A Review on Dense Retrieval

João Coelho
86448
joao.vares.coelho@tecnico.ulisboa.pt

Abstract

With the availability of large data collections, first stage retrieval has shifted towards the usage of neural methods, mostly based on the Transformer architecture, often referred to as dense retrievers. This review starts by exploring classical retrieval models and their limitations, followed by an analysis on state-of-the-art dense retrievers, also addressing underlying machine learning architectures and techniques. Benchmarks and evaluation metrics are presented, and possible research directions discussed.

1 Introduction

Information retrieval plays a critical role in a wide range of applications, ranging from search engines to question answering systems. The ability to efficiently and accurately retrieve relevant information from large data collections is essential in enabling users to access and make sense of the vast amount of available information. Traditional sparse retrieval algorithms, such as the well-known BM25 algorithm, were the backbone of retrieval systems for many years. However, these approaches often face challenges in capturing semantic relationships and effectively handling complex queries.

The emergence of large-scale data collections and advancements in machine learning have prompted a paradigm shift in first stage retrieval. Neural methods, particularly those based on the Transformer architecture, have gained significant attention and achieve state-of-the-art results in many recent benchmarks. These models, often also referred to as dense retrievers, leverage deep learning to capture intricate semantic relationships and enhance the retrieval capabilities of traditional sparse retrievers.

This paper starts by presenting a background on classical information retrieval models, using their drawbacks to motivate the need for alternative retrieval techniques. Then, the base machine learning architectures and techniques necessary for the training and usage of dense retrievers are addressed. Moving into the field of dense retrieval, the Transformer architecture is introduced, and a study on state-of-the art dense retrievers is conducted. To conclude the study, methods that combine neural approaches and sparse retrievers are addressed, in an attempt of using methods that combine the best of both techniques. Furthermore, benchmark datasets and evaluation metrics that are commonly used to assess the performance of retrieval models are presented. Finally, potential research directions are highlighted in order to promote future work.

2 Information Retrieval Background

Information Retrieval's (IR) core task is **ad-hoc retrieval**, where an information need is posed by an user to a system, which should return relevant documents that satisfy the user's needs. Before diving into the topic dense retrieval, a description of the three families of early IR models, namely Boolean models, Vector Space models, and Probabilistic models, is presented. In depth information of these methods can be consulted in the following books: [5, 125].

2.1 Boolean Models

Boolean models [93, 94] operate on the principles of boolean logic and set theory. Both documents and queries are represented as a **bag-of-words**, i.e., a set of unordered and (assumed to be) independent

terms. When using one such model, information needs are formulated using boolean operators such as AND, OR, and NOT to establish relationships between query terms. Documents are then classified as relevant or irrelevant by assessing whether or not they satisfy the logical query constraint. One significant characteristic of Boolean Models is that since it focuses solely on determining the presence or absence of query terms within documents, the model does not provide a measure of relevance or prioritize the results based on their significance to the query.

Extended Boolean Models [164, 99] were proposed as a method to surpass the inability to rank documents, being an intermediate between Boolean models and the Vector Space models described in the next subsection.

2.2 Vector Space Models

Vector Space models [163] rely on the assumption that textual similarity can be approximated through spatial proximity, i.e., if there is a vector space where documents are represented, the most relevant documents to a query should be those closer to the query's representation.

In this approach, for a collection of documents $D = \{d_1, d_2, ...\}$, the vocabulary V is defined as the set of unique tokens among all documents. Each document in the collection is represented as a vector, $d_i = (t_1, t_2, ..., t_{|V|})$, where for each d_i, $t_v = 0$ if $t_v \notin d_i$ or $t_v = 1$ if $t_v \in d_i$.

Instead of having binary values, **term weighting** can be considered [162, 161]. This way, documents are represented by $d_i = (t_1, t_2, ..., t_{|V|})$, but $t_v = w_{v,i}$. One way of computing the weights is by using TF-IDF [119, 83]:

$$w_{v,i} = \mathrm{tf}(v, i) \times \mathrm{idf}(v) \ , \tag{1}$$

where $\mathrm{tf}(v, i)$ is the term frequency of t_v on document d_i, and $\mathrm{idf}(v)$ is the inverse document frequency of term t_v, given by:

$$\mathrm{idf}(v) = \log \frac{|D|}{|\{d_{i'} \in D | t_v \in d_{i'}\}|} \ . \tag{2}$$

The motivation for this metric is that terms that appear very often in a document are important (except for stop-words), but terms that appear in the document and are simultaneously rare among the collection of documents are, in principle, context-related, and should also be more important.

Instead of TF-IDF weights, **term distribution models** can be considered, where the objective is to derive a probabilistically motivated weighting scheme for the vectorial representations [124]. For instance, a 2-Poisson distribution has been suggested as a method to model term frequencies [15], assuming that there are two classes of documents associated with a term: the class of relevant documents (with high average occurrences), and the class of irrelevant documents (with low average occurrences):

$$\mathbb{P}(X = \mathrm{tf}) = \lambda \frac{e^{-\mu_1} (\mu_1)^{\mathrm{tf}}}{\mathrm{tf!}} + (1 - \lambda) \frac{e^{-\mu_2} (\mu_2)^{\mathrm{tf}}}{\mathrm{tf!}} \ . \tag{3}$$

In the previous equation, X is a random variable representing the number of occurrences of a term, λ is the probability of a document being in the relevant class, and μ_1 and μ_2 are the average number of occurrences of a word in the relevant and irrelevant classes, respectively. Empirically, the occurrence distribution for most terms has been shown to be monotonically decreasing. This property is not always captured by a 2-Poisson distribution, with proposed extensions relying on the mixture of more Poisson distributions [130, 25].

To compute the similarity between two $|V|$-dimensional representations, r_1 and r_2, the cosine similarity can be used:

$$\mathrm{sim}_{\cos}(r_1, r_2) = \frac{r_1 \cdot r_2}{||r_1||\ ||r_2||} = \frac{\sum_{k=1}^{|V|} r_{1k} r_{2k}}{\sqrt{(\sum_{k=1}^{|V|} (r_{1k})^2)} \sqrt{(\sum_{k=1}^{|V|} (r_{2k})^2)}} \ . \tag{4}$$

As such, instead of classifying a document as relevant or irrelevant, vector space models are able to rank a list of documents according to their relevance or similarity to the query.

It is important to note that the approaches discussed in this section result in **sparse representations**, given the high number of terms in the vocabulary, which can be a computational bottleneck. Also, all terms are considered to be independent, since a bag-of-words representation is still being followed. This means that the cosine similarity between two different word representations is always 0, even if they are related.

2.3 Probabilistic Models

Probabilistic Models for IR rely on the **probabilistic ranking principle** [156], which states that documents should be ranked by probability of relevance towards a query. Multiple techniques for the estimation of this probability, denoted by $\mathbb{P}(R|d)$, have been proposed. Most works have a common basis, the Binary Independence Model (BIM) [207, 157], which considers multiple assumptions (i) probability depends only on query and document, (ii) there is a subset R of relevant documents, (iii) index terms are independent, and (iv) non-query terms are equally likely to appear in relevant and irrelevant documents. Moreover, both queries and documents are represented as binary term incidence vectors, and terms not appearing in the query do not affect ranking. The objective is to rank documents by the (log) odds of relevance, $\log \frac{\mathbb{P}(R|d)}{\mathbb{P}(\overline{R}|d)}$, as it as been shown to minimize wrong judgments [51]:

$$\log \frac{\mathbb{P}(R|d)}{\mathbb{P}(\overline{R}|d)} = \log \frac{\mathbb{P}(d|R)\mathbb{P}(R)}{\mathbb{P}(d|\overline{R})\mathbb{P}(\overline{R})} \approx \log \frac{\mathbb{P}(d|R)}{\mathbb{P}(d|\overline{R})} \ . \tag{5}$$

After applying Bayes rule, the prior $\mathbb{P}(R)$ (i.e., the probability of a randomly sampled document from the collection being relevant) is assumed to be independent of the document and, as such, constant across documents. Probabilistic IR models differ in the estimation of $\mathbb{P}(d|R)$. In the simplest approach, term independence is considered, which allows representing the probability as a product of individual term probabilities (the following derivation is adapted from [169]):

$$\mathbb{P}(d|R) = \prod_{t_i \in q,d} \mathbb{P}(t_i|R) \prod_{t_j \in q,\overline{d}} (1 - \mathbb{P}(t_j|R)) \ . \tag{6}$$

In the previous equation, t_i are the terms that common to the query and the document, and t_j are query terms absent from the document. Denoting $p_i = P(t_i|R)$ and $u_i = P(t_i|\overline{R})$ the (log) odd relevance expression can be rewritten:

$$\log \frac{\mathbb{P}(d|R)}{\mathbb{P}(d|\overline{R})} = \log \frac{\prod_{t_i \in q,d} p_i \prod_{t_j \in q,\overline{d}} (1 - p_j)}{\prod_{t_i \in q,d} u_i \prod_{t_j \in q,\overline{d}} (1 - u_j)} \ . \tag{7}$$

For a given query, the above expression can be rewritten with respect to only the terms present in a document by adding the constant $\log \prod_{t_i \in q} \frac{1-u_i}{1-p_i}$:

$$\log \frac{\prod_{t_i \in q,d} p_i \prod_{t_j \in q,\overline{d}} (1 - p_j)}{\prod_{t_i \in q,d} u_i \prod_{t_j \in q,\overline{d}} (1 - u_j)} = \log \prod_{t_i \in q,d} \frac{p_i(1-u_i)}{u_i(1-p_i)} = \sum_{t_i \in q,d} \log \frac{p_i(1-u_i)}{u_i(1-p_i)} \ . \tag{8}$$

Multiple ranking functions can be produced by using different estimations for p_i and u_i. For instance, probabilistic justification to the previously heuristically motivated inverse document frequency metric can be provided with two assumptions [36], (i) by considering $p_i = \frac{1}{2}$ for all query terms, which yields $\log \frac{p_i}{1-p_1} = 0$, and (ii) assuming that the vast majority of documents in a collection are irrelevant to a query, estimating u_i as $\frac{|\{d_{k'} \in D | t_i \in d_{k'}\}|}{|D|}$. This gives the following ranking function:

$$\log \frac{p_i(1-u_i)}{u_i(1-p_i)} = \log \frac{|D| - |\{d_{k'} \in D | t_i \in d_{k'}\}|}{|\{d_{k'} \in D | t_i \in d_{k'}\}|} \approx \log \frac{|D|}{|\{d_{k'} \in D | t_i \in d_{k'}\}|} \ , \tag{9}$$

which is similar to the idf function presented in Equation 2. Other authors [61] have raised problems with the previous constant estimate of p_i, stating that, empirically, p_i grows with the term frequency of term i in the document, proposing the usage of $p_i = \frac{1}{3} + \frac{2}{3}\frac{\text{tf}(i,d)}{|D|}$.

One important ranking function derived from probabilistic IR principles is the BM25 [155], which is to this day perhaps the most commonly used lexical retrieval baseline. While the BIM postulates binomial distributions for terms, BM25 relies on 2-Poisson mixtures. Given a document d and a query q with n tokens, $(q_1, ..., q_n)$, the relevance score is computed as follows:

$$\text{score}_{\text{BM25}}(q,d) = \sum_{i=1}^{n} \text{idf}(i) \times \frac{\text{tf}(i,d) \times (k_1+1)}{\text{tf}(i,d) + k_1 \times \left(1 - b + b \times \frac{|d|}{\text{avgdl}}\right)} , \tag{10}$$

where b and k_1 are tunable hyperparameters, and avgld is the average length of documents in the collection.

Since textual data may be composed of several fields (e.g., documents from news collections may be divided in headline and body), BM25F [210] was proposed as an extension. The authors argue that given the different length of the fields, computing the BM25 scores for each field and combining them linearly raises problems, like not keeping the non-linear relationship between term weights and term frequencies, difficulties in performing the field length normalization, and unstable idf computation. To address these issues, the authors proposed an approach that weights term frequencies accordingly to their field importance. Given the decomposition of a document in a set of fields, F, the scoring method is given by:

$$\widetilde{\text{tf}}(i,d) = \sum_{z \in F} w^z \times \frac{\text{tf}^z(i,d) \times (k_1+1)}{\text{tf}^z(i,d) + k_1 \times \left(1 - b^z + b^z \frac{|d^z|}{\text{avgdl}^z}\right)} , \tag{11}$$

where w^z is the weight associated with field z. The remaining terms with superscript z maintain their previous definition, now over the field z instead of the full document. The field-level term frequencies are then used to compute the ranking score:

$$\text{score}_{\text{BM25F}}(q,d) = \sum_{i=1}^{n} \text{idf}(i) \times \widetilde{\text{tf}}(i,d) . \tag{12}$$

BM25+ [120] is another extension, addressing the fact that the term-frequency normalization by document length is not lower-bounded properly, which may over-penalize large documents. The alternative scoring method is given by:

$$\text{score}_{\text{BM25+}}(q,d) = \sum_{i=1}^{n} \text{idf}(i) \times \left(\frac{\text{tf}(i,d) \times (k_1+1)}{\text{tf}(i,d) + k_1 \times \left(1 - b + b \times \frac{|d|}{\text{avgdl}}\right)} + \delta \right) . \tag{13}$$

The only difference when compared to the original BM25 is the addition of an hyperparameter δ.

2.4 Semantic Similarity

The ability to capture the underlying semantics or contextual meaning of the concepts is one of the main motivations for the usage of dense retrievers, as the methods discussed so far are all built on top of the concept of **lexical similarity**, i.e., frequency of terms and weighted degrees of overlap between query and document terms. One prominent problem in such a scenario is **vocabulary mismatch** [52, 212], caused by terms between queries and documents being lexically different but semantically similar. One pioneering method to deal with this problem was Latent Semantic Indexing [40], which leverages single-value decomposition to find a low-rank approximation of the matrix that describes the occurrences of terms in documents, resulting in lower-dimensional representations for the documents and mitigating the problem of identifying synonymy.

In a different perspective, **query expansion** methods aim to augment queries with new terms [47, 145, 189, 6]. Sampling methods for selecting new terms vary (ontology-based [10], relevance feedback [160], random walk models [28], among others [21]), but the intuition is that by incorporating synonyms and/or related terms, the expanded query can capture a wider range of documents and increase the chances of retrieving the desired information. Similarly, instead of query expansion, **document expansion** can be considered, which follows the same intuition of the former, but adds new terms to the document instead [91, 113, 177]. This allows the expansion to be done offline (i.e., prior to indexing), which may result in faster run times for the end-user, but early tests with classical retrieval models showed that query expansion has better performance [11].

3 Machine Learning for Information Retrieval

This section covers the background machine learning techniques used in IR tasks, specifically those necessary for understanding the architectures behind dense retrievers. The Multi Layer Perceptron is introduced, and commonly used loss functions behind the task of Learning to Rank are addressed. Then, a brief overview of word embeddings is provided, as an application of feed forward networks to generate dense representations. Finally, fundamental concepts behind state-of-the-art dense retrievers are presented through an overview of the Transformer architecture and the BERT model.

3.1 Multi Layer Perceptron

The Perceptron [158] is an algorithm for supervised learning of binary classifiers which aims to mimic an artifical neuron. Given an n-dimensional input vector $x \in \mathbb{R}^n$, the output of the Perceptron, y, is computed as follows:

$$y = \mathrm{h}(w^\top x + b) \ , \tag{14}$$

where $w \in \mathbb{R}^n$ is the learnable weight vector and b is the bias. Originally, the activation function h is the signal function, which returns 1 if the input is positive or -1 otherwise. To train the Perceptron, w can be initialized randomly and updated following the equation $w = w + \mu(\hat{y} - y)x$, where $\hat{y}$ is the true label for the input vector being processed and μ is the learning rate. The main issue with this model is that it can only classify linearly separable sets of vectors. However, that problem can be solved by extending the Perceptron, connecting multiple layers of neurons with non-linear activation functions, which is the idea behind a Multi Layer Perceptron, often also referred to as a **feed-forward neural network** (FFNN). It comprises one input layer, at least one hidden layer, and one output layer. Each layer l contains one or more neurons, $n^{[l]}$. For a single hidden layer, let $x^{[0]} \in \mathbb{R}^{n^{[0]}}$ be the input vector. The output, y, is given by:

$$y = \mathrm{h}^{[2]}(\mathbf{W}^{[2]}\mathrm{h}^{[1]}(\mathbf{W}^{[1]}x^{[0]} + b^{[1]}) + b^{[2]}) \ , \tag{15}$$

where $\mathrm{h}^{[i]}$,$\mathbf{W}^{[i]} \in \mathbb{R}^{n^{[i-1]} \times n^{[i]}}$, and $b^{[i]} \in \mathbb{R}^{n^{[i]}}$ are the activation function, weight matrix and bias vector of the i^{th} layer, respectively. Error functions (e.g., mean squared error or the cross-entropy) are used to compare the expected value and the network output for a given input, and model training consists in minimizing one such error (also named cost or loss) function. Gradient Descent is an optimization technique that in order to minimize a given function, updates parameters by taking steps in the direction of steepest descent in the gradient of the error function:

$$\theta_{t+1} = \theta_t - \mu \nabla E_{\theta_t} \ . \tag{16}$$

In the previous expression, t denotes the time-step, E is the error function subject to minimization with respect to parameters θ, and μ is the learning rate.

Variations of gradient descent are applied alongside the back-propagation algorithm to train neural networks such as Multi Layer Perceptrons. After a forward pass, this procedure computes the partial derivatives of the cost function with respect to the different parameters, propagating this information back through the network.

Beyond standard gradient descent, one popular first-order gradient-based optimizer is Adam [88]. Together with adaptive learning rates for each parameter, this method stores an exponentially decaying average of past squared gradients (i.e., an estimate of the second raw moment of the gradient), v_t, and an exponentially decaying average of past gradients (i.e., an estimate of the first moment of the gradient), m_t. At each time-step t the gradient $g_t = \nabla E_{\theta_t}$ is computed, updating the averages:

$$m_t = \frac{(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i} g_i}{1-\beta_1^t} \ , \tag{17}$$

$$v_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i} g_i^2}{1-\beta_2^t}} \ , \tag{18}$$

and the parameter update rule is given by:

$$\theta_{t+1} = \theta_t - \frac{\mu m_t}{v_t + \epsilon} \ , \tag{19}$$

where β_1 and β_2 are hyperparameters to account for how much of the previous gradients should be taken in consideration when updating m_t and v_t. Overall, this approach yields small steps for parameters that are updated very frequently, with larger ones for the others. Extensions include the AdamW [118] optimizer, widely used in NLP tasks, which modifies the weight decay term's contribution to the update step, ensuring the weights are not directly affected. More recently, light-weight estimators for the diagonal of the Hessian matrix have been proposed, allowing for the usage of second-order optimizers which have shown promising results in pre-training of language models [108].

3.2 Learning to Rank

Learning to Rank is a group of techniques that use machine learning algorithms to build ranking models, e.g. for IR applications [112]. Given a set of n documents, D, and a query q, the goal is to find out which $d \in D$ are relevant to the query, and sort them by their relevance.

Pointwise Learning to Rank formulates the problem as a regression or classification task. Let $\hat{\mathrm{y}}(q, d_i)$ be the graded relevance of d_i with respect to query q and f the learnt function. The objective is to minimize loss functions such as the Mean Squared Error:

$$L_{\text{mse}} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{f}(q, d_i) - \hat{\mathrm{y}}(q, d_i))^2 \ , \tag{20}$$

Another approach, Pairwise Learning to Rank, takes pairs of documents as input, learning whether or not one is more relevant than the other. For example, RankSVM [81] is a ranking algorithm based on Support Vector Machines, minimizing functions such as the Hinge Loss:

$$L_{\text{hinge}} = \sum_{\hat{\mathrm{y}}(q,d_i)>\hat{\mathrm{y}}(q,d_j)} \max\left(0, 1 - (\mathrm{f}(q, d_i) - \mathrm{f}(q, d_j))\right) \ . \tag{21}$$

One last technique is Listwise Learning to Rank, where the objective is to directly optimize ranking metrics such as the Mean Average Precision or the Normalized Discounted Commutative Gain. Those and other metrics will be addressed further ahead in this document (Section 5.1), but for instance, once the ranking function f is learnt, let there be a list of documents ordered by it for a given query. The Mean Average Precision (MAP) is computed as follows:

$$\text{MAP} = \frac{\sum_{q\in Q}\text{AP}(q)}{|Q|} \ , \tag{22}$$

$$\text{AP(q)} = \frac{\sum_{i=1}^{n}\text{P@}i \times \mathbb{I}_i}{|S_q^*|} \ . \tag{23}$$

In the previous equations, Q is the set of all queries, S_q^* is the set of relevant documents for query q and $\mathbb{I}_i$ evaluates to 1 The Bi-f the document at rank i is relevant to q and to 0 otherwise. P@i is the precision at the i^{th} returned document, i.e., the fraction of the top-i retrieved documents that are relevant. Functions such as the MAP are not differentiable, since they depend on the ranked position of the documents, which is a piecewise linear function. As such, gradient-based optimizers cannot be used directly. Multiple methods have been proposed to solve this, under the setting of listwise ranking. For instance, RankGP [205] aims to optimize the MAP resorting to genetic algorithms (a meta-heuristic for local search) instead of gradient based optimizers, by using MAP as the fitness function. Other strategies include the usage of fully-differentiable soft-ranking functions [12] which allows the usage of rank-based functions within the loss.

3.3 Word Embeddings

Arriving from the Natural Language Processing (NLP) community, word embeddings were an important step on the representation of textual information [170]. Word2Vec [127] is an algorithm for learning word embeddings that represent words as dense vectors in a continuous space, that has shown good ability of capturing semantic and syntactic relationships between words.

There are two main variants of Word2Vec. Both are built using a simple feed-forward neural network with one hidden layer. In the first, CBOW [127], the network is trained to predict a word given its surrounding neighbors (i.e., the word's context) as input. Formally, given the sequence $t_1, t_2, ..., t_N$ and context size c, the objective of the CBOW approach is to maximize:

$$\frac{1}{N}\sum_{n=1}^{N}\log(\mathbb{P}(t_n|t_{n-c},...,t_{n-1},t_{n+1},...,t_{n+c}))\ . \tag{24}$$

The second approach, named Skip-gram [128], uses the target word as input, with the training task being to try to find the context of the word. Formally, the objective of the Skip-gram approach is to maximize:

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{-c<=j<=c,j\neq 0}\log(\mathbb{P}(t_{n+j}|t_n))\ . \tag{25}$$

The probability of predicting the output term t_O given t_I can be calculated using the softmax function:

$$\mathbb{P}(t_O|t_I)=\frac{\exp\left(r_{t_O}^{\top}r_{t_I}\right)}{\sum_{i=1}^{|V|}\exp\left(r_{t_i}^{\top}r_{t_I}\right)}\ , \tag{26}$$

where r_{t_x} is the representation of t_x and $|V|$ is the size of the vocabulary. Given that $|V|$ can be very large, other methods have been used to compute this probability, such as hierarchical softmax or negative sampling [127, 128].

Multiple works have used word embeddings in the context of IR tasks [159, 43, 53, 60, 129], by considering the methods described above to generate representations for queries and documents, or different approaches such as GloVe [142] or density-based representations [187]. However, the major limitation of word embeddings was still unattended: words with multiple meanings will have only one representation.

To address this, the usage of **contextual embeddings** was adopted, which are context-dependent representations that capture the use of words in multiple scenarios [111]. Unlike traditional word embedding techniques, contextual embeddings learn a vector that is a function of an input sequence that contains the target token. As a result, the same word may have different representations if used in different sentences. To further explain how these can be generated, the following subsections will introduce the Transformer architecture and the BERT model.

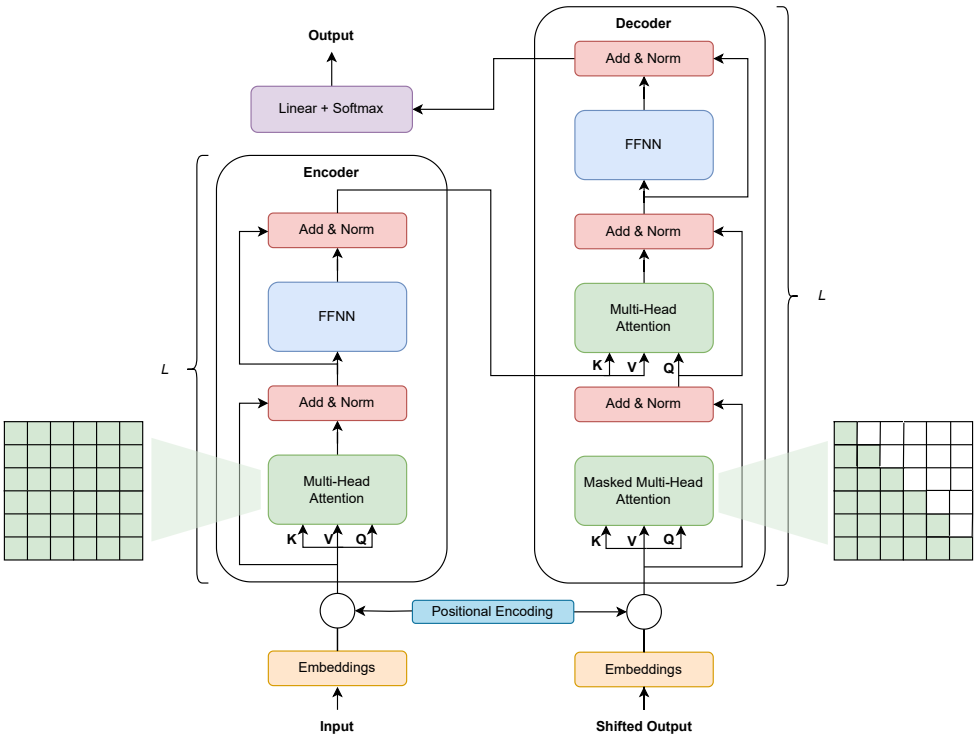

Figure 1: Overview of the Transformer architecture.

3.4 The Transformer Architecture

The Transformer neural network architecture [186] is designed for sequence-to-sequence tasks and has proven to be highly effective in NLP [84]. Since its release, many variations of the original formulation have been proposed [106]. Figure 1 provides an overview of the original, also known as *vanilla*, architecture. It consists of two main components: a stack of L encoders, and a stack of L decoders.

The encoder takes an input sequence and processes it by applying **attention** mechanisms to capture the relationships between different elements of the sequence. It also makes use of a feed forward layer, residual connections, and normalization layers. Attention allows each element to attend to all other elements in the sequence, enabling the model to establish the relative importance of each token to the others. This is done following a Query-Key-Value attention model, with multiple attention heads. Each head independently considers three weight matrices that are learned alongside the model, $\mathbf{W^Q}$, $\mathbf{W^K}$, and $\mathbf{W^V}$. The matrix representation of u input sentences of size N tokens, $\mathbf{X} \in \mathbb{R}^{N \times u}$, is multiplied by those matrices, originating $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{N \times d_v}$, which are used in the self-attention computation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} = \mathbf{A}\mathbf{V} \ . \tag{27}$$

In the previous equation, d_k is the dimension of the rows of $\mathbf{K}$, and its square root is used as a normalization factor for more stable gradients during training. The outputs from each attention head are concatenated and projected back to the original dimension. $\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)$ is referred to as the **attention matrix**, and is the major bottleneck in the Transformer, as it has quadratic complexity with respect to the input size N, given the product $\mathbf{Q}\mathbf{K}^\top$. Multiple variations of the vanilla Transformer were proposed to deal with this specific issue ([179], discussed in Section 6.1).

The decoder takes the output from the encoder and generates a target sequence. It also uses attention mechanisms, with slight differences. The first attention layer on the decoder is often referred to as

masked, causal, or auto-regressive attention, as it is restricted to attend only to previous tokens. The second attention layer performs cross attention, as takes the learned projections for $\mathbf{K}$ and $\mathbf{V}$ from the encoder, and the projections for $\mathbf{Q}$ from the previous decoder layer.

The positional encoding step consists in adding a vector that captures positional information to the input embeddings. The computation of this vector was originally proposed such that each position corresponds to a sinusoid, alternating between sine and co-sine functions:

$$\begin{aligned} \mathrm{PE}(p, 2i) &= \sin(p/10000^{2i/N}) \ , \\ \mathrm{PE}(p, 2i+1) &= \cos(p/10000^{2i/N}) \ . \end{aligned} \tag{28}$$

In the previous equation, p is the position and i is the dimension. Other strategies for positional embeddings have been proposed, for instance learnable positional embeddings [58], rotary position embeddings [172], or ALiBi [143], which does not use positional embeddings explicitly, but captures positional information by adding a linear bias to the attention computation.

From the Transformer architecture, two distinct types of model families emerged: BERT [42] and GPT [148]. GPT, short for Generative Pre-trained Transformer, is known for its application in language generation tasks. It is trained using a causal language modeling objective, where the model learns to predict the next word in a sentence based on the preceding words, i.e., given the set of tokens $T = \{t_1, t_2, ..., t_n\}$ and a context of k tokens, maximize the log-likelihood $\sum_i \log \mathbb{P}(t_i | t_{i-k}, ..., t_{i-1}; \Theta)$, where the conditional probability is modeled by a Transformer decoder with parameters Θ. As for BERT, which stands for Bidirectional Encoder Representations from Transformers, it uses a masked language modeling objective that focuses on capturing bidirectional context of words in a sentence, using only the encoder of the Transformer. Given the predominance of BERT-based models for dense retrieval, BERT will be further discussed in the next section.

3.5 Bidirectional Encoder Representation from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a neural language model that achieved state-of-the-art performance on multiple tasks when it was published [42], also having been made publicly available. The available models were trained on books and Wikipedia data in a semi-supervised fashion. Multiple variations on training data and objectives have been proposed (e.g., RoBERTa [114] and ELECTRA [26]), and variations of BERT have been tuned for different textual domains (e.g., BioBERT [98] for biomedical text, FinBERT [4] for the finance domain, SciBERT [8] for scientific articles, and BERTweet [131] for tweets).

BERT consists of Transformer encoders. Instead of feeding the output of the encoder to the decoder, each token representation is kept as an embedding. For dense retrieval, the token representations are pooled into a single vector to represent the whole input sentence. Common pooling strategies include mean pooling, or CLS-pooling, where a special [CLS] token is prepended to the input string, and its embedding is used as a representation for the whole sequence.

To pre-train the encoder stack, the authors proposed two tasks. The first was to find the right word for some masked position of a sentence. As such, they employed a masked language modeling objective, where 15% of the input is masked. Besides, some words are randomly replaced by another, and the model is asked to find the right one for the position. The output associated with the masked/replaced words is fed to a feed-forward network with softmax, yielding probabilities for words over the vocabulary that are used to predict the missing word. The second objective was to detect whether or not two sentences follow one another. A special token, [SEP], is used to separate the sentences, and the output associated with the [CLS] token is fed to a classifier.

4 Dense Retrieval

Dense retrievers are mostly used for the task of **first-stage retrieval**, also known as **full-retrieval**. In this task, the objective is, for a given query, to retrieve the top-N most relevant documents from a collection. As such, given the potentially large size of the document collection, methods for this task need to be efficient. In this section, the bi-encoder architecture is introduced as the standard architecture for the task in hand. Furthermore, a review on state-of-the art dense retrieval is conducted.

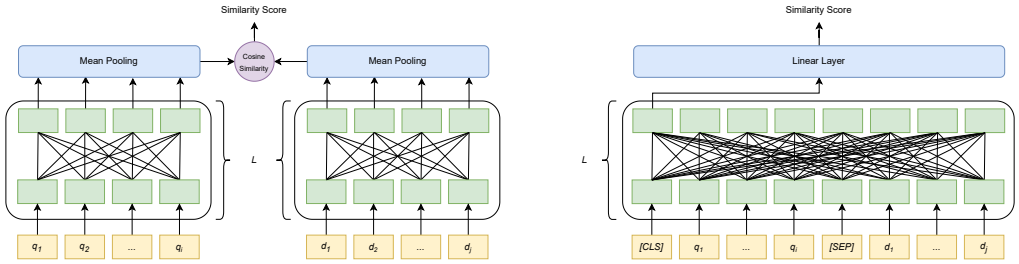

Figure 2: Bi-encoder as two stacks of L encoders using mean pooling (left), and cross-encoder as a stack of L encoders, with CLS-pooling.

4.1 The Bi-Encoder and Cross-Encoder Architectures

Dense retrievers are mostly based on the **bi-encoder** architecture. It works by independently mapping queries and documents to a D-dimensional embedding space, where D, while high (usually 768), is much lower than the vocabulary size V discussed in Section 2.2. Consider Figure 2, left, where a query $q = \{q_1, q_2, ..., q_i\}$ is being compared to a document $d = \{d_1, d_2, ..., d_j\}$. The encoder, usually a Transformer encoder, independently processes the query and the document. Then, the token embeddings are pooled into a single representation, and the cosine similarity function (Eq. 4) is used to compute the similarity between the query and the document representations, i.e., a proxy of the relevance of the document towards the query. This architecture is contrasted by the **cross-encoder** (Figure 2, right), where a query and a document are concatenated into a single sentence, and a representation that encompasses the pair is generated and fed to a linear layer to produce a similarity score. This allows for bi-directional attention between query and document terms, resulting in more accurate relevance judgments. However, it has inferior computational performance as it does not allow for offline document indexing, since representations are not independent. As such, the cross-encoder is mostly used for **re-ranking** tasks [135, 141, 134, 214], i.e., given the top-N documents for a query, re-rank them. For neural full-retrieval (i.e., the top-N retrieval from the whole collection), which is the focus of this survey, bi-encoders are the preferred architecture.

For bi-encoders, the encoder that processes queries and documents may be the same, as proposed in works such as SBERT [150], or follow approaches such as DPR [86] which use an independent encoder for queries and documents. A computationally efficient, yet powerful, variation in the similarity computation named late interaction was introduced in ColBERT [87], which given a query q, k' documents will be retrieved for each token representation in the query (i.e., documents are scored against the individual query token representations), producing $|q| \times k'$ results. Then, from those, the unique ones will be re-ranked, exhaustively scoring each document with respect to the whole query.

Given the independence between the computation of query and document representations, offline indexing of individual document representations is possible when using a bi-encoder, followed by search through methods that support the fast execution of maximum inner product searches such as FAISS [82]. If exhaustive search over the whole collection is not feasible, approximate nearest neighbor (ANN) techniques such as Product Quantization [77] and Hierarchical Navigable Small Worlds [123] can be employed for faster retrieval.

Product Quantization involves dividing high-dimensional vectors into smaller subvectors and quantizing each subvector independently, reducing the computational complexity of similarity search by decomposing it into multiple lower-dimensional searches. As for the Hierarchical Navigable Small Worlds technique, it constructs a hierarchical graph structure, where each node in the graph represents a document, enabling efficient navigation between nodes based on their similarity, and allowing for fast retrieval by exploring a small subset of the entire document collection.

4.2 Dense Retrieval Training Setup

Bi-encoders are built on top of pre-trained Transformer encoders, such as the aforementioned BERT model. For dense retrieval, the language semantics captured during pre-training are often not enough, with embeddings from out-of-the-box BERT-like models achieving subpar results [150]. As such,

a fine-tuning step is used, which usually leverages large amounts labeled data. For instance, given a set of train queries Q, for each $q \in Q$ there is a set of documents from the whole collection D , $D_q \in D$, that are relevant towards q, which is usually very small (e.g., 1). Since human annotators only label relevant documents, the set of irrelevant documents towards q, $D_{\overline{q}}$, needs to be sampled from D, which can be very large.

As such, when building a batch for training, the process of choosing negative examples for a given query, often referred to as **negative sampling**, is a crucial process. If sampled randomly, the similarity between the documents and the query is likely to be very low, not contributing to the learning process. Conversely, choosing the documents that despite being negatives are the most similar to the query (hardest negatives), will make the distance between the embeddings very small, which may lead to collapsed models [197].

Multiple works exclusively used randomly sampled negative examples [71, 146]. This has been shown to be suboptimal [211], as it may lead to a large number of easy negatives, motivating the need for hard negative sampling techniques. For instance, lexical techniques such as BM25 have been used to sample negative examples [57, 86], as well as dense retrieval models, which can be done in a static fashion, i.e., sample the negatives for the training queries prior to training (e.g., REALM [64]), or dynamically, as in strategies such as ANCE [201], where the model being trained samples the negative examples from an ANN index of the corpus, which is parallelly updated with the learning process to select more realistic negative training instances. Extensions include ANCE-Tele [174], which showed that dynamic sampling may induce the model into catastrophic forgetting [89] of negative groups, proposing the usage of teleportation negatives, which consist of a combination of negative examples from previous iterations, and estimation of potential hard negatives in future training iterations using the neighbors of the positive document. ADORE and STAR [211] are two training techniques that also employ dynamic hard negative sampling, but argue that random negatives also have to be introduced in order to achieve more stable training.

Models such as RocketQA [146] use the notion of **denoised negative sampling**, where in order to avoid false negatives, only the examples scoring below a threshold are considered. Besides, this model also uses in-batch and cross-batch negatives. In-batch negatives consist of using the positive documents associated with a query as a negative example for other queries in the same batch. Cross-batch negatives can be employed on multi-GPU training, where examples in different batches are shared. These techniques highly boost the effective training data, and its results can be further improved by considering careful batch construction [67], e.g., following a strategy where similar queries are grouped together in the same batch, since hard negative documents sampled for a given query are probably also challenging for similar queries.

During training, given the training pairs (q, d) in a batch, and their respective labels, the bi-encoder computes the similarity between them following the logic presented in Figure 2, left. Then, loss functions such as the cross-entropy can be used to train the model [87]:

$$L_{\text{crossentropy}} = -\sum_{d \in D'_q} \log(\text{score}(q, d)) - \sum_{d \in D'_{\overline{q}}} \log(1 - \text{score}(q, d)) \ . \tag{29}$$

In the previous equation, D'_q and $D'_{\overline{q}}$ are the sets of positive and negative documents for query q, respectively, provided within the same training batch. Other loss functions such as the hinge loss, presented in Section 3.2, are also commonly used [150], rewritten here for convenience:

$$L_{hinge} = \sum_{d^+ \in D'_q, d^- \in D'_{\overline{q}}} \max(0, \text{score}(q, d^-) - \text{score}(q, d^+) + \epsilon) \ , \tag{30}$$

where the objective is to have positive documents scoring at least the value of the margin ϵ higher than a negative one. PAIR [151] introduced a slightly different approach to training, which leverages a combination of query-centric and document-centric similarity, i.e., instead of considering only the similarity between queries and documents, the similarity between positive and negative documents within a batch is also used to guide the loss.

4.3 Improving Dense Retrieval Models

In order to improve on the standard training setup previously described, multiple methods have been proposed [102]. For instance, RocketQA [146] and AugSBERT [180] propose a **data augmentation** technique that consist in using a strong cross-encoder to label (q, d) pairs that are not associated with relevance judgements, to be used for the training of a bi-encoder. The new pairs to be labeled with the cross-encoder can be sampled using different techniques, such as (i) random sampling, (ii) kernel density estimation, which aims to get a similar label distribution to the set of already labeled pairs, (iii) BM25 sampling, where the top-k documents returned by BM25 for each query are selected, and (iv) semantic search sampling which uses a model trained on the already labeled set to retrieve the top-k most similar sentences. **Query generation** is another specific case of data augmentation that can be used e.g. for Information Retrieval tasks. Doc2Query [136] is the first work that used transformer-based models to generate queries given an input document. Later, DocT5Query [133] was proposed as an extension that replaced the transformer with a T5 model. In Doc2Query– [59], the authors argue that since generation models can hallucinate, some output queries may not be useful, showing that properly filtering the queries improves their results in downstream tasks. More recently, Large Language Models have been used for query generation. For instance, InPars [14] proposes to few-shot prompt GPT-3 [19] with context examples, and also shows that query filtering, by removing queries with low generation probability, improves results on downstream Information Retrieval tasks. This model has been extended in InPars-v2 [78], where the model was changed to an open-source alternative, GPT-J [192], and a neural model is used to assign relevance scores to the generated query and respective document, removing pairs below a threshold. The authors of DRAGON [105] provide an in-depth analysis of the above augmentation techniques (query and label), proposing a training approach with diverse artificial queries and sources of labelling supervision, achieving strong results in retrieval benchmarks.

Current research directions also analyze and improve the robustness of dense retrievers against misspellings [168] and typos [215]. Efficiency is also a concern, with methods for embedding dimension reduction being proposed [121, 116]. Regarding larger models, their generalization ability has been studied [132], and their use in the task of **knowledge distillation** for dense retrieval is also an active research area.

The objective of knowledge distillation is to use a larger, stronger model to guide the training of a smaller one. One possibility for this technique is to use a cross-encoder to teach a bi-encoder. As previously stated, the properties of bi-encoders make them computationally more efficient. However, cross-encoders are better at capturing query-document interactions, usually performing better than bi-encoders, in terms of result quality. Previous studies have addressed distillation processes that go from larger models to smaller versions, for example by using the outputs of the large model as targets [80, 165]. However, when distilling from a cross-encoder to a bi-encoder, trying to fit the cross-encoder's outputs directly is not optimal, since the range and magnitude of the cross-encoder scores (sigmoid of logits) differ from those of the bi-encoder (cosine similarity of vector embeddings). Approaches for cross-architecture distillation have been attempted, for example by optimizing the margin between scores [66], e.g., given a triplet (q, d^+, d^-), a teacher model M_t and a student model M_s, minimize the mean squared error between $\text{score}_{M_t}(q, d^+) - \text{score}_{M_t}(q, d^-)$ and $\text{score}_{M_s}(q, d^+) - \text{score}_{M_s}(q, d^-)$.

The same authors proposed TCT-ColBERT [103], consisting of a dual encoder student, and a ColBERT [87] teacher, i.e., distilling from a bi-encoder with late interaction to a regular bi-encoder. The teacher computes scores for the positive documents and in-batch negatives, and the student is trained by combining the standard cross entropy loss component with a Kullback-Leibler divergence between the distributions of student and teacher scores. In subsequent work the TCT-ColBERT was further improved with hard negative sampling [104].

Regarding the size and performance of the teacher, other authors [107] argued that it is common for a strong teacher model to result in a bad student through distillation, due to the gap between teacher and student. They proposed PROD as a progressive distillation method for dense retrievers, as this type of distillation had shown to be effective in other downstream tasks [152, 1]. PROD consists of two progression mechanisms. First, via teacher progressive distillation, the capability of teacher models is gradually improved by resorting to different architectures. Then, via data progressive distillation, student models start learning from all the data, moving to samples of the data that contain examples where the student performs poorly.

Another technique that is common to state-of-the-art dense retrievers is **in-domain pre-training** [137, 55, 126, 198]. For instance, for the specific case of the CoCondenser [54, 55], a masked language modeling objective is augmented with a contrastive loss. This contrastive loss is query-agnostic, as it only considers the document corpus. More specifically, given a set of documents $\{d_1, ..., d_n\}$, a pair of spans is extracted from each one, $\{(s_{1,1}, s_{1,2}), ..., (s_{n,1}, s_{n,2})\}$. Given the representation for each span in a batch, $r_{s_{i,j}}$, the contrastive loss can be written as:

$$L_{i,j}^{\text{contrastive}} = -\log \frac{\exp(\text{sim}_{\cos}(r_{s_{i,1}}, r_{s_{i,2}}))}{\sum_{k=1}^{n} \sum_{l=1}^{2} \mathbb{I}_{ij \neq kl} \exp\left(\text{sim}_{\cos}(r_{s_{h,i}}, r_{s_{j,k}})\right)} \ . \tag{31}$$

The previous equation resembles noise-contrastive estimation [63], the idea being that spans close together should have similar representations, while those in different documents should have different representations. RetroMAE [115, 199] is a different retrieval-oriented pre-training approach that aims to force an higher quality from the representation output by the encoder. This approach follows a masked auto-encoding workflow, composed of an encoder and a shallow decoder, where the input text is masked, with the decoder input being masked more aggressively. Given the representation from the encoder and the highly masked input text, the shallow decoder attempts to reconstruct the original text via masked language modeling.

4.4 Bridging Dense and Sparse Retrieval

While dense retrieval has shown good results, some authors [24, 167] have argued that (i) dense retriever training suffers from objective mismatch between the language model pre-training task (masked language modeling) and the dense vector similarity-based fine-tuning, and (ii) dense indexes have a large size, potentially contributing to high retrieval latency. While some techniques to address the above remarks have been discussed previously in this document (in-domain pre-training and approximate nearest neighbor search), other authors have studied the combination of neural models and sparse retrieval [209, 75, 96], allowing the usage of term-based inverted indexes for faster ad-hoc search, and benefiting from explicit lexical match and interpretability.

In order to have the importance of a term varying with its context, DeepCT [38] proposed a weighting scheme that uses a BERT model to generate term embeddings, and a linear regression model to predict term weights for each term. Still, the vocabulary mismatch inherent to most sparse retrieval methods is not addressed with this approach. Doc2Query [136] and DocT5Query [133] address the mismatch problem by using a Transformer-based model to generate queries for a given document. Then, documents are extended with the generated queries. This re-weights terms and expands the document with new vocabulary, which can mitigate the mismatch problem upon a user query.

SpaDE [24] can be seen as a combination of the two previous approaches, as it uses a BERT model with a linear model on top to learn term weights, and a second BERT encoder with a MLM head to learn expansion terms. Other recent methods include the SPLADE [48, 95, 49] model family, which try to improve on SparTerm [7] by adding a log-saturation effect in the weight estimation, which prevents certain terms from dominating, and ensures sparsity in the document and query representations. The ranking loss used in SPLADE is based on the maximization of the probability of a positive document given a query, together with a sparsity regularization penalty [140]. The previously mentioned RetroMAE [115, 199] has a sparse counterpart, LexMAE [167]. In this approach, the same structure of RetroMAE's masked auto-encoding workflow is used, composed of an encoder and a shallow decoder that tries to reconstruct an aggressively masked version of the input text. However, the decoder is conditioned on a bag-of-words bottleneck, that is generated by multiplying the encoder's embedding matrix by an intermediate representation obtained through pooling logit vectors.

5 Information Retrieval Evaluation

Evaluation is a crucial aspect of information retrieval systems, allowing to assess their effectiveness by comparing to different strategies. However, evaluating retrieval systems is not without its challenges and complexities. One such challenge arises from the inherent bias of IR collections, where the vast majority of documents are irrelevant to a given query. Another significant aspect is the type

of relevance assessments, which can be categorized into binary or graded. Addressing these non-trivialities is essential for ensuring robust evaluation methodologies that provide meaningful insights into the performance of information retrieval systems.

5.1 Metrics

This section summarizes the most commonly used metrics for IR. More details and other metrics can be consulted in the following sources: [5, 125].

Precision and Recall

Precision and Recall can be considered the backbone metrics for IR systems, as they assess the ability of a model to retrieve relevant documents. The precision computes the fraction of retrieved documents that are relevant. Let S be the set of retrieved documents for a given query, and S^* the set of relevant documents. The precision can be computed as follows:

$$\mathrm{P} = \frac{|S \cap S^*|}{|S|} \ . \tag{32}$$

As for recall, it computes the percentage of relevant documents returned:

$$\mathrm{R} = \frac{|S \cap S^*|}{|S^*|} \ . \tag{33}$$

Some systems may need an higher recall and others an higher precision. However, having both an high precision and recall has proven not to be possible in normal situations. As such, a trade-off between precision and recall may sometimes be needed. One metric that is often considered for that purpose is the F_1 measure, which consists in the harmonic mean between precision and recall:

$$\mathrm{F}_1 = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}} \ . \tag{34}$$

It is often useful to compute the above metrics at the k^{th} retrieved document, i.e., P@k=$\frac{|S_k \cap S^*|}{k}$, R@k=$\frac{|S_k \cap S^*|}{|S^*|}$, and $\mathrm{F}_1@k$ = $\frac{2\mathrm{P@}k\mathrm{R@}k}{\mathrm{P@}k+\mathrm{R@}k}$.

One metric that is often used in other tasks that is not usually considered in an IR setup is the accuracy, i.e., the fraction of classifications that are correct. This is due to data skewness in large collections, where most documents are irrelevant towards the information need. As such, a system that rates all documents as irrelevant would have a very high accuracy, while not being useful for the end user.

The above metrics are based on set theory, i.e., do not consider the ranking order of the documents. One possibility to deal with this is through the usage of precision-recall curves, which show how precision changes as recall increases. However, there are also specific metrics to deal with such a scenario.

Order-aware metrics

One of the most standard order-aware metrics in the IR community is the Mean Average Precision (MAP) [173]. Given a collection of queries, it computes the arithmetic mean for the Average Precision (AP) for each query, i.e., the average of the precision value obtained for the set of top-k documents existing after each relevant document is retrieved:

$$\mathrm{MAP} = \frac{\sum_{q \in Q} \mathrm{AP}(q)}{|Q|} \ , \tag{35}$$

$$\mathrm{AP(q)} = \frac{\sum_{i=1}^{n} \mathrm{P@}i \times \mathbb{I}_i}{|S_q^*|} \ . \tag{36}$$

In the previous equations, Q is the set of all queries, S_q^* is the set of relevant documents for query q and $\mathbb{I}_i$ evaluates to 1 if the document at rank i is relevant to q and to 0 otherwise.

Another commonly used metric is the Mean Reciprocal Rank (MRR) [30], which focuses on the position of the first relevant document in the ranked list for each query.

$$\text{MRR@k} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}(i,k)} \ , \tag{37}$$

where rank(i,k) is a function that, given the top-k results for the i^{th} query, returns the position (rank) of the first relevant document.

Graded relevance

All metrics discussed so far only consider binary relevance, i.e., they do not account for graded relevance (i.e., instead of binary judgments, documents can have different levels of relevance). The Normalized Discounted Cumulative Gain (nDCG) [76] is the standard metric for cases where the full order of the items and their relative relevance should be taken into account. It starts by computing the DCG:

$$\text{DCG@}k = \mathbb{R}_1 + \sum_{i=2}^{k} \frac{\mathbb{R}_i}{\log_2(i)} \ , \tag{38}$$

where $\mathbb{R}_i$ evaluates to the relevance value of the document at rank i. The index of the document up to which the ranking is considered is represented by k. The DCG is normalized with the ideal DCG, i.e., the DCG of a perfectly sorted result:

$$\text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \ . \tag{39}$$

5.2 Benchmarks

As the field of IR has evolved, so have the benchmarks used to evaluate the performance of various information retrieval systems. In recent years, there has been a significant shift towards utilizing larger and more comprehensive benchmarks to reflect the scale and complexity of modern information retrieval tasks. For instance, Cranfield, one of the pioneering collections, consists of 225 queries and 1398 documents. Nowadays, datasets such as the ClueWeb22 [139] contain approximately 10 billion documents. This section provides a summary of the most commonly used benchmarks for IR experiments, focusing on their dimensions and associated tasks.

5.2.1 ClueWeb

The ClueWeb is a line of datasets consisting of large collections of web documents of different languages. The first iteration, ClueWeb09, contains 1 billion web documents in 10 languages, also providing a web graph with approximately 8 billion out-links. The most recent iteration, ClueWeb22 [139], contains 10 billion documents. For the most important webpages within the collection (200 million, of which 43% are in English), there is information regarding the HTML, the web graph, clean text, semantic annotations, and screenshots.

5.2.2 TREC

The U.S. National Institute of Standards and Technology (NIST) has run an IR evaluation series since 1992, accommodated in the Text Retrieval Conference (TREC). There are many different TREC tracks. For instance, the TREC Deep Learning track focuses on document and passage retrieval, with new queries and graded relevance judgments released every year since 2019 [32, 33, 34, 35].

The TREC Web track, that ran from 1999 to 2014, leveraged the ClueWeb dataset for multiple tasks, such as named page finding, topic distillation, and ad-hoc retrieval. The dataset Robust04 [188] was built by considering topics (queries) that performed poorly in the TREC Ad-hoc retrieval track, that is also no longer running.

Table 1: Summary of the approaches used in MS-MARCO data.

	Sparse Retrieval					Dense Retrieval					
	BM25	**DocT5Query [133]**	**DeepCT [38]**	**SPLADE++ [49]**	**LexMAE [167]**	**ANCE [201]**	**Contriever [74]**	**CoCondenser [55]**	**RetroMAE [115]**	**CoTMAE [198]**	**DRAGON+ [105]**
MS-MARCO (Dev)											
MRR@10	18.7	21.5	24.3	38.8	42.6	33.9	34.1	38.6	35.4	39.9	39.0
R@1000	59.2	89.1	91.3	98.2	98.8	95.9	97.9	98.4	97.5	98.5	98.6
TREC Deep Learning 2019											
NDCG@10	51.2	-	55.1	74.3	73.7	-	67.8	71.5	68.8	70.0	74.4
TREC Deep Learning 2020											
NDCG@10	47.7	-	-	71.8	72.8	-	66.1	68.1	71.4	67.8	72.3

Other recent tracks include TREC Covid [153], which dealt with retrieval on top of medical documents regarding the Covid-19, and TREC Clinical Trials [154], which evaluates the effectiveness of systems in retrieving relevant clinical trial documents for a specific patient description.

5.2.3 BEIR

BEIR [181] is a heterogeneous benchmark containing datasets for multiple IR tasks. It contains 19 datasets, where some can be used for ad-hoc retrieval multiple domains (MS-MARCO [20], Robust04 [188], TREC Covid [153], NFCorpus [17], BioASQ [184], TREC News [171]), but it also contains datasets for tasks such as Question Answering (Natural Questions [92], FiQA [122], HotpotQA [204]), Duplicate Question Retrieval (CQADupStack [69], Quora [147]), Argument Retrieval (ArguAna [190], Touche-2020 [13]), Citation Prediction (SciDocs [27]), and Fact Checking (FEVER [182], Climate-FEVER [44], SciFact [191]).

Some of the above collections (MS-MARCO, NFCorpus, BioASQ, NaturalQuestions, HotpotQA, FiQA, and FEVER) have training sets of considerable sizes, allowing in-domain training of retrieval models. The other collections only contain test splits, hence zero-shotting models trained on other collections is often the used strategy.

5.2.4 MS-MARCO

MS-MARCO [20] is a benchmark dataset that can be used for document retrieval and passage retrieval. It also has support for the tasks of key-phrase extraction, natural language generation, and question answering. For document retrieval, the available data comprises 3.2 million documents and over 1 million queries from BING. From those queries, 362 thousand are labeled with human annotations concerning binary relevance judgments. For passage retrieval, the 3.2 million documents are split into 8 million passages. Clean anchor-text data has been released for this dataset [50].

Associated with the MS-MARCO data, the ORCAS [31] dataset provides click-based data for 1.4 million documents, consisting of 10 million queries associated with 18 million connections to documents.

More recently, a second iteration of the dataset has been released, which contains 12 million documents, and there is also a mapping between passages and the documents from which they were extracted.

The previously mentioned TREC Deep Learning tracks leverage data from this collection, the first years from version 1, and since 2021 from version 2. Table 1 summarizes the some of the most commonly referred to methods and their results in these datasets.

6 Research Directions

This section covers possible research directions that are in line with what was addressed in this review. For each topic, related work is briefly introduced to contextualize the task, and open problems are addressed.

6.1 Long Document Retrieval

The dense retrievers addressed in this survey are based on the Transformer architecture, which has a computational bottleneck on the self-attention mechanism, due to its quadratic complexity with respect to input size. This hinders the possibility of using vanilla transformers on long inputs, which is why usually input sentences are truncated to 512 tokens. In the following subsections, two techniques to overcome this issue are addressed, namely aggregation strategies, where the long document is split into multiple passages, and long-document transformer architectures, which leverage local, global, and sparse attention patterns to lower the memory cost of performing the attention computation.

6.1.1 Aggregation Strategies

In the context of document retrieval, aggregations strategies rely on segmenting the document into passages. Then, either only the first passage is used (FirstP), or all passages are mapped to the embedding space, and the score for the document is obtained by max (MaxP) or sum (SumP) pooling the passage scores [37, 206]. Combinations of these strategies are possible, for instance K-MaxP-Avg [23] which considers the average of the top-K passages scores as the document score.

Instead of aggregating scores, other works such as PARADE [101] study the aggregation of the vectorial representations of document chunks. The authors considered simple aggregation methods, such as max pooling the representations, and more complex ones, for instance using a Transformer encoder to output a pooled representation for the concatenation of passage representations. MORES+ [56] feed the independent representations of query and document chunks to a Transformer encoder that uses self-attention over the query tokens, and cross-atention between query tokens and document tokens.

Fusion-in-Decoder [73] (FiD) is similar to MORES+, but uses both the encoder and decoder of the Transformer. It was originally proposed for Question Answering, where multiple question-context pairs are encoded independently, and the representations are then concatenated and fed as input to the decoder. This is particularly useful for tasks with short output sizes, since the decoder has $O(nk + k^2)$ complexity, where n is the input size and k is the output size. FiD-Light [68] and FiDO [39] are extensions to the original FiD, that deal with lowering the computational cost of decoding. The former proposes to reduce input length by a constant factor, while the latter removes cross-attention from some decoder layers entirely. Both strategies achieve similar inference speedups, but FiDO has overall better results. While for Question Answering tasks the contexts may be fully independent, retrieval tasks may receive as input one large document that potentially needs global contextualization. SLED [72] is a FiD-based model that deals with such tasks by splitting the document in multiple overlapping chunks, achieving competitive results in the SCROLLS benchmark [166].

Possible future work in this area includes the usage of FiD-like models for dense retrieval. The above models are mostly used as cross-encoders, yielding a score rather than a representation. Although dense retrievers are usually encoder-only, the decoder can also be included and used as a pooler, i.e., decode a single token given the encoder output and use its representation as the embedding for the input sentence. This way, instead of having e.g. MaxP indexes (where each document will be represented by one vector per passage), a FiD technique can be used when encoding a document, by splitting it into passages in the encoder, and fusing the representations prior to decoding the single token. This approach scales linearly with the number of passages when training and generating the representations, and has much smaller index size when compared to the MaxP approach since each document has a single representation.

6.1.2 Long Document Transformers

Work related to Transformers for long inputs make the assumption that the full attention matrix may not be necessary to model all interactions between the tokens. For instance, the Longformer [9] uses a local window attention, and chooses some tokens to hold global attention. Similarly, BigBird [208]

follows the same principles, but chooses random tokens in the input to have global attention, achieving better performance on QA tasks than the Longformer. LSG Attention [29] also leverages the previous mechanisms. However, special tokens to be attended globally are preprended to the input (instead of selecting tokens from the input).

Instead of selecting the tokens that are going to be attended based on their location, some methods have been proposed to select them based on their content. For instance, the Reformer [90] uses an attention method based on locality-sensitive hashing (LSH). In the self-attention, when computing $\mathrm{softmax}(\mathbf{QK}^T)$, for each $q_i \in \mathbf{Q}$, this method selects only the subset of $\mathbf{K}$ containing the top-N vectors closer to q_i. Learning-to-Hash attention [176] aims to overcome some issues with models based on the previous mechanism, such as the randomness of LSH, and the assumption of shared embedding space between query and key vectors.

A different school of thought leverages low-rank approximations of the attention matrix instead of selecting tokens for which attention is computed. For instance, the Linformer [194] applies low-rank decomposition to the attention matrix, and the Nyströmformer [202] uses the Nyström method to approximate the attention matrix.

The above and other models have been used in benchmarks such as the Long Range Arena [178], but its usage in retrieval tasks is still underrepresented. Previous studies [18] have argued that retrieval collections such as MS-MARCO and Robust04 are not particularly well-suited for long-range modeling evaluation, as they have a relevance bias towards the first passage of the document. With novel collections such as ClueWeb22, a larger study on this aspect could be conducted, and the inclusion of queries that motivate modeling larger sequences studied, also considering the ideas behind recent work on Large Language Models context size [109, 45, 22, 185].

6.2 Structured Document Retrieval

Regarding the specific case of web documents, besides being long, they also contain an inherent structure provided by their HTML DOM tree, as well as visual features. Given the availability of benchmarks such as ClueWeb22 that contain this sort of information, and previous work showing that it may be useful to incorporate it during the training of retrieval models [62], it is now possible to study the actual contribution that this data can have to web search.

One possible path is to study how to incorporate structure and visual information when training dense retrievers. For instance, for the task of the keyphrase extraction, BLING-KPE [200] uses hybrid positional embeddings to represent structural information and visual features, by concatenating a vector representation of those features to the regular positional encodings. SMART-KPE [195] also leverages structural information and visual features, but does so by having a second transformer to encode them. There are also Transformer variations to deal with this type of data, for instance the LayoutXML [203] which is a multimodal transformer capable of processing text, layout, and the image of a page. The Webformer [193] is mostly used for web information extraction, having been finetuned with HTML data and containing special tokens for HTML tags. As such, devising ways to incorporate structure information when generating representations for web documents could improve retrieval results. That can be done for instance through prompting, in the case of passage-level features, or in the case of token-level features through positional embeddings or variations to the attention mechanism. Pre-training tasks regarding structural information and visual features can also be employed. Other authors [62] have proposed masked dom node prediction, parent-child modeling, brother node modeling, and children order recovery. With the ClueWeb22 data, other tasks can be considered, for instance title extraction, visual feature prediction, and field type classification.

Another direction is to include structure and visual information when filtering a document corpus for training. Following work on scalling laws for neural language models stating the training data should grow as the number of parameters of a model grows [85, 65, 2], recent studies have been assessing the impact of different filtering mechanisms and data design strategies when selecting data for training [97, 117]. The same could be verified for a retriever: if training on a very large collection, corpus filtering should be conducted. For that effect, the inclusion of structure and visual features can provide additional context and cues that aid in understanding the content and relevance of a document.

6.3 Information Retrieval and Large Language Models

Currently, the paradigm of machine learning for natural language processing tasks has shifted tremendously in a short time span, favoring transfer learning from pre-trained language models to downstream tasks over creating specialized models for each task. This has resulted in groundbreaking state-of-the-art performance across multiple fields, with foundational models like LLaMa [183], PALM 2 [3] and GPT-4 [138] leading the way.

Some fine-tuning approaches have demonstrated impressive zero-shot generalization to different tasks, for instance the usage reinforcement learning with human feedback [46], and instruct fine-tuning [196]. This, however, still requires training models with a very high number of parameters, which hinders end-to-end tuning especially under limited hardware conditions. Solutions like low-rank adaptation (LoRA) [70] aim to overcome this issue by selecting only a small percentage of trainable parameters, by decomposing weight matrices into low-rank factors, and quantizing the model to 16-bit, or even 4-bit in QLoRA [41].

Prompt-based methods [110], are particularly useful when training is not possible, since instead of fine-tuning the model, the generation capabilities of LLMs are directly exploited. Previous work has shown very convincing evidence on how prompt usage substantially improves results [213], specially the few-shot prompting technique [19], which conditions the output generation on a small set of provided examples from the task in hand.

Given the relevance of such models, it is important to assess how to use them to aid retrieval and vice-versa. Retrieval-augmented generation is an active research area, leveraging retrieval models to obtain more accurate and contextually relevant responses to factual questions [79, 100, 16, 149]. Traditional language models generate sentences based on input prompts but may lack context understanding, leading to inaccuracies. Retrieval-augmented generation includes a retrieval component that retrieves relevant information from a document collection based on the query. The generation is then conditioned on the retrieved information, enhancing the generation process.

Query generation is a specific case of data augmentation that was discussed in this review, as it can be used in IR tasks e.g. for document expansion [136, 133, 59] or synthetic data generation [14, 78]. As such, leveraging LLMs to generate queries can be useful for IR collections lacking training data, or for potentially better expansion terms. Moreover, current methods such as InPars [14, 78] truncate the document text on the first tokens, which can bias data towards the first passage. Methods to generate queries focusing on the whole sequence may create more robust relevance distributions and promotes long sequence modeling for retrieval.

Finally, there are studies using LLMs as re-rankers through prompting [144, 175]. This can be done in a pair-wise fashion (i.e., prompt the model with two documents and a query, and ask which one is the most relevant one towards the query), or list-wise fashion (i.e., prompt the model with a numbered list of documents and a query, and ask to re-arrange the numbers with respect to relevance towards the query).

7 Conclusion

This survey presented a review on dense retrieval. It covered a wide range of topics, starting from the early classic IR models, to the current state-of-the art neural retrieval methods. The training setup of dense retrievers was addressed, and multiple extensions such as negative sampling and knowledge distillation explained. Models that combine neural with sparse methods were also examined. For completion, this document also contains a brief background in the machine learning techniques used in this task (architectures, optimizers, and loss functions), and an overview of IR benchmarks and metrics. Finally, challenges and promising research directions are highlighted in order to promote future work on the area.

References

[1] A. Andonian, S. Chen, and R. Hamid. Robust Cross-Modal Representation Learning with Progressive Self-Distillation. In *Conference on Computer Vision and Pattern Recognition*, 2022.

[2] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, E. Chu, J. H. Clark, L. E. Shafey, Y. Huang, K. Meier-Hellstern, G. Mishra, E. Moreira, M. Omernick, K. Robinson, S. Ruder, Y. Tay, K. Xiao, Y. Xu, Y. Zhang, G. H. Ábrego, J. Ahn, J. Austin, P. Barham, J. A. Botha, J. Bradbury, S. Brahma, K. Brooks, M. Catasta, Y. Cheng, C. Cherry, C. A. Choquette-Choo, A. Chowdhery, C. Crepy, S. Dave, M. Dehghani, S. Dev, J. Devlin, M. Díaz, N. Du, E. Dyer, V. Feinberg, F. Feng, V. Fienber, M. Freitag, X. Garcia, S. Gehrmann, L. Gonzalez, and et al. Palm 2 technical report. *ArXiv*, abs/2305.10403, 2023.

[3] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. PaLM 2 Technical Report. *ArXiv*, abs/2305.10403, 2023.

[4] D. Araci. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *ArXiv*, abs/1908.10063, 2019.

[5] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[6] J. Bai, J. Nie, G. Cao, and H. Bouchard. Using query contexts in information retrieval. In *International Conference on Research and Development in Information Retrieval*, 2007.

[7] Y. Bai, X. Li, G. Wang, C. Zhang, L. Shang, J. Xu, Z. Wang, F. Wang, and Q. Liu. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. *ArXiv*, abs/2010.00768, 2020.

[8] I. Beltagy, K. Lo, and A. Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[9] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The Long-Document Transformer. *ArXiv*, abs/2004.05150, 2020.

[10] J. Bhogal, A. MacFarlane, and P. W. H. Smith. A review of ontology based query expansion. *Information Processing and Management*, 43(4):866–886, 2007.

[11] B. Billerbeck and J. Zobel. Document expansion versus query expansion for ad-hoc retrieval. In *Australasian Document Computing Symposium*, 2005.

[12] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast Differentiable Sorting and Ranking. In *International Conference on Machine Learning*, 2020.

[13] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen. Overview of Touché 2020: Argument Retrieval. In *Conference and Labs of the Evaluation Forum*, 2020.

[14] L. H. Bonifacio, H. Abonizio, M. Fadaee, and R. F. Nogueira. InPars: Data Augmentation for Information Retrieval using Large Language Models. *ArXiv*, abs/2202.05144, 2022.

[15] A. Bookstein, D. R. Swanson, et al. Probabilistic models for automatic indexing. *Journal of the Association for Information Science and Technology*, 25(5):312–316, 1974.

[16] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving Language Models by Retrieving from Trillions of Tokens. In *International Conference on Machine Learning*, 2022.

[17] V. Boteva, D. G. Ghalandari, A. Sokolov, and S. Riezler. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *European Conference on Information Retrieval Research*, 2016.

[18] L. Boytsov, T. Lin, F. Gao, Y. Zhao, J. Huang, and E. Nyberg. Understanding Performance of Long-Document Ranking Models through Comprehensive Evaluation and Leaderboarding. *ArXiv*, abs/2207.01262, 2022.

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. In *Conference on Neural Information Processing Systems*, 2020.

[20] D. F. Campos, T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, and B. Mitra. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Workshop on Cognitive Computation co-located with the Annual Conference on Neural Information Processing Systems*, 2016.

[21] C. Carpineto and G. Romano. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Computing Surveys*, 44(1):1:1–1:50, 2012.

[22] S. Chen, S. Wong, L. Chen, and Y. Tian. Extending Context Window of Large Language Models via Positional Interpolation. *ArXiv*, abs/2306.15595, 2023.

[23] X. Chen, B. He, L. Sun, and Y. Sun. ICIP at TREC-2020 Deep Learning Track. In *Text REtrieval Conference*, 2020.

[24] E. Choi, S. Lee, M. Choi, H. Ko, Y.-I. Song, and J. Lee. SpaDE: Improving sparse representations using a dual document encoder for first-stage retrieval. In *International Conference on Information and Knowledge Management*, 2022.

[25] K. W. Church and W. A. Gale. Poisson mixtures. *Natural Language Engineering*, 1(2): 163–190, 1995.

[26] K. Clark, M. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*, 2020.

[27] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Meeting of the Association for Computational Linguistics*, 2020.

[28] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *International Conference on Information and Knowledge Management*, 2005.

[29] C. Condevaux and S. Harispe. LSG attention: Extrapolation of pretrained transformers to long sequences. In *Conference on Knowledge Discovery and Data Mining*, volume 13935, pages 443–454, 2023.

[30] N. Craswell. Mean Reciprocal Rank. In *Encyclopedia of Database Systems*. 2009.

[31] N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck. ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search. *ArXiv*, abs/2006.05324, 2020.

[32] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. Overview of the TREC 2019 Deep Learning Track. In *Text REtrieval Conference (TREC)*, 2020.

[33] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos. Overview of the TREC 2020 Deep Learning Track. In *Text REtrieval Conference (TREC)*, 2021.

[34] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and J. Lin. Overview of the TREC 2021 Deep Learning Track. In *Text REtrieval Conference (TREC)*, 2022.

[35] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, J. Lin, E. Voorhees, and I. Soboroff. Overview of the TREC 2022 Deep Learning Track. In *Text REtrieval Conference (TREC)*, 2023.

[36] W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of documentation*, 1979.

[37] Z. Dai and J. Callan. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Conference on Research and Development in Information Retrieval*, 2019.

[38] Z. Dai and J. Callan. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *ArXiv*, abs/1910.10687, 2019.

[39] M. de Jong, Y. Zemlyanskiy, J. Ainslie, N. FitzGerald, S. Sanghai, F. Sha, and W. Cohen. FiDO: Fusion-in-Decoder optimized for stronger performance and faster inference. *ArXiv*, abs/2212.08153, 2022.

[40] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6): 391–407, 1990.

[41] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *ArXiv*, abs/2305.14314, 2023.

[42] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[43] F. Diaz, B. Mitra, and N. Craswell. Query Expansion with Locally-Trained Word Embeddings. In *Annual Meeting of the Association for Computational Linguistics*, 2016.

[44] T. Diggelmann, J. L. Boyd-Graber, J. Bulian, M. Ciaramita, and M. Leippold. CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims. *arXiv*, abs/2012.00614, 2020.

[45] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, and F. Wei. LongNet: Scaling Transformers to 1, 000, 000, 000 Tokens. *ArXiv*, abs/2307.02486, 2023.

[46] P. Fernandes, A. Madaan, E. Liu, A. Farinhas, P. H. Martins, A. Bertsch, J. G. C. de Souza, S. Zhou, T. Wu, G. Neubig, and A. F. T. Martins. Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation, 2023.

[47] B. M. Fonseca, P. B. Golgher, B. Pôssas, B. A. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *International Conference on Information and Knowledge Management*, pages 696–703, 2005.

[48] T. Formal, B. Piwowarski, and S. Clinchant. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *International Conference on Research and Development in Information Retrieval*, 2021. URL `https://doi.org/10.1145/3404835.3463098`.

[49] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *International Conference on Research and Development in Information Retrieval*, 2022.

[50] M. Fröbe, S. Günther, M. Probst, M. Potthast, and M. Hagen. The Power of Anchor Text in the Neural Retrieval Era. In *European Conference on Information Retrieval Research*, 2022.

[51] N. Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[52] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The Vocabulary Problem in Human-System Communication. *Communications of the ACM*, 30(11), 1987.

[53] D. Ganguly, D. Roy, M. Mitra, and G. J. F. Jones. Word Embedding based Generalized Language Model for Information Retrieval. In *International Conference on Research and Development in Information Retrieval*, pages 795–798, 2015.

[54] L. Gao and J. Callan. Condenser: a Pre-training Architecture for Dense Retrieval. In *Empirical Methods in Natural Language Processing*, 2021.

[55] L. Gao and J. Callan. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[56] L. Gao and J. Callan. Long Document Re-ranking with Modular Re-ranker. In *International Conference on Research and Development in Information Retrieval*, 2022.

[57] L. Gao, Z. Dai, Z. Fan, and J. Callan. Complementing Lexical Retrieval with Semantic Residual Embedding. *ArXiv*, abs/2004.13969, 2020.

[58] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252, 2017.

[59] M. Gospodinov, S. MacAvaney, and C. Macdonald. Doc2Query–: When Less is More. In *European Conference on Information Retrieval Research*, 2023.

[60] M. Grbovic, N. Djuric, V. Radosavljevic, and N. Bhamidipati. Search Retargeting using Directed Query Embeddings. In *International Conference on World Wide Web*, 2015.

[61] W. R. Greiff. A Theory of Term Weighting Based on Exploratory Data Analysis. In *International Conference on Research and Development in Information Retrieval*, 1998.

[62] Y. Guo, Z. Ma, J. Mao, H. Qian, X. Zhang, H. Jiang, Z. Cao, and Z. Dou. Webformer: Pre-training with Web Pages for Information Retrieval. In *International Conference on Research and Development in Information Retrieval*, 2022.

[63] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[64] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. REALM: Retrieval-Augmented Language Model Pre-Training. *ArXiv*, abs/2002.08909, 2020.

[65] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training Compute-Optimal Large Language Models. *ArXiv*, abs/2203.15556, 2022.

[66] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *ArXiv*, abs/2010.02666, 2020.

[67] S. Hofstätter, S. Lin, J. Yang, J. Lin, and A. Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *International Conference on Research and Development in Information Retrieval*, 2021.

[68] S. Hofstätter, J. Chen, K. Raman, and H. Zamani. FiD-Light: Efficient and Effective Retrieval-Augmented Text Generation. *ArXiv*, abs/2209.14290, 2022.

[69] D. Hoogeveen, K. M. Verspoor, and T. Baldwin. CQADupStack: A Benchmark Data Set for Community Question-Answering Research. In *Australasian Document Computing Symposium (ADCS)*, 2015.

[70] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

[71] J. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang. Embedding-based Retrieval in Facebook Search. In *Conference on Knowledge Discovery and Data Mining*, 2020.

[72] M. Ivgi, U. Shaham, and J. Berant. Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11:284–299, 2023.

[73] G. Izacard and E. Grave. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2021.

[74] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research*, 2022.

[75] K. Jang, J. Kang, G. Hong, S. Myaeng, J. Park, T. Yoon, and H. Seo. Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[76] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[77] H. Jégou, M. Douze, and C. Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[78] V. Jeronymo, L. H. Bonifacio, H. Abonizio, M. Fadaee, R. de Alencar Lotufo, J. Zavrel, and R. F. Nogueira. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. *ArXiv*, abs/2301.01820, 2023.

[79] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active Retrieval Augmented Generation. *ArXiv*, abs/2305.06983, 2023.

[80] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for Natural Language Understanding. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[81] T. Joachims. Optimizing Search Engines using Clickthrough Data. In *International Conference on Knowledge Discovery and Data Mining*, 2002.

[82] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *ArXiv*, abs/1702.08734, 2017.

[83] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 60(5):493–502, 2004.

[84] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha. AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing. *ArXiv*, abs/2108.05542, 2021.

[85] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling Laws for Neural Language Models. *ArXiv*, abs/2001.08361, 2020.

[86] V. Karpukhin, B. Oguz, S. Min, P. S. H. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[87] O. Khattab and M. Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *International Conference on Research and Development in Information Retrieval*, 2020.

[88] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[89] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[90] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*, 2020.

[91] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *Conference on Research and Development in Information Retrieval*, pages 194–201, 2004.

[92] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[93] F. W. Lancaster. *Information retrieval systems : Characteristics, Testing, and Evaluation*. John Wiley & Sons New York, 2nd ed. edition, 1979.

[94] F. W. Lancaster and E. G. Fayen. *Information retrieval: On-Line*. Melville Pub. Co Los Angeles, 1973.

[95] C. Lassance and S. Clinchant. An Efficiency Study for SPLADE Models. In *International Conference on Research and Development in Information Retrieval*, 2022.

[96] C. Lassance, T. Formal, and S. Clinchant. Composite Code Sparse Autoencoders for First Stage Retrieval. In *Conference on Research and Development in Information Retrieval*, 2021.

[97] A. Lee, B. Miranda, and S. Koyejo. Beyond Scale: the Diversity Coefficient as a Data Quality Metric Demonstrates LLMs are Pre-trained on Formally Diverse Data. *ArXiv*, abs/2306.13840, 2023.

[98] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4): 1234–1240, 2020.

[99] J. H. Lee. Properties of Extended Boolean Models in Information Retrieval. In *Conference on Research and Development in Information Retrieval*, 1994.

[100] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Task. In *Annual Conference on Neural Information Processing Systems*, 2020.

[101] C. Li, A. Yates, S. MacAvaney, B. He, and Y. Sun. PARADE: Passage Representation Aggregation for Document Reranking. *ArXiv*, abs/2008.09093, 2020.

[102] J. Lin, R. F. Nogueira, and A. Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. 2021.

[103] S. Lin, J. Yang, and J. Lin. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *ArXiv*, abs/2010.11386, 2020. URL `https://arxiv.org/abs/2010.11386`.

[104] S. Lin, J. Yang, and J. Lin. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Workshop on Representation Learning for NLP*, pages 163–173, 2021.

[105] S. Lin, A. Asai, M. Li, B. Oguz, J. Lin, Y. Mehdad, W. Yih, and X. Chen. How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval. *ArXiv*, abs/2302.07452, 2023.

[106] T. Lin, Y. Wang, X. Liu, and X. Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[107] Z. Lin, Y. Gong, X. Liu, H. Zhang, C. Lin, A. Dong, J. Jiao, J. Lu, D. Jiang, R. Majumder, and N. Duan. PROD: Progressive Distillation for Dense Retrieval. In *International World Wide Web Conference*, 2023.

[108] H. Liu, Z. Li, D. Hall, P. Liang, and T. Ma. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training. *arXiv*, abs/2305.14342, 2023.

[109] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the Middle: How Language Models Use Long Contexts. *ArXiv*, abs/2307.03172, 2023.

[110] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9), 2023.

[111] Q. Liu, M. J. Kusner, and P. Blunsom. A Survey on Contextual Embeddings. *ArXiv*, abs/2003.07278, 2020.

[112] T. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[113] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Conference on Research and Development in Information Retrieval*, 2004.

[114] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692, 2019.

[115] Z. Liu and Y. Shao. RetroMAE: Pre-training Retrieval-oriented Transformers via Masked Auto-Encoder. *ArXiv*, abs/2205.12035, 2022.

[116] Z. Liu, H. Zhang, C. Xiong, Z. Liu, Y. Gu, and X. Li. Dimension Reduction for Efficient Dense Retrieval via Conditional Autoencoder. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

[117] S. Longpre, G. Yauney, E. Reif, K. Lee, A. Roberts, B. Zoph, D. Zhou, J. Wei, K. Robinson, D. Mimno, and D. Ippolito. A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity. *ArXiv*, abs/2305.13169, 2023.

[118] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[119] H. P. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[120] Y. Lv and C. Zhai. Lower-Bounding Term Frequency Normalization. In *International Conference on Information and Knowledge Management*, 2011.

[121] X. Ma, M. Li, K. Sun, J. Xin, and J. Lin. Simple and Effective Unsupervised Redundancy Elimination to Compress Dense Vectors for Passage Retrieval. In *Conference on Empirical Methods in Natural Language Processing*, pages 2854–2859, 2021.

[122] M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, and A. Balahur. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *International World Wide Web Conference*, pages 1941–1942, 2018.

[123] Y. A. Malkov and D. A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[124] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[125] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[126] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, and X. Song. COCO-LM: Correcting and Contrasting Text Sequences for Language Model Pretraining. In *Annual Conference on Neural Information Processing Systems*, 2021.

[127] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*, 2013.

[128] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Annual Conference on Neural Information Processing Systems*, 2013.

[129] B. Mitra, E. T. Nalisnick, N. Craswell, and R. Caruana. A Dual Embedding Space Model for Document Ranking. *ArXiv*, abs/1602.01137, 2016.

[130] F. Mosteller and D. L. Wallace. Applied Bayesian and classical inference : the case of the Federalist papers. 1984.

[131] D. Q. Nguyen, T. Vu, and A. T. Nguyen. BERTweet: A pre-trained language model for English Tweets. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[132] J. Ni, C. Qu, J. Lu, Z. Dai, G. H. Ábrego, J. Ma, V. Y. Zhao, Y. Luan, K. B. Hall, M. Chang, and Y. Yang. Large Dual Encoders Are Generalizable Retrievers. In *Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, 2022.

[133] R. Nogueira and J. Lin. From doc2query to docTTTTTquery. *Technical Report*, 6, 2019.

[134] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[135] R. F. Nogueira and K. Cho. Passage Re-ranking with BERT. *ArXiv*, abs/1901.04085, 2019.

[136] R. F. Nogueira, W. Yang, J. Lin, and K. Cho. Document Expansion by Query Prediction. *ArXiv*, abs/1904.08375, 2019.

[137] B. Oguz, K. Lakhotia, A. Gupta, P. S. H. Lewis, V. Karpukhin, A. Piktus, X. Chen, S. Riedel, S. Yih, S. Gupta, and Y. Mehdad. Domain-matched Pre-training Tasks for Dense Retrieval. In *Findings of the Association for Computational Linguistics*, 2022.

[138] OpenAI. GPT-4 Technical Report. *ArXiv*, abs/2303.08774, 2023.

[139] A. Overwijk, C. Xiong, X. Liu, C. VandenBerg, and J. Callan. ClueWeb22: 10 Billion Web Documents with Visual and Semantic Information. *ArXiv*, abs/1902.06006, 2022.

[140] B. Paria, C. Yeh, I. E. Yen, N. Xu, P. Ravikumar, and B. Póczos. Minimizing FLOPs to Learn Efficient Sparse Representations. In *International Conference on Learning Representations*, 2020.

[141] R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *International Conference on Knowledge Discovery and Data Mining*.

[142] J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[143] O. Press, N. Smith, and M. Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*, 2022.

[144] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, and M. Bendersky. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. *ArXiv*, 2023.

[145] Y. Qiu and H. Frei. Concept Based Query Expansion. In *International Conference on Research and Development in Information Retrieval.*, 1993.

[146] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, and H. Wang. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*, 2021.

[147] Quora. Quora Dataset Release: Question Pairs. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2017.

[148] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[149] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham. In-Context Retrieval-Augmented Language Models. *ArXiv*, abs/2302.00083, 2023.

[150] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing*, 11 2019.

[151] R. Ren, S. Lv, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J. Wen. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. In *Findings of the Association for Computational Linguistics*, 2021.

[152] M. Rezagholizadeh, A. Jafari, P. S. M. Saladi, P. Sharma, A. S. Pasand, and A. Ghodsi. Pro-KD: Progressive Distillation by Following the Footsteps of the Teacher. In *International Conference on Computational Linguistics*, 2022.

[153] K. Roberts, T. Alam, S. Bedrick, D. Demner-Fushman, K. Lo, I. Soboroff, E. M. Voorhees, L. L. Wang, and W. R. Hersh. Searching for scientific evidence in a pandemic: An overview of TREC-COVID. *Journal of Biomedical Informatics*, 121:103865, 2021.

[154] K. Roberts, D. Demner-Fushman, E. M. Voorhees, S. Bedrick, and W. R. Hersh. Overview of the TREC 2021 Clinical Trials Track. In *Text REtrieval Conference (TREC)*, 2021.

[155] S. Robertson and H. Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

[156] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 1977.

[157] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the Association for Information Science and Technology*, 27(3):129–146, 1976.

[158] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, 65(6), 1958.

[159] D. Roy, D. Ganguly, S. Bhatia, S. Bedathur, and M. Mitra. Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance. In *International Conference on Information and Knowledge Management*, 2018.

[160] I. Ruthven, M. Lalmas, and C. J. van Rijsbergen. Incorporating user search behavior into relevance feedback. *Journal of the Association for Information Science and Technology*, 54(6): 529–549, 2003.

[161] G. Salton. Developments in automatic text retrieval. *Science*, pages 974–980, 1991.

[162] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[163] G. Salton, A. Wong, and C. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11), 1975.

[164] G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[165] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[166] U. Shaham, E. Segal, M. Ivgi, A. Efrat, O. Yoran, A. Haviv, A. Gupta, W. Xiong, M. Geva, J. Berant, and O. Levy. SCROLLS: standardized comparison over long language sequences. In *Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, 2022.

[167] T. Shen, X. Geng, C. Tao, C. Xu, X. Huang, B. Jiao, L. Yang, and D. Jiang. LexMAE: Lexicon-Bottlenecked Pretraining for Large-Scale Retrieval. In *International Conference on Learning Representations*, 2023.

[168] G. Sidiropoulos and E. Kanoulas. Analysing the Robustness of Dual Encoders for Dense Retrieval Against Misspellings. In *International Conference on Research and Development in Information Retrieval*, pages 2132–2136, 2022.

[169] A. Singhal. Modern Information Retrieval: A Brief Overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.

[170] N. A. Smith. Contextual Word Representations: A Contextual Introduction. *ArXiv*, abs/1902.06006, 2019.

[171] I. Soboroff, S. Huang, and D. Harman. TREC 2020 News Track Overview. In *Text Retrieval Conference*, 2020.

[172] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv*, abs/2104.09864, 2021.

[173] W. Su, Y. Yuan, and M. Zhu. A Relationship between the Average Precision and the Area Under the ROC Curve. In *International Conference on The Theory of Information Retrieval*, pages 349–352, 2015.

[174] S. Sun, C. Xiong, Y. Yu, A. Overwijk, Z. Liu, and J. Bao. Reduce Catastrophic Forgetting of Dense Retrieval Training with Teleportation Negatives. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

[175] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, and Z. Ren. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *ArXiv*, abs/2304.09542, 2023.

[176] Z. Sun, Y. Yang, and S. Yoo. Sparse Attention with Learning to Hash. In *International Conference on Learning Representations*, 2022.

[177] T. Tao, X. Wang, Q. Mei, and C. Zhai. Language Model Information Retrieval with Document Expansion. In *Conference of the North American Chapter of the Association of Computational Linguistics*, 2006.

[178] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long Range Arena : A Benchmark for Efficient Transformers. In *International Conference on Learning Representations*, 2021.

[179] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient Transformers: A Survey. *ACM Computing Surveys*, 55(6), 2022.

[180] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[181] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[182] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819, 2018.

[183] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models, 2023.

[184] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, Y. Almirantis, J. Pavlopoulos, N. Baskiotis, P. Gallinari, T. Artières, A. N. Ngomo, N. Heino, É. Gaussier, L. Barrio-Alvers, M. Schroeder, I. Androutsopoulos, and G. Paliouras. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16: 138:1–138:28, 2015.

[185] S. Tworkowski, K. Staniszewski, M. Pacek, Y. Wu, H. Michalewski, and P. Milos. Focused Transformer: Contrastive Training for Context Scaling. *ArXiv*, abs/2307.03170, 2023.

[186] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In *Annual Conference on Neural Information Processing Systems*, 2017.

[187] L. Vilnis and A. McCallum. Word Representations via Gaussian Embedding. In *International Conference on Learning Representations*, 2015.

[188] E. Voorhees. Overview of the TREC 2004 Robust Retrieval Track. In *Text REtrieval Conference (TREC)*, 2005.

[189] E. M. Voorhees. Query Expansion Using Lexical-Semantic Relations. In *International Conference on Research and Development in Information Retrieval*, 1994.

[190] H. Wachsmuth, S. Syed, and B. Stein. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *Meeting of the Association for Computational Linguistics*, pages 241–251, 2018.

[191] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, and H. Hajishirzi. Fact or Fiction: Verifying Scientific Claims. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[192] B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, 2021.

[193] Q. Wang, Y. Fang, A. Ravula, F. Feng, X. Quan, and D. Liu. WebFormer: The Web-page Transformer for Structure Information Extraction. *ArXiv*, abs/2202.00217, 2022.

[194] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-Attention with Linear Complexity. *CoRR*, abs/2006.04768, 2020.

[195] Y. Wang, Z. Fan, and C. P. Rosé. Incorporating Multimodal Information in Open-Domain Web Keyphrase Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.

[196] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. *ArXiv*, abs/2212.10560, 2022.

[197] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl. Sampling Matters in Deep Embedding Learning. In *International Conference on Computer Vision*, pages 2840–2848, 2017.

[198] X. Wu, G. Ma, M. Lin, Z. Lin, Z. Wang, and S. Hu. ConTextual Masked Auto-encoder for Dense Passage Retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4738–4746, 2023.

[199] S. Xiao and Z. Liu. RetroMAE v2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models. *ArXiv*, abs/2211.08769, 2022.

[200] L. Xiong, C. Hu, C. Xiong, D. Campos, and A. Overwijk. Open Domain Web Keyphrase Extraction Beyond Language Modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019.

[201] L. Xiong, C. Xiong, Y. Li, K. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*, 2021.

[202] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. In *Conference on Artificial Intelligence*, pages 14138–14148, 2021.

[203] Y. Xu, T. Lv, L. Cui, G. Wang, Y. Lu, D. Florêncio, C. Zhang, and F. Wei. LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding. *ArXiv*, abs/2104.08836, 2021.

[204] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

[205] J.-Y. Yeh, J. Y. Lin, H.-R. Ke, and W.-P. Yang. Learning to rank for information retrieval using genetic programming. In *Proceedings of ACM SIGIR Workshop on Learning to Rank for Information Retrieval*, 2012.

[206] Z. A. Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin. Applying BERT to Document Retrieval with Birch. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[207] C. T. Yu and G. Salton. Precision weighting—an effective automatic indexing method. *Journal of the ACM*, 23(1):76–88, 1976.

[208] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big Bird: Transformers for Longer Sequences. In *Conference on Neural Information Processing Systems*, 2020.

[209] H. Zamani, M. Dehghani, W. B. Croft, E. G. Learned-Miller, and J. Kamps. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *International Conference on Information and Knowledge Management*, 2018.

[210] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson. Microsoft Cambridge at TREC 13: Web and Hard Tracks. In *Text Retrieval Conference*, 2004.

[211] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *International Conference on Research and Development in Information Retrieval*, 2021.

[212] L. Zhao and J. Callan. Term Necessity Prediction. In *Iternational Conference on Information and Knowledge Management*, 2010.

[213] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen. A Survey of Large Language Models, 2023.

[214] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. *ArXiv*, abs/2210.10634, 2022.

[215] S. Zhuang and G. Zuccon. CharacterBERT and Self-Teaching for Improving the Robustness of Dense Retrievers on Queries with Typos. In *International Conference on Research and Development in Information Retrieval*, 2022.