain rank, indicating that retrieving additional documents may yield minimal gains in terms of relevance [REF0]. Therefore, an original sample of a sufficient number of documents, such as 5000, has been deemed suitable for experiments [REF0].

Different approaches have been proposed for learning to rank, including pointwise, pairwise, and listwise approaches [REF1]. These approaches provide frameworks and algorithms for ranking models, each with its own theoretical properties [REF1]. For instance, PRank is a ranker based on the perceptron algorithm, which maps a feature vector to the reals using learned weights [REF4]. PRank also learns the values of increasing thresholds to determine the rank of a document [REF7]. On the other hand, LambdaRank and LambdaMART update their parameters differently, with LambdaRank updating all weights after each query and LambdaMART updating only a few parameters at a time [REF3].

Ordinal regression has also been employed in learning to rank, where the problem is cast as learning the mapping of an input vector to an ordered set of numerical ranks [REF4]. The positions of rank boundaries play a critical role in the final ranking function [REF4]. Additionally, the use of sampling has been motivated by the need for efficient application of a learned model, reducing the number of documents for which features are calculated [REF5]. This is particularly advantageous when certain features are computationally expensive [REF5].

In learning to rank, the independence and identically distributed (i.i.d) assumption does not hold for every example document in the training data [REF6]. Instead, documents associated with each query form a group, where the groups

are i.i.d but the documents within a group are not i.i.d [REF6]. Furthermore, the learning time of many LTR techniques increases with the number of documents in the sample [REF6]. Therefore, careful consideration of the sample size is necessary to balance efficiency and effectiveness [REF6].

In summary, text representations for ranking in neural information retrieval systems involve the use of LTR features. These features are designed to optimize the ranking process by considering factors such as learning time, different learning approaches, ordinal regression, and the use of sampling. By leveraging these features, researchers aim to improve the efficiency and effectiveness of ranking models in information retrieval applications.

A.1.3 Word Embeddings

Word embeddings have become a popular approach for representing text in neural information retrieval systems. Word embeddings are dense vector representations that capture semantic and syntactic relationships between words. These representations have shown promising results in various natural language processing tasks, including information retrieval and ranking.

One common method for generating word embeddings is through the use of word2vec models [REF5]. Word2vec models, such as skip-gram and continuous bag-of-words (CBOW), learn word embeddings by predicting the context words given a target word or vice versa. These models leverage the co-occurrence statistics of words in a large corpus to learn meaningful representations. The skip-gram model, for example, learns to predict the surrounding words given a target word, while the CBOW model predicts the target word given its context. These models have demonstrated the ability to capture linguistic patterns and relationships between words [REF5].

Another popular approach for generating word embeddings is the GloVe (Global Vectors) model [REF6]. GloVe directly captures the global corpus statistics by training on word-word co-occurrence counts. The model uses a weighted least squares approach to learn word vectors that exhibit meaningful substructure. GloVe has shown state-of-the-art performance on word analogy tasks and word similarity tasks [REF3]. The advantage of GloVe is that it captures both semantic and syntactic relationships between words, making it suitable for various NLP tasks, including information retrieval and ranking.

Word embeddings have been evaluated and compared in the context of ranking tasks. For example, in a study comparing different word embedding models, it was found that the SWOW-RW and SWOW-PMI models outperformed other models in predicting guesser responses [REF4]. Similarly, the associative models, which leverage free association data, were found to emphasize semantic relationships not well-represented within distributional semantic models (DSMs) trained on linguistic corpora [REF0]. These findings suggest that word embeddings derived from different models can have varying degrees of effectiveness in capturing semantic relationships and predicting relevant information for ranking tasks.

It is worth noting that the choice of word embedding model depends on

the specific requirements of the ranking task. Different models may excel in different aspects, such as capturing semantic relationships, syntactic patterns, or contextual information. Therefore, it is important to carefully evaluate and select the appropriate word embedding model based on the specific needs of the neural information retrieval system.

In conclusion, word embeddings have emerged as a powerful tool for representing text in neural information retrieval systems. Models such as word2vec and GloVe have shown promising results in capturing semantic relationships and improving ranking performance. However, the choice of word embedding model should be carefully considered based on the specific requirements of the ranking task. Further research and evaluation are needed to explore the effectiveness of different word embedding models in various neural information retrieval scenarios.

A.2 Interaction-focused Systems

A.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have gained significant attention in the field of neural information retrieval due to their ability to capture local patterns and interactions between queries and documents. In this section, we will explore the use of CNNs in interaction-focused systems for information retrieval.

One important consideration in the design of CNN-based models is the selection of hyperparameters. Limited computational resources necessitate the careful determination of hyperparameter ranges based on pilot experiments and domain insights [REF0]. For instance, the choice of the number of filters, kernel size, and pooling sizes can significantly impact the performance of the model [REF1]. Additionally, the selection of hyperparameters such as the dimensionality of the embeddings and the number of layers can also influence the model's effectiveness [REF2].

Positional information plays a crucial role in capturing the interactions between queries and documents. However, incorporating positional information in deep neural IR models is non-trivial. Models like MatchPyramid and local DUET have attempted to account for positional information by incorporating convolutional layers based on similarity matrices between queries and documents [REF1]. Despite these efforts, these models have struggled to outperform the DRMM model, indicating the challenges in effectively utilizing positional information in multi-dimensional interactions [REF1].

One of the key advantages of CNNs in neural IR is their ability to model n-grams. Traditional IR approaches treat n-grams as discrete terms, which can lead to data sparsity and an explosion in the parameter space. CNNs address this issue by learning a convolutional layer that forms n-grams from individual word embeddings, allowing for the matching of n-grams of different lengths [REF2]. This approach enables the model to capture the semantic relationships between query terms and document content, enhancing the retrieval performance [REF2].

Several interaction-focused models have been proposed in the literature. For instance, Deep Match Tree defines interactions in the product space of dependency trees, leveraging a deep neural network to make matching decisions based on these local interactions [REF3]. Match-SRNN, on the other hand, models the recursive matching structure in local interactions to capture long-distance dependencies [REF3]. These models have been evaluated on various tasks, including short text matching, community-based question answering, and paper citation matching [REF3]. However, it is important to note that most of these deep matching models are primarily designed for semantic matching problems, which differ significantly from the relevance matching problem in ad-hoc retrieval [REF3].

The application of deep learning in information retrieval has the potential to revolutionize the field. Deep neural networks have the ability to discover hidden structures and features at different levels of abstraction, which can be beneficial for IR tasks [REF4]. The success of deep learning in other domains, such as computer vision, speech recognition, and natural language processing, suggests that it can have a significant impact on IR as well [REF4].

In summary, CNNs have emerged as a powerful tool in interaction-focused systems for neural information retrieval. By effectively capturing local patterns and interactions between queries and documents, CNN-based models have shown promise in improving retrieval performance. However, challenges remain in incorporating positional information and designing models that can handle multi-dimensional interactions. Further research is needed to explore and enhance the capabilities of CNNs in neural information retrieval.

A.2.2 Pre-trained Language Models

Pre-trained language models have revolutionized the field of neural information retrieval by providing a powerful foundation for various natural language processing tasks. In recent years, there has been a surge of interest in interaction-focused systems that leverage pre-trained language models to enhance the retrieval process. These systems aim to improve the effectiveness and efficiency of information retrieval by incorporating user interactions and feedback into the retrieval process.

One prominent example of interaction-focused systems is the use of pre-trained language models such as BERT (Bidirectional Encoder Representations from Transformers) [REF1]. BERT has demonstrated remarkable performance across a wide range of tasks, including natural language inference, question answering, and sentiment analysis. Its success can be attributed to its ability to capture contextual information and generate meaningful representations of text.

The application of pre-trained language models in interaction-focused systems involves fine-tuning the models on specific retrieval tasks. This process typically involves two stages: pre-training and fine-tuning. During pre-training, the language model is trained on a large corpus of text to learn general language understanding [REF2]. Fine-tuning, on the other hand, involves training the

model on task-specific data to adapt it to the retrieval task at hand.

One key advantage of pre-trained language models is their ability to capture semantic relationships between words and phrases. This enables them to understand the context of a query and retrieve relevant information accordingly. For example, BERT has been shown to outperform previous state-of-the-art models on various benchmark datasets, achieving significant improvements in accuracy [REF1].

Another important aspect of interaction-focused systems is the incorporation of user interactions and feedback. These systems aim to leverage user feedback, such as relevance judgments or explicit feedback, to improve the retrieval process. By incorporating user interactions, the models can adapt and refine their retrieval strategies based on user preferences and information needs.

The effectiveness of interaction-focused systems using pre-trained language models has been demonstrated in various studies. For instance, researchers have explored the use of BERT in question answering systems, where user interactions play a crucial role in refining the answers provided [REF4]. By incorporating user feedback, these systems can iteratively improve the quality of the answers and provide more accurate and relevant information.

Furthermore, the use of pre-trained language models in interaction-focused systems has also shown promise in text generation tasks. Models such as BART (Bidirectional and Auto-Regressive Transformers) have been developed to generate coherent and contextually relevant text [REF2]. By fine-tuning these models on specific text generation tasks, they can generate high-quality responses to user queries, enhancing the overall user experience.

In conclusion, pre-trained language models have emerged as a powerful tool in the development of interaction-focused systems for neural information retrieval. These models, such as BERT and BART, provide a solid foundation for capturing semantic relationships and understanding user interactions. By incorporating user feedback and fine-tuning the models on specific retrieval tasks, interaction-focused systems can significantly enhance the effectiveness and efficiency of information retrieval. Future research in this area should focus on exploring novel techniques to further improve the performance of these systems and address the challenges associated with user interactions in the retrieval process.

A.2.3 Ranking with Encoder-only Models

In recent years, there has been a growing interest in developing neural information retrieval systems that focus on capturing the interaction between queries and documents. These systems aim to improve the ranking performance by leveraging the contextual information present in the query-document pairs. One approach that has gained attention is the use of encoder-only models, which learn representations of terms and capture context from the entire text [REF0].

Encoder-only models have shown promising results in ad-hoc document ranking tasks. By incorporating contextualized language models into existing neural ranking architectures, these models can effectively capture the relevance infor-

mation between queries and documents. This is achieved by using multiple similarity matrices, one for each layer of the language model. The use of contextualized embeddings in these models has been found to significantly improve ranking performance, surpassing state-of-the-art results on benchmark datasets such as Robust 2004 and WebTrack 2012-2014 [REF0].

Furthermore, the combination of encoder-only models with BERT's classification mechanism has been shown to further enhance ranking performance. This approach, known as CEDR (Contextualized Embeddings for Document Ranking), leverages the power of BERT's contextualized representations and achieves state-of-the-art results in ad-hoc document ranking tasks [REF0].

To ensure efficient computation, various techniques have been explored in the implementation of encoder-only models. For instance, reduced precision arithmetic has been applied during decoding on TPUs, resulting in minimal loss in log perplexity and no loss in BLEU scores [REF1]. Additionally, the use of residual connections between LSTM layers in the encoder and decoder networks has been found to improve gradient flow during training, enabling the training of deeper networks [REF2].

The availability of large-scale datasets has played a crucial role in advancing the field of neural information retrieval. Previously, the limited availability of datasets hindered rapid progress and hindered the comparison of different models. However, the release of datasets such as MS MARCO and TREC CAR has provided researchers with valuable resources for training data-hungry neural models [REF3]. MS MARCO, for example, comprises a large collection of anonymized questions sampled from Bing's search query logs, along with a corpus of passages for the passage ranking task [REF3].

In the context of passage re-ranking, BERT has been widely adopted as a re-ranker due to its effectiveness in estimating the relevance of candidate passages to a given query [REF5]. By feeding the query as sentence A and the passage text as sentence B, BERT can effectively score and re-rank candidate passages based on their relevance to the query [REF5].

The design of interaction-focused systems often involves arranging models as stages in a pipeline, balancing the size of the candidate set with the complexity of the model [REF7]. This approach allows for the benefits of richer models while controlling the increased inference latencies. For example, monoBERT and duoBERT, which are pointwise and pairwise classification models of document relevance, respectively, have been integrated into a multistage ranking architecture to achieve improved effectiveness with only a modest increase in inference latency [REF7].

In conclusion, interaction-focused systems that leverage encoder-only models have shown promising results in ad-hoc document ranking tasks. By incorporating contextualized language models and utilizing large-scale datasets, these systems have achieved state-of-the-art performance. The efficient implementation of these models, along with the use of multistage ranking architectures, further enhances their effectiveness in capturing the interaction between queries and documents.

A.2.4 Ranking with Encoder-decoder Models

IEncoder-decoder models have gained significant attention in the field of neural information retrieval due to their ability to capture the interaction between queries and documents. These models, inspired by the success of transformer-based architectures like BERT and GPT, have shown promising results in various natural language processing tasks. In this section, we delve into the use of encoder-decoder models for ranking in interaction-focused systems.

One key aspect of encoder-decoder models is their ability to generate relevant and coherent responses by modeling the conditional probability distribution of the target sequence given the input sequence. This makes them well-suited for ranking tasks in information retrieval, where the goal is to retrieve the most relevant documents given a query.

To train encoder-decoder models for ranking, various objectives have been proposed. One commonly used objective is the denoising objective, also known as masked language modeling [REF1]. In this objective, the model is trained to predict missing or corrupted tokens in the input sequence. This approach has been shown to improve performance and has become a standard practice in pre-training encoder-decoder models [REF1].

Another important consideration in training encoder-decoder models is the choice of pre-training data. Large and diverse datasets, such as the Colossal Clean Crawled Corpus (C4), have been used to boost performance in downstream tasks [REF2]. The use of such datasets allows the model to learn generalizable knowledge that can be applied to a wide range of ranking tasks.

In the context of ranking with encoder-decoder models, the architecture plays a crucial role. The standard encoder-decoder architecture uses fully-visible masking in the encoder and the encoder-decoder attention, with causal masking in the decoder [REF3]. However, alternative architectures, such as decoder-only prefix language models, have also been explored [REF4]. These models leverage fully-visible self-attention over the input, which can improve the ranking performance.

Furthermore, the scalability of encoder-decoder models has been a topic of interest. Scaling up the model size, training steps, and ensembling techniques have been investigated to improve performance [REF8]. It has been observed that increasing the model size and training steps can lead to better ranking results, highlighting the importance of computational resources in training encoder-decoder models.

In addition to the technical aspects, the evaluation of encoder-decoder models for ranking in interaction-focused systems is crucial. Metrics such as precision@k and mean average precision (MAP) are commonly used to assess the effectiveness of these models in retrieving relevant documents.

Overall, encoder-decoder models have shown great potential in ranking tasks within interaction-focused systems. Their ability to capture the interaction between queries and documents, coupled with effective training objectives and large-scale pre-training data, make them a promising approach for improving the retrieval performance in neural information retrieval systems.

[REF1] Dai, Z., Le, Q. (2015). Semi-supervised sequence learning. In Advances in Neural Information Processing Systems.

[REF2] Radford, A., et al. (2018). Improving language understanding by generative pre-training. URL: https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf

[REF3] Taylor, W. (1953). Cloze procedure: A new tool for measuring readability. Journalism Quarterly.

[REF4] Devlin, J., et al. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics.

[REF8] Lan, Z., et al. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. In International Conference on Learning Representations.

A.2.5 Fine-tuning Interaction-focused Systems

Fine-tuning interaction-focused systems is an important aspect of neural information retrieval. In this context, fine-tuning refers to the process of optimizing the performance of a system by adjusting its parameters based on user interactions and feedback. This section explores the significance of fine-tuning in interaction-focused systems and discusses relevant approaches and techniques.

One approach to fine-tuning interaction-focused systems is to leverage side-information. Side-information refers to additional data that can provide insights into the underlying structure and relationships within the dataset [REF0]. By incorporating more side-information, the quality of the clusterings found by the system can be improved [REF0]. For instance, in clustering problems, algorithms that learn a distance metric based on side-information have shown promising results [REF2]. These algorithms optimize the metric learning process by formulating it as a convex optimization problem, enabling the derivation of efficient and local-optima-free algorithms [REF0].

Another aspect of fine-tuning interaction-focused systems is the generalization of learned metrics to previously unseen data. Traditional methods such as Multidimensional Scaling (MDS) and Locally Linear Embedding (LLE) focus on finding embeddings for the points in the training set, limiting their ability to generalize to new data [REF1]. In contrast, fine-tuning approaches that learn a full metric over the input space can generalize more easily to unseen data [REF1]. This capability is crucial in scenarios where the system needs to adapt to new information and user preferences.

Supervised learning settings, such as nearest neighbor classification, have also explored the definition and learning of metrics for classification tasks [REF2]. While these methods often yield good metrics for classification, their effectiveness in learning general metrics for other algorithms, such as K-means, remains uncertain, especially when dealing with less structured information [REF2]. However, recent approaches have shown promise in leveraging similarity information to learn distance metrics for clustering tasks [REF2].

Fine-tuning interaction-focused systems can also involve the use of clustering with side information [REF4]. This approach involves learning a distance metric using similarity information and then clustering the data based on that metric [REF4]. By incorporating side information, such as indicators of whether two points belong to the same or different clusters, the clustering performance can be significantly improved [REF4]. This approach has been demonstrated to outperform traditional clustering algorithms like K-means and constrained K-means [REF4].

To optimize the fine-tuning process, various algorithms and techniques have been proposed. These include convex optimization methods [REF9], iterative projection methods [REF8], and the use of the Newton-Raphson method [REF7]. These algorithms enable efficient and local-minima-free optimization of the system's parameters, ensuring that the fine-tuning process is effective and reliable [REF9].

In summary, fine-tuning interaction-focused systems plays a crucial role in optimizing their performance. By leveraging side-information, generalizing learned metrics, and incorporating clustering with side information, these systems can adapt to user preferences and improve their retrieval capabilities. The use of efficient optimization algorithms further enhances the fine-tuning process, ensuring its effectiveness and reliability.

A.2.6 Dealing with long texts

Dealing with long texts poses a significant challenge in neural information retrieval systems. Long documents often contain multiple passages that may be relevant to a user's query, requiring effective methods to handle and retrieve the most relevant information. In this section, we discuss various approaches and techniques employed by interaction-focused systems to address this challenge.

One approach to handling long texts is the use of passage-based retrieval models. These models aim to identify and rank relevant passages within a document, rather than treating the entire document as a single unit [REF4]. For instance, PARADE (Passage Ranking with Document-level Context) is an end-to-end document reranking model that incorporates diverse relevance signals from the full text into ad-hoc ranking [REF0]. By aggregating relevance signals across passages, PARADE improves the effectiveness of retrieval, particularly when the number of relevant passages per document is low [REF0]. Similarly, Birch-Passage, an improved variant of the Birch model, utilizes passages instead of sentences as input and is trained end-to-end on the target corpus [REF2]. These passage-based models demonstrate the importance of considering passage-level relevance in long document retrieval.

Another approach is the use of passage aggregation techniques to determine the overall relevance score of a document. ELECTRA-MaxP, for example, adopts the maximum score of passages within a document as the overall relevance score [REF2]. This approach allows for a more comprehensive representation of the document's relevance, taking into account the varying degrees of relevance across different passages. ELECTRA-KNRM, on the other hand, is

a kernel-pooling neural ranking model that leverages the query-document similarity matrix to capture the relevance between passages and the query [REF3]. These techniques highlight the importance of effectively aggregating evidence from multiple passages to improve retrieval performance.

In addition to passage-based models and aggregation techniques, the choice of dataset also plays a crucial role in addressing the challenge of long texts. Different datasets exhibit variations in document length and the number of relevant passages per document. For instance, the TREC DL and MS MARCO datasets share similarities in query overlap, suggesting that queries in both collections can be sufficiently answered by a single highly relevant passage [REF6]. On the other hand, the Genomics dataset contains "natural" passages that can be longer, requiring consideration when drawing conclusions [REF1]. Understanding the characteristics of the dataset is essential for developing effective retrieval models tailored to specific contexts.

Furthermore, the evaluation of interaction-focused systems on long texts is facilitated by the availability of benchmark collections. The TREC-COVID challenge, for example, was developed to address the urgent demand for reliable retrieval of COVID-19 academic literature [REF7]. This challenge utilizes the CORD-19 dataset, which is a dynamic collection enlarged over time [REF7]. The availability of such benchmark collections enables researchers to evaluate and compare the performance of their systems in retrieving information from long texts.

In summary, interaction-focused systems employ various techniques to handle long texts in neural information retrieval. Passage-based models, passage aggregation techniques, dataset characteristics, and benchmark collections all contribute to addressing the challenges associated with long documents. These approaches aim to improve the effectiveness of retrieval by considering passage-level relevance and effectively aggregating evidence from multiple passages.

A.3 Representation-focused Systems

A.3.1 Single Representations

Representation-focused systems in neural information retrieval aim to improve the effectiveness of retrieval models by focusing on the representation of documents and queries. In this section, we will discuss the use of single representations in these systems, highlighting various approaches and their impact on retrieval performance.

One approach to representation-focused systems is the use of dense retrieval models. Khattab and Zaharia [REF0] introduced a late-interaction operator on top of BERT encoders, demonstrating the feasibility of full dense retrieval in information retrieval tasks. Das et al. [REF0] proposed an iterative passage retrieval method for open-domain question answering, where relevant passages are retrieved using reformulated question vectors. Seo et al. [REF0] took a different approach by encoding candidate answer phrases as vectors and directly retrieving answers to input questions. These approaches show promising results

in improving retrieval accuracy in various tasks.

Model distillation is another prominent approach in representation-focused systems. It involves training a smaller model based on the predictions of a larger model [REF1]. This approach can enhance the performance of a smaller model when trained on specific task data. Quantization techniques have also been explored to reduce model sizes, making neural networks suitable for embedded systems [REF1]. These approaches provide alternative methods to improve retrieval performance by leveraging distilled or quantized models.

Hard negative sampling is a technique that has been widely used to optimize retrieval performance. STAR [REF3] effectively minimizes top-K pairwise errors by employing hard negative sampling. BM25 Neg and ANCE [REF3] use static hard negative sampling, achieving compromised top-ranking performance. ADORE [REF3] improves top-ranking performance by training the document encoder using STAR. These results demonstrate the effectiveness of hard negative sampling in optimizing retrieval performance.

End-to-end training has been shown to improve ranking performance in representation-focused systems. ADORE [REF2] utilizes end-to-end training to optimize ranking performance for different compressed indexes. The results indicate that different compression techniques can benefit from end-to-end training, making ADORE suitable for improving the performance of compressed indexes. Additionally, ADORE improves retrieval performance by mapping queries closer to relevant documents, enhancing the overall retrieval effectiveness [REF2].

Efficiency is another important aspect of representation-focused systems. ANCE serves as an efficiency baseline due to its competitive effectiveness [REF6]. The training efficiency of proposed methods, such as STAR and ADORE, is evaluated in terms of training time and computational resources [REF6]. These evaluations provide insights into the efficiency gains achieved by the proposed methods.

In summary, representation-focused systems in neural information retrieval employ various techniques to enhance retrieval performance. Dense retrieval models, model distillation, hard negative sampling, end-to-end training, and efficiency considerations are among the key approaches used in these systems. These techniques contribute to improving the effectiveness and efficiency of retrieval models in neural information retrieval tasks.

A.3.2 Multiple Representations

Representation-focused systems in neural information retrieval often explore the use of multiple representations to enhance retrieval performance. These systems leverage different types of representations, such as down-projected representations and hybrid models, to strike a balance between fidelity and generalization [REF0].

One approach is to use down-projected representations, where the output representations of models like BERT are transformed using a feed-forward layer with a reduced dimensionality [REF0]. These down-projected representations,

referred to as ME-BERT-k, allow for more efficient computation while still capturing important information from the original representations.

Another approach is the use of hybrid models, which combine the strengths of both sparse and dense representations [REF0]. These models linearly combine the scores of a sparse system and a dense system using a trainable weight, resulting in a hybrid model. For example, a hybrid model combining ME-BERT and BM25-uni is referred to as HYBRID-ME-BERT-uni [REF0]. By leveraging both sparse and dense representations, hybrid models aim to achieve a balance between fidelity and generalization.

The use of multiple representations in representation-focused systems is motivated by the limitations of traditional approaches. For instance, traditional bag-of-words (BOW) models rely on exact lexical match between query and document terms, which can lead to vocabulary mismatch issues [REF3]. Neural IR models, on the other hand, employ soft matching techniques to model complex matching and address vocabulary and semantic mismatch problems [REF3]. The introduction of contextualized representations from deep language models further enhances the ability to capture semantic similarity [REF3].

The effectiveness of multiple representations has been demonstrated in various studies. For example, ColBERT proposes a late interaction paradigm that combines contextualized embeddings of queries and documents, achieving competitive effectiveness while significantly reducing computational cost compared to models like BERT [REF4, REF6]. Dual encoders based on BERT have also shown promising results in first-stage retrieval, leveraging learned dense low-dimensional encodings to improve recall and exploit machine learning for generalization [REF1, REF5].

In summary, representation-focused systems in neural information retrieval often leverage multiple representations to enhance retrieval performance. These systems explore down-projected representations and hybrid models to strike a balance between fidelity and generalization. The use of multiple representations addresses limitations of traditional approaches and allows for better modeling of complex matching and semantic similarity.

A.3.3 Fine-tuning Representation-focused Systems

Representation-focused systems in neural information retrieval aim to improve the retrieval performance by fine-tuning the representation models. Fine-tuning involves training the models on specific tasks or datasets to enhance their ability to capture relevant information and generate accurate representations. In this section, we discuss the fine-tuning of representation-focused systems and its impact on retrieval effectiveness.

One common approach to fine-tuning representation-focused systems is to leverage pre-trained models and adapt them to the retrieval task at hand. For instance, Chen et al. [REF0] applied pre-processing code from DrQA to extract clean text portions from Wikipedia articles and split them into passages for retrieval. They then fine-tuned the models using question-answering datasets, such as SQuAD and Natural Questions, to improve the retrieval accuracy [REF0]

[REF1]. The authors found that even with a small number of training examples, a dense passage retriever trained using a general pre-trained language model outperformed traditional retrieval methods like BM25 [REF2]. This suggests that fine-tuning representation models can significantly enhance retrieval performance.

In fine-tuning representation-focused systems, the choice of negative examples plays a crucial role in learning high-quality encoders. Different types of negatives, such as random passages, BM25 top passages, or gold passages from other questions, can be considered [REF4]. The selection of negatives can impact the performance of the system, and it has been observed that reusing gold passages from the same batch as negatives can improve computation efficiency while achieving excellent performance [REF4]. Additionally, in-batch negative training, which reuses negative examples within the same batch, has been shown to be an effective and memory-efficient approach to increase the number of training examples and improve model performance [REF8].

Dense vector representations have proven to be valuable in retrieval tasks, as they can capture semantic relevance even without exact token matching [REF9]. By training discriminatively on labeled pairs of queries and documents, dense encoders can provide high similarity scores to semantically relevant text pairs [REF9]. Furthermore, the use of pre-trained models and cross-attention mechanisms has shown effectiveness in passage or dialogue re-ranking tasks [REF9]. These approaches complement sparse vector methods and offer the flexibility to generate task-specific representations.

In summary, fine-tuning representation-focused systems in neural information retrieval involves adapting pre-trained models to the retrieval task at hand. This process can significantly enhance retrieval performance by leveraging the power of dense vector representations and optimizing the choice of negative examples. By fine-tuning representation models, researchers have achieved improved accuracy and surpassed traditional retrieval methods like BM25.

A.4 Retrieval Architectures and Vector Search

A.4.1 Retrieval architectures

Since no reference provided for this section, no text was generated.

A.4.2 MIP and NN Search Problems

Retrieval Architectures and Vector Search - MIP and NN Search Problems

In the field of neural information retrieval, retrieval architectures and vector search play a crucial role in efficiently retrieving relevant information. One popular approach is the use of Multi-Index Hashing (MIP) and Nearest Neighbor (NN) search problems. These techniques leverage the power of hashing functions to map high-dimensional data into compact binary codes, enabling fast and accurate similarity search.

MIP and NN search problems involve the use of hash functions to map data points into hash codes, which can then be used to efficiently retrieve similar

items. In MIP, the goal is to find the most similar items to a given query point, while in NN search, the objective is to find the nearest neighbors of a query point within a dataset.

The L2-ALSH (Asymmetric Locality Sensitive Hashing) is a widely used hash function for MIP and NN search problems. It combines the standard L2 hash function with a spherical multi-Gaussian random vector and a uniformly distributed random variable [REF0]. The collision probabilities of L2-ALSH are related to the desired notion of similarity between objects [REF3]. This approach has been shown to be effective in approximate nearest neighbor search and is widely used in various settings [REF3].

The L2-ALSH hash function is defined as hL2a,b(x) = aTx + br [REF0]. It maps data points to hash codes based on their inner product similarity. The hash codes generated by L2-ALSH can be used to efficiently retrieve similar items by comparing the hamming distances between hash codes [REF3]. This approach has been proven to be effective in approximate nearest neighbor search and has been widely adopted in the field [REF3].

Another important aspect of retrieval architectures and vector search is the consideration of asymmetry in hash functions. Asymmetric hashes, such as L2-ALSH, allow for different mappings of query points and database vectors, enabling efficient retrieval of similar items [REF4]. The power of asymmetry in LSH and binary hashing has been explored in recent studies, showing that it can enable LSH when symmetric LSH is not possible and yield shorter and more accurate hashes [REF5].

It is worth noting that the choice between tree-based methods and LSH-based methods depends on the specific requirements and characteristics of the dataset. While tree-based methods, such as cone trees, have been proposed for inner product search, LSH-based methods are often preferred due to their independence from the dimensionality of the data [REF5]. However, the exact regimes where LSH-based methods outperform tree-based methods are still under investigation [REF5].

In conclusion, retrieval architectures and vector search techniques, such as MIP and NN search problems, along with the use of L2-ALSH and asymmetry in hash functions, play a crucial role in efficient and accurate information retrieval. These approaches enable fast similarity search and have been widely adopted in various domains. Further research is needed to explore the optimal use of these techniques in different settings and to compare them with other retrieval methods.

A.4.3 Locality sensitive hashing approaches

Locality sensitive hashing (LSH) is a popular technique used in information retrieval systems to efficiently search for similar objects in high-dimensional spaces. LSH achieves this by mapping similar objects to the same or nearby hash buckets, allowing for fast retrieval of nearest neighbors. In recent years, there has been a growing interest in incorporating neural networks into the LSH framework to enhance retrieval performance. This section explores retrieval

architectures and vector search techniques, with a focus on locality sensitive hashing approaches.

One approach to improve the efficiency of LSH is through the use of perturbation vectors. Perturbation vectors are generated by adding small random perturbations to the original query vector, resulting in a set of vectors that are close to the query. The likelihood of finding points close to the query is related to the score of the perturbation vectors [REF0]. Perturbation vectors with lower scores have a higher probability of yielding points near the query. However, generating perturbed queries in a data-independent way can be challenging, and many hashed buckets by the perturbed queries may be redundant [REF1].

To address these challenges, the multi-probe LSH method has been proposed. This method computes a non-overlapped bucket sequence based on the probability of containing similar objects, instead of generating perturbed queries [REF1]. By carefully deriving probing sequences, the multi-probe LSH method probes multiple hash buckets in a systematic manner, achieving desired search accuracy and query time with reduced space requirements [REF1]. Experimental results have shown that the multi-probe LSH method is more space-efficient than the basic LSH and entropy-based LSH methods [REF1].

The query-directed probing sequence is another technique used in LSH to reduce the number of hash tables and probes required to achieve the desired recall precisions [REF2]. Compared to the step-wise probing sequence, the query-directed probing sequence requires significantly fewer hash tables and probes [REF2]. This approach has been shown to be effective in achieving high recall with fewer hash tables, making it a promising technique for efficient vector search [REF5].

The step-wise probing method is motivated by the property of locality sensitive hashing, where buckets that are one step away from the query bucket are more likely to contain objects close to the query than buckets that are two steps away [REF3]. This method first probes all the 1-step buckets, then the 2-step buckets, and so on [REF3]. The total number of n-step buckets in an LSH index with L hash tables and M hash functions per table is $L \times M\hat{n} \times 2\hat{n}$ [REF3]. By probing buckets in a step-wise manner, the step-wise probing method aims to efficiently explore buckets that are likely to contain nearest neighbors [REF3].

The multi-probe LSH method has been shown to achieve similar time efficiencies as the basic LSH method while reducing the number of hash tables required [REF6]. It also reduces the space requirement compared to the entropy-based LSH method, while achieving the same search quality [REF6]. The multi-probe LSH method focuses on improving the space and time efficiency of LSH for high-dimensional similarity search [REF6]. It has been compared to other LSH methods, but a detailed comparison with other indexing techniques is beyond the scope of this work [REF6].

In conclusion, retrieval architectures and vector search techniques play a crucial role in enhancing the efficiency and effectiveness of information retrieval systems. Locality sensitive hashing approaches, such as the multi-probe LSH method and query-directed probing sequence, offer promising solutions for efficient vector search. These techniques reduce the number of hash tables and

probes required, while achieving desired search accuracy and query time. Further research and experimentation are needed to explore the full potential of these approaches in the context of neural information retrieval.

[REF0] -Cx i (δ i) 2 = e -C P i $\times$ i ((δ i) 2) [REF1] Multi-probe LSH method [REF1] [REF2] Query-directed probing sequence [REF2] [REF3] Stepwise probing method [REF3] [REF5] Approximation for success probabilities [REF5] [REF6] Comparison of multi-probe LSH with other methods [REF6]

A.4.4 Vector quantisation approaches

Vector quantisation is a widely used technique in retrieval architectures for efficient information retrieval. It involves the process of encoding vectors into a set of codewords or centroids, which are then used for indexing and searching purposes. In this section, we will explore the concept of vector quantisation and its applications in retrieval architectures, specifically focusing on the use of vector quantisation for efficient search and retrieval [REF8].

To obtain precise distances in vector quantisation, it is crucial to limit the quantisation error. This can be achieved by having a sufficiently large number of centroids or codewords [REF0]. However, this poses several challenges. Firstly, a large number of samples are required to learn the quantiser, which can be computationally expensive [REF0]. Secondly, the complexity of the algorithm itself can be prohibitive [REF0]. Lastly, the memory required to store the floating-point values representing the centroids can be a limiting factor [REF0].

One approach to address these challenges is the hierarchical k-means (HKM) algorithm, which improves the efficiency of the learning stage and the corresponding assignment procedure [REF0]. However, the aforementioned limitations still apply, particularly in terms of memory usage and the size of the learning set [REF0].

Another important aspect of vector quantisation is the use of pre-processing techniques to enhance the efficiency of quantisation. These techniques aim to remove redundancy in the signal, reduce signal variance, or concentrate signal energy, resulting in better performance for a given bit rate and complexity [REF1]. For example, predictive quantisation involves removing linear predictions based on past reconstructed values from the signal, and quantising the resulting prediction residual [REF1]. In addition, transforming vectors or blocks of input symbols using linear and orthogonal transforms can also improve quantisation efficiency [REF1].

Efficient computation of distances is crucial in retrieval architectures. The use of the Hamming space allows for efficient distance computation, particularly through table lookups [REF2]. In large datasets, an exhaustive comparison of the query vector with all codes is impractical. To address this, a modified inverted file structure can be used to rapidly access the most relevant vectors [REF2]. This structure utilizes a coarse quantiser and short codes to represent vectors in the associated list, computed using a product quantiser [REF2].

Vector quantisation offers desirable structural properties, such as nested tree-structured codes that are optimized for progressive transmission applications

[REF3]. These codes allow for progressive improvement of a signal as more bits arrive. Additionally, clustering algorithms used to design vector quantisers can enhance specific features of the original signal, such as identifying small tumors in medical images [REF3]. While scalar techniques remain dominant in many applications, vector extensions are finding increasing use in signal compression and other signal processing applications [REF3].

In the context of information retrieval, vector quantisation has been applied to various domains. For instance, in image retrieval, global descriptors such as GIST descriptors can be mapped to short binary codes using vector quantisation techniques [REF4]. This mapping allows for efficient search of Euclidean nearest neighbors by approximating the search using Hamming distances between codes [REF4]. Similarly, binary signatures can be used to refine quantised descriptors in bag-of-features image search frameworks [REF4].

In conclusion, vector quantisation plays a crucial role in retrieval architectures, enabling efficient search and retrieval of information. Despite the challenges associated with quantisation error, computational complexity, and memory usage, techniques such as hierarchical k-means and pre-processing methods have been developed to address these limitations. The use of the Hamming space and inverted file structures further enhance the efficiency of distance computation and access to relevant vectors. Vector quantisation offers desirable structural properties and has found applications in various domains, including image retrieval.

A.4.5 Graph approaches

Graph-based approaches have gained significant attention in the field of neural information retrieval due to their ability to capture complex relationships and dependencies among data points. These approaches leverage the concept of navigable small-world networks (NSW) to construct efficient retrieval architectures [REF8]. The NSW algorithm explores the neighborhood of the closest elements in a greedy manner, continuously improving the known k closest elements [REF0]. By iteratively shrinking the search radius, the algorithm can achieve remarkable performance on various datasets [REF2].

Kleinberg's navigability criterion, which utilizes regular lattice graphs augmented with long-range links, has inspired the development of many K-Nearest Neighbor Search (K-NNS) and Approximate K-Nearest Neighbor Search (K-ANNS) algorithms [REF1]. However, these approaches suffer from polylogarithmic complexity scalability and require prior knowledge of the data distribution, limiting their applicability [REF1] [REF4]. In contrast, the NSW algorithm offers a decentralized graph construction mechanism that is suitable for data in arbitrary spaces [REF4]. It has been suggested that the NSW model may even explain the navigability of large-scale biological neural networks [REF4].

The NSW algorithm demonstrates several desirable properties for neural information retrieval. It converges to high recall rates with each point comparing only to a small fraction of the dataset on average [REF5]. Moreover, the algorithm is easy to implement and requires minimal extra data structures [REF5].

The performance advantage of the NSW algorithm can be further enhanced by optimizing the overlap between neighbors on different layers of the graph [REF6]. Simulations have shown that the proposed selection of parameters leads to significant speedup and improved performance [REF6].

Empirical evaluations of the NSW algorithm have demonstrated its efficiency and scalability. The algorithm exhibits polynomial time complexity, regardless of the dataset size [REF3]. The scan rate curves of different datasets form parallel straight lines, indicating consistent performance across various datasets [REF3]. Comparative studies have shown that the NSW algorithm outperforms existing methods such as Recursive Lanczos Partitioning and Locality Sensitive Hashing in terms of both speed and accuracy [REF5].

In conclusion, graph-based approaches, particularly the NSW algorithm, offer promising solutions for neural information retrieval. These approaches leverage the concept of navigable small-world networks to construct efficient retrieval architectures. By exploring the neighborhood of closest elements in a greedy manner, the NSW algorithm achieves high recall rates with minimal computational overhead. Furthermore, the algorithm is easy to implement and exhibits consistent performance across different datasets.

A.4.6 Optimisations

In the field of neural information retrieval, retrieval architectures and vector search play a crucial role in optimizing the efficiency and effectiveness of information retrieval systems. Various techniques and approaches have been proposed to enhance the retrieval process and improve the accuracy of search results. This section discusses some of the key architectures and optimizations used in retrieval systems.

One approach that has been explored is the use of decomposition techniques to optimize the computation of inverted lists [REF0]. By decomposing the computation of tables used during the scan of the inverted list, the computational cost can be reduced. This decomposition technique enables more efficient retrieval by reducing the number of multiply-add operations required. However, it should be noted that this decomposition may not be suitable for all scenarios, especially when memory usage is a concern.

Another important aspect of retrieval architectures is the use of joint learning structures. Two types of joint learning structures have been identified: tree-based approaches and product quantization (PQ) based approaches [REF1]. Tree-based approaches require special approximate training techniques, which can complicate their adoption in real-world industrial retrieval systems. On the other hand, PQ-based approaches have been designed for smaller computer vision tasks and may not be applicable to large-scale information retrieval tasks. To address this limitation, advancements have been made in product quantization based embedding indexes jointly trained with deep retrieval models.

In the context of dense retrieval (DR), the storage and retrieval of token-level representations have been a challenge due to the large index size [REF2]. Previous DR models have relied on brute-force search for candidate retrieval, which

can be computationally expensive. However, recent studies have proposed more efficient architectures for DR models, enabling faster retrieval of candidates.

Dense retrieval methods often retrieve a significant number of new documents compared to sparse retrievals [REF3]. This indicates that DR methods have low overlap with traditional retrieval methods, such as BM25. The high hole rate and low recall metric suggest that DR methods may have advantages in certain retrieval tasks, such as the TREC 2020 Deep Learning Track.

Optimizations in vector search also play a crucial role in improving retrieval efficiency. For instance, in end-to-end ranking, the choice of similarity function can impact the performance of the retrieval system [REF4]. Different distance metrics, such as cosine similarity and L2 distance, have been used in vector search. Additionally, the configuration of the index, including the number of partitions and the number of nearest neighbors to retrieve, can also affect the retrieval performance.

Training strategies and techniques are also important considerations in retrieval architectures. For example, the training of the DPR model involves in-batch negative sampling and the combination of training data from multiple datasets [REF5]. The use of gradient straight-through estimators and appropriate techniques for non-differentiable operations, such as arg min, are crucial for training models with quantization steps.

Implementation details and hyperparameter settings are also critical in retrieval architectures. For instance, the training of the BERT-Siamese model involves the use of RoBERTa base and specific optimization techniques, such as the LAMB optimizer [REF6]. These implementation details can significantly impact the convergence and performance of retrieval models.

In conclusion, retrieval architectures and vector search optimizations are essential components of neural information retrieval systems. Techniques such as decomposition, joint learning structures, and efficient indexing strategies contribute to the improvement of retrieval efficiency and accuracy. Additionally, training strategies, implementation details, and hyperparameter settings play a crucial role in the performance of retrieval models.

A.5 Learned Sparse Retrieval

A.5.1 Document expansion learning

Document expansion learning is a crucial aspect of learned sparse retrieval, aiming to improve the effectiveness of queries by expanding them with additional terms. Traditional approaches to query expansion have primarily focused on unstructured bag-of-words queries, neglecting the potential benefits of structured queries [REF2]. However, research has shown that structured queries, such as Boolean conjunctive normal form (CNF) queries, can be more effective in certain domains and provide better control over expansion [REF2] [REF3].

One key challenge in document expansion learning is predicting the term mismatch probability, which refers to the likelihood of a query term not appearing in relevant documents [REF4]. Prior research has demonstrated that

accurately estimating this probability can significantly enhance retrieval accuracy [REF0]. To address this challenge, several methods have been proposed to predict the term recall probability (P(t|R)), which is closely related to term mismatch [REF1]. These methods leverage various features and techniques to estimate P(t|R) and incorporate it into retrieval models [REF1].

The use of predicted term mismatch probabilities as term weights has been explored to improve traditional retrieval models, such as Okapi BM25 and language models [REF1]. By incorporating these predicted probabilities, retrieval models can better capture the relevance between query terms and documents, leading to improved retrieval performance [REF1]. Experimental results have shown that using true or predicted term recall values can outperform standard baselines in ad hoc retrieval tasks [REF1].

Furthermore, document expansion learning can be guided by the diagnosis of term mismatch probabilities [REF4]. This diagnosis can be used to suggest manual query reformulation, guide interactive query expansion, or motivate other responses [REF4]. For instance, the diagnosis can guide the creation of Boolean CNF structured queries that selectively expand problematic query terms while keeping the rest of the query unchanged [REF4]. Experiments conducted on TREC Ad-hoc and Legal Track datasets have demonstrated the effectiveness of this diagnostic approach, reducing user effort and producing simple yet effective structured queries that outperform their bag-of-words counterparts [REF4] [REF9].

In summary, document expansion learning plays a vital role in learned sparse retrieval by leveraging predicted term mismatch probabilities and structured queries. By effectively expanding queries with additional terms, retrieval models can better capture the relevance between queries and documents, leading to improved retrieval performance. The use of structured queries, such as Boolean CNF queries, and the diagnosis of term mismatch probabilities further enhance the effectiveness of document expansion learning.

A.5.2 Impact score learning

Impact score learning is a crucial aspect of learned sparse retrieval, which aims to assign weights to terms in order to capture their importance in the retrieval process. Several approaches have been proposed in the literature to learn impact scores and improve the effectiveness and efficiency of retrieval models.

One approach is the use of the MaxScore query processing algorithm, which allows for "sum of impact" scoring [REF0]. This algorithm, combined with the common index file format and the BM25 scoring method, has been shown to achieve effective retrieval results [REF0]. Another technique, called TILDEv2, integrates the best-of-breed from recent advances in neural retrieval and employs a novel use of the original TILDE as an efficient passage expansion technique [REF1] [REF2]. TILDEv2 significantly improves the effectiveness of the original TILDE while reducing its index size [REF2].

uniCOIL is another model that utilizes learned impact weights for sparse retrieval [REF3]. It achieves good effectiveness by adding expansion to make

up for the lost expressivity of weight vectors [REF3]. Furthermore, uniCOIL represents the state of the art in sparse retrieval using learned impact weights, outperforming DeepImpact [REF3]. These findings highlight the potential of impact score learning in improving retrieval effectiveness.

In the context of evaluation, the effectiveness of learned sparse retrieval models is typically measured using metrics such as MRR@10, nDCG@10, and MAP [REF4]. Statistical significance tests, such as paired two-tailed t-tests with Bonferroni correction, are commonly employed to compare the performance of different methods [REF4]. Additionally, query latency is evaluated within both CPU and GPU environments to assess the efficiency of the models [REF4].

Both dense and sparse learned representations leverage transformers, but sparse approaches project the learned knowledge back into the sparse vocabulary space, enabling the utilization of existing techniques for efficient query evaluation [REF5]. The tradeoffs between output quality, query latency, and index size are important considerations in the design space of modern information retrieval techniques [REF5]. While learned representations for information retrieval show promise, the advantages and disadvantages of dense versus sparse approaches are still being explored [REF5].

In summary, impact score learning plays a crucial role in learned sparse retrieval. Various techniques, such as the MaxScore query processing algorithm, TILDEv2, and uniCOIL, have been proposed to improve retrieval effectiveness and efficiency. Evaluation metrics and statistical tests are used to assess the performance of these models. The tradeoffs between output quality, query latency, and index size are important factors to consider in the design of information retrieval techniques [REF8].

A.5.3 Sparse representation learning

Sparse representation learning is a fundamental aspect of neural information retrieval systems. It involves the development of models and techniques that enable efficient and effective retrieval of relevant information from large collections of documents. In this section, we discuss the concept of learned sparse retrieval and its significance in the field of information retrieval.

One approach to learned sparse retrieval is the use of sparse representation models such as BOW (Bag-of-Words) and SPLADE (Sparse Lexical Adaptive Document Embeddings) [REF0]. These models aim to represent documents and queries in a sparse manner, where only a subset of the features or terms are considered relevant for retrieval. This sparsity enables efficient computation and storage, making it suitable for large-scale retrieval tasks.

EPIC (Expansion via Prediction of Importance with Contextualization) is another approach that combines term importance modeling and expansion using contextualized language models [REF1]. EPIC builds query and document representations by incorporating salience and expansion techniques. This approach has shown promising results in narrowing the effectiveness gap between practical and computationally expensive retrieval methods.

The effectiveness of learned sparse retrieval models is not solely dependent

on their efficiency but also on their interpretability. Sparse models, such as SPLADE, provide interpretable representations where the dimensions directly correspond to the terms in the lexicon [REF1]. This interpretability allows for better understanding and analysis of the retrieval process.

To improve the efficiency of sparse representation learning, regularization techniques are often employed. Regularization helps in reducing the computational cost by promoting sparsity in the learned representations. For example, FLOPS regularization has been shown to be advantageous over L1 regularization in terms of decreasing computing cost [REF4]. Additionally, regularization schedules, such as gradually increasing the regularization weight, have been proposed to prevent the model from getting stuck in local minima during training [REF6].

The trade-off between effectiveness and efficiency is a crucial consideration in learned sparse retrieval. Models like SPLADE have demonstrated efficiency levels equivalent to sparse BOW models while outperforming other dense retrieval models [REF0]. This trade-off is often visualized by varying the efficiency (FLOPS) and observing the corresponding effectiveness (MRR@10) [REF4].

In conclusion, learned sparse retrieval techniques play a vital role in neural information retrieval systems. These techniques enable efficient and effective retrieval of relevant information from large document collections. Sparse representation models, such as BOW and SPLADE, along with regularization and interpretability considerations, contribute to the advancement of learned sparse retrieval in the field of information retrieval.