Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Rolv-Arild Braaten
August Bobakk Indal

Exploring the Viability of Multilingual Zero-shot Neural Document Retrieval

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

NTNU
Norwegian University of
Science and Technology

Rolv-Arild Braaten
August Bobakk Indal

Exploring the Viability of Multilingual Zero-shot Neural Document Retrieval

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

With the fast progress of deep learning, major strides have been made in many fields of computer science. However, until recently, neural retrieval methods have struggled to see success beyond that of methods developed in the 1990s. Large search companies have recently started applying neural retrieval to their search, but they currently have a monopoly on datasets containing large amounts of non-English labeled data. This makes it necessary to develop retrieval methods that can perform well without seeing labeled data (zero-shot) regardless of language, to elevate retrieval performance for everyone. This Thesis explores the field of neural information retrieval (IR), and the potential for application of these neural models in multilingual and zero-shot settings. Select models are tested on the Text REtrieval Conference (TREC) Spanish dataset in a zero-shot fashion to evaluate their potential.

Results show that for multilingual zero-shot retrieval, the most important aspect is pre-training as much as possible on data with labeled relevance and getting good at the ranking task before transferring to new languages. Using a multilingual model provides a boost in performance, and increasing input length also has positive effects for document-level retrieval. The best model tested uses a multilingual BERT (Bidirectional Encoder Representations from Transformers) model trained on the Microsoft Machine Reading Comprehension (MS MARCO) passage retrieval dataset, and displays an 11% increase over previous state-of-the-art results on TREC Spanish by achieving an nDCG@20 (Normalized Discounted Cumulative Gain of the top 20 retrieved documents) of 0.739.

A limitation of typical Transformers such as BERT is their inability to view the entire document at once due to length restrictions. In neural IR, the use of efficient Transformers, designed specifically for handling longer sequences, has thus far remained unexplored. In this Thesis, two efficient Transformer architectures (BigBird and Longformer) are tested, with comparable results to the previous state-of-the-art.

Sammendrag

Med den raske framgangen som har vært i dyp læring har man sett store steg innen flere informatikkfelt. Til tross for dette har man ikke før nylig sett vellykket bruk av dyp læring innen informasjonsgjennfinning (IR). De store søkemotorselskapene har begynt å ta i bruk nevrale metoder til søk, men problemet er at de sitter med et monopol på store datasett med ikke-engelsk annotert data. Dette gjør det nødvendig å utvikle metoder innenfor IR som kan gjøre det bra uten annotert data (zero-shot) og uansett språk, slik at man kan heve nivået på IR-systemer for alle. Denne oppgaven utforsker feltet nevral informasjonsgjenfinning og potensialet til flerspråklige nevrale modeller for søk, uten bruk av annotert data. Utvalgte modeller er testet på Text REtrieval Conference (TREC) Spanish datasettet for å evaluere deres potensial.

Resultatene viser at for flerspråklig zero-shot informasjonsgjennfinning er det viktigste å trene på så mye engelsk annotert data som mulig, og bli god på informasjonsgjennfinningsdelen før man overfører det man har lært til nye språk. Bruk av flerspråklige modeller øker ytelsen i forhold til engelskspråklige modeller. Å gi modeller mulighet til å se på større deler av dokumentet om gangen er også en fordel. Den beste modellen som er testet bruker en flerspråklig versjon av BERT (Bidirectional Encoder Representations from Transformers) trent på Microsoft Machine Reading Comprehension (MS MARCO) passage retrieval datasettet. Den får et resultat som er 11% bedre enn tidligere state-of-the-art på TREC Spanish, med en nDCG@20 (Normalized Discounted Cumulative Gain på topp 20 hentede dokumenter) score på 0.739.

En begrensning med vanlige Transformer-modeller slik som BERT er deres manglede evne til å se på hele dokumentet på en gang fordi de har restriksjoner på sekvenslengden. Innenfor nevral IR har effektive Transformere laget spesielt for å håndtere lengre sekvenser, til nå ikke vært utforsket. I denne rapporten utforskes to effektive Transformer-arkitekturer (BigBird og Longformer), med resultat som er sammenlignbart med tidligere state-of-the-art.

Preface

This Thesis was written as the final assignment to complete the degree of Master of Science in Computer Science at the Norwegian University of Science and Technology (NTNU). It was written in collaboration with Norconsult AS and supervised by Björn Gambäck.

We want to thank Björn for his invaluable guidance for the duration of the Thesis, as well as the preceding specialization project. Thanks to Norconsult and Thomas Hjelde Thoresen for offering the project and inspiring its direction. Also, a big thanks the NTNU HPC group for providing IDUN to use for model training. Thanks to Mostafa Dehghani and Andrew Yates for allowing the use of their diagrams. Finally, a big thanks to Joakim Sæther for his support, help, and friendship throughout the entire Bachelor's and Master's degree.

Braaten, Rolv-Arild
Indal, August Bobakk

Trondheim, 10th June 2021

Contents

List of Figures

List of Tables

1 Introduction

Search is one of the most ubiquitous technologies in the world today. Google is the most visited website in the world, with billions of searches every day (Nayak, 2019). Many of the world's most popular websites, as well as devices like smartphones and computers, include search bars for more straightforward navigation. Organizations use search to manage internal documents.

Recent advances in natural language processing (NLP), most notably Transformers (Vaswani et al., 2017), have also enabled advances in search technology, with Google using Transformers to improve 7% of their search results globally (Raghavan, 2020). A better understanding of the contents of both documents and queries is critical to improving search results by enabling human-like comprehension of text.

Despite these advances, a significant drawback of many techniques using deep learning is that it requires training with labeled data. For this reason, neural-augmented search is only available to a select few companies with the data to support it. Overcoming this obstacle is a critical aspect of applying these new techniques effectively for varying domains and languages.

This Master's Thesis will explore the field of neural information retrieval, or neural IR for short. The main objective is obtaining relevant documents from large collections by searching with keywords, taking advantage of deep language understanding enabled by neural networks. A lacking area of research is the use of these methods for languages other than English where there is a lack of labeled data. This will be the main focus of the Thesis.

1.1 Background and Motivation

This Master's Thesis is done in cooperation with Norconsult AS, looking to improve their in-house document search solution. Their current solution is a prototype using TF-IDF+LSI (detailed in subsection 2.2.2 and subsection 2.2.4). It is only used by a small pilot group, making it difficult to collect enough labeled data. Motivated by this, a few restrictions are intentionally placed on our solution, directing our research:

- Because of a lack of labeled training data, the solution should work with very few, and preferably zero, labeled training samples. This is challenging since many modern solutions depend on deep learning, which relies on massive amounts of data

to function optimally. Using a model without training on labeled examples from the relevant dataset is known as *zero-shot* learning. In the case of retrieval this means not training with labels of whether or not documents are relevant or not.

- The data contains a mix of primarily Norwegian and English, meaning a solution should be designed to work across different languages. This is also an obstacle, as most datasets and architectures are designed with only the English language in mind. The term *multilingual* is used for models and datasets designed with multiple languages in mind.

Beyond this, there are no restrictions regarding the choice of model or dataset. During the pre-study to this Thesis, a distinct lack of research within both zero-shot and multilingual neural retrieval was noticed, further driving the decision to make this the focus.

1.2 Goals and Research Questions

Due to the aforementioned lack of labels in conjunction with multilingual data, this Thesis has one main goal, which is further divided into two research questions:

Goal *Find a document ranking model that gives good results across languages using zero-shot learning.*

The aim is to find or construct a model that requires little to no training examples on the target dataset, but maintains good performance. Breaking this down, there are two main steps constructed to achieve this.

Research question 1 *What are the zero-shot capabilities of neural document ranking models?*

Zero-shot models combat the problem of not having labeled data. In particular, exploring the current neural document ranking models for zero-shot learning is essential in order to not only measure them against each other, but also to outline the properties each model possesses and draw inspiration from them. Based on a literature review outlining the methods used in neural IR today, testing will be performed to achieve this.

Research question 2 *How can neural document ranking models be adapted to work for new languages?*

Most of the available datasets with labels for information retrieval are in English, how well can models trained on English perform on non-English datasets. We wish to explore the cross-lingual capabilities of these models. Specifically, how they can be modified to support new languages apart from the obvious solution of throwing more data at the problem. By using English datasets for training and then the TREC-Spanish dataset for evaluation, we can simulate zero-shot transfer learning and then measure the results.

1.3 Research Method

Due to the nature of neural IR, with multiple labeled datasets available, this Thesis will use an experimental approach. Since the focus is on zero-shot performance across languages, models will be tested by transferring models between training datasets, and without specific training on the evaluation dataset in a different language.

1.4 Contributions

List of main contributions, ordered by estimated importance.

C1 Testing of various pre-trained language models, fine-tuned on MS MARCO or Robust04, on the TREC Spanish dataset, including the first use of Efficient Transformer architectures on the retrieval task.

C2 A comparison of these models and their properties in light of test results, showing that zero-shot multilingual neural retrieval is feasible, but with room for future improvements.

C3 Arguments for selecting Transformer-based retrieval models as the main focus of research rather than dense- or interaction-based models.

1.5 Thesis Structure

This Thesis is constructed as such:

Chapter 2 is an overview of the theoretical background required to understand the current state of neural IR. It covers classic IR terminology, as well as the machine learning models that make up the most modern models.

Chapter 3 covers the specifics of these modern neural ranking models, categorized by model type, as well as the role of zero-shot transfer. It also describes the datasets used by previous researchers to evaluate performance.

Chapter 4 describes the architecture used, as well as the reasoning behind the choices made. It also covers the tools and libraries which were used to conduct experiments.

Chapter 5 presents the experimental plan, setup and the results.

Chapter 6 is an evaluation of the research questions an discussion of these results and what went right/wrong and why.

Chapter 7 concludes the Thesis, mentions contributions and proposes directions for future research.

2 Background Theory

To understand neural IR, fundamental knowledge of both traditional retrieval systems and machine learning is required. This chapter will give an introduction to the most important terminology and components used in today's research. First, basic text processing concepts are covered. Second, relevant classical retrieval methods and metrics. And lastly, some fundamental machine learning architectures, including the recently successful Transformer, and some loss functions which are often used for training of retrieval models.

2.1 Text processing

With any computer system dealing with text, it is essential to process it somehow so that the system gets the most useful information possible. These are a couple of the most common ways.

Stop words
: In most languages, many words serve as "filler" words and contribute little to the meaning of the text by themselves. In English, these are words like 'the', 'and', 'an'. These words are often filtered out to reduce noise, data usage, and processing, although with the obvious downside of potential information loss.

Bag-of-words
: Instead of using the entire text as input, it is common instead to count up the number of occurrences of each word and store it as a multiset (bag). This removes all contextual and grammatical information but is useful for shallow comparisons of texts or as features for a model. For example, the text 'it was the best of times, it was the worst of times' would become *{'it': 2, 'was': 2, 'the': 2, 'of': 2, 'times': 2, 'best': 1, 'worst': 1}*

N-grams
: An extension to bag-of-words, which is common for reducing the vocabulary size is *n-grams*. Instead of regarding each word as a token, each set of n consecutive characters, or words, is the token. This is typically done using an n of 1 (unigram), 2 (bigram) or 3 (trigram). For instance, with character trigrams, the word 'banana' will be converted into *{'ban': 1, 'ana': 2, 'nan': 2}*.

Textpiece

Additional methods exist as an extension to the idea of n-grams. Instead of picking sequences of a given length, one could imagine picking whichever sequences occur the most often. A word like 'the' could have its own token, while words like 'parking' might be split into 'park' and 'ing', potentially conserving linguistic nuances like morphemes. This is what methods like WordPiece (Wu et al., 2016) and SentencePiece (Kudo and Richardson, 2018) do, tokenizing the text based on common substrings, with their main difference being the length of text it uses as its basis. For the most powerful machine learning models, this is typically what is used.

2.2 Retrieval

In broad terms, text retrieval is about finding **documents** containing relevant information. Each document consists of several **terms**, typically a word or a group of consecutive words which represent some potentially relevant information. To find the relevant documents, the user issues a **query**, which states the information needs in some way. In addition, the word **passage** is common. It refers to a shorter document, often a small extract such as a sentence or paragraph from a larger document, for which the information content is more specific. The set of all terms is called the **vocabulary**, and the set of all documents is called the **corpus**.

2.2.1 Evaluation

Pertinent to any retrieval task is measuring the effectiveness of a model, which can be done in several different ways. Typically one wants all the most relevant documents near the top of a ranking, which the measurement should reflect. These are some of the most common measures:

Precision measures the proportion of retrieved documents that are relevant.

$$P = \frac{|relevant \cap retrieved|}{|retrieved|} \tag{2.1}$$

It is common to use this along with some threshold, fixing the number of retrieved items. This is called *precision at n*, or *P@n* for short. For instance, measuring the precision at 20 documents retrieved would be *P@20*.

Recall is similar to precision but gives the proportion of all relevant documents that are retrieved.

$$R = \frac{|relevant \cap retrieved|}{|relevant|} \tag{2.2}$$

Average Precision ($AveP$) measures the average precision at each relevant document.

$$AveP = \frac{\sum_{k=1}^{n} P(k) \times rel(k)}{|relevant|} \tag{2.3}$$

Where $P(k)$ is the precision at rank k, n is the number of documents in the collection, and $rel(k)$ is a function indicating whether the k-th item is relevant (1 if relevant, 0 if not). An extension to average precision is Mean Average Precision (MAP), which averages AveP across several queries.

Mean Reciprocal Rank (MRR) is the multiplicative inverse of the position of the first relevant document, averaged over a set of queries. This can also be viewed as the harmonic mean of the positions of the first relevant document.

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{FirstRel_q} \tag{2.4}$$

Where Q is the set of queries, q is a query and $FirstRel_q$ is the position of the first relevant document in the ranking for q. Like precision, a cutoff value n can be included (denoted as $MRR@n$), meaning if the position of the first relevant document is larger than n, then the reciprocal rank would be zero (equivalent to $FirstRel$ approaching infinity).

nDCG , short for Normalized Discounted Cumulative Gain, is a more intricate retrieval measure for graded relevance. Its basis is *Cumulative Gain* ($CG@p$), which is defined as the sum of relevance gradings up to item p:

$$CG@p = \sum_{i=1}^{p} rel_i \tag{2.5}$$

Since documents higher up in the ranking are generally more important than those lower down, one can use *Discounted Cumulative Gain* ($DCG@p$).

$$DCG@p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} \tag{2.6}$$

or alternately, a more commonly used version which places higher weight on relevant documents:

$$DCG@p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.7}$$

Since DCG can be hard to interpret across different queries and corpuses, a way to normalize is useful. To do this, one first needs to calculate the best possible result, *Ideal Discounted Cumulative Gain* ($IDCG@p$). This uses the sorted list of documents by relevance up to position p, REL_p:

$$IDCG@p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.8}$$

Which leads into the final measure of nDCG:

$$nDCG@p = \frac{DCG@p}{IDCG@p} \tag{2.9}$$

2.2.2 TF-IDF

TF-IDF (Sparck Jones, 1972) is a classical bag-of-words method for weighting of terms. It consists of two parts:

Term Frequency (TF) is supposed to weight words based on how often they occur in a document. There are different ways of doing this, including binary (1 if the term is in the document and 0 otherwise), the raw number of occurrences, relative frequency in the document, or more advanced formulas like the log of the raw count.

Inverse Document Frequency (IDF) is supposed to discount terms that appear in many documents. For example, since words like 'the' will occur in almost every English document, its signal should be negligible in a bag-of-words setting. As with TF, there are multiple ways of achieving this, most of which including some variation of $\log \frac{N}{n_t}$, with N being the total number of documents, and n_t the number of documents in which the term appears.

For retrieval, the TF and IDF are multiplied together and summed for each query term to produce a final relevance score. Additional latency reduction is gained from storing these values in an *inverted index*, where each term has a list of document it appears in, making this a very fast and easy retrieval mechanism.

2.2.3 BM25

BM25, short for (Best Matching 25) is another classical term weighting method based on TF-IDF. While several variations exist (Kamphuis et al., 2020), the original formula proposed by Robertson et al. (1995) was:

$$BM25(q,d) = \sum_{t \in q} \log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{tf_{td}}{k_1 \cdot \left(1 - b + b \cdot \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} \tag{2.10}$$

where d is the document, q is the query, t is the term, N is the number of documents in the corpus, df_t is the number of documents containing t, tf_{td} is the frequency of t in the document, L_d and L_{avg} are the number of terms in the current document and average document respectively, and k_1 and b are tunable parameters.

Although BM25 was originally introduced in the 90s, it has stood the test of time due to its high speed, accuracy, and simplicity, still being used as baselines for ranking algorithms to beat and as initial retrieval into more advanced and computationally intensive re-rankers.

BM25 is also commonly used in conjunction with a relevance model, particularly RM3. RM3 improves search results by doing an initial BM25 ranking, analyzing top documents to find terms which are estimated to be likely to occur in relevant documents, and doing a second round of retrieval using an expanded query including these terms.

2.2.4 Embeddings

Although bag-of-words methods seem to be effective and have low latency due to methods like inverted indexing, some critical pieces are missing. The ability to detect similar words is one of them. If a query contains 'dog', and a relevant document only contains 'chihuahua', then a bag-of-words method will discard that document as irrelevant for not containing any query terms. This is where embeddings come in.

Embeddings are vector representations of an item that contain some relevant information. In NLP, these are often words where the embedding is some fixed-size dense vector that contains useful knowledge about the meaning of the word. This means that words like 'dog' and 'chihuahua' will have similar embeddings (typically measured by cosine or euclidean distance), while a word like 'finances' will be far away. One can also extend the idea to entire documents, which has proven to be quite difficult. Here is a rundown of a few prominent methods:

Latent Semantic Indexing (LSI, Deerwester et al. 1990) is a method for detecting "topics" in a collection of documents by performing singular value decomposition (matrix factorization), or SVD, on a term-document matrix to produce fixed-size vector representation for both terms and documents. For a more nuanced signal, it is common to use TF-IDF values for the term-document matrix. By manipulating these matrices one can compare both documents and terms.

Word2Vec (Le and Mikolov, 2014) is a framework for creating word vectors. It uses a simple two-layer neural network, which either predicts surrounding words based on a single word (skip-gram), or predicts a single word based on surrounding words (continuous bag-of-words, or CBOW). This enables representations to be created which contain some information about the context in which words are generally used. The assumption is that words that occur in similar contexts have similar meanings.

Doc2Vec (Mikolov et al., 2013) is an extension to Word2Vec, which introduces an additional vector for each paragraph or document to the training. This can either be added as additional information along with the word vectors to inform the prediction of a single next word (distributed memory) or by using the paragraph vector alone to predict all the words in the paragraph.

GloVe (Pennington et al., 2014) is short for Global Vectors for Word Representation, a more mathematically rigorous approach to the word vector task. It trains a model by optimizing such that the dot product of two word-vectors gives the log of their probability of co-occurrence. This achieved better results compared to both Word2Vec and SVD on a variety of tasks.

Additionally, for very large embedding collections, exact or approximate nearest-neighbor or inner-product search methods can be used to find the best matches for any embedding efficiently. Similarity is typically measured using either cosine distance/similarity $(-1, 1)$, euclidean distance $(0, \infty)$, or dot/inner product $(-\infty, \infty)$. Specifically:

$$
\begin{aligned}
DotProduct(A, B) &= \sum_{i=1}^{d} A_i B_i \\
EuclideanDist(A, B) &= \sqrt{\sum_{i=1}^{d} (A_i - B_i)^2} \\
CosineSim(A, B) &= \cos\theta = \frac{\sum_{i=1}^{d}}{\sqrt{\sum_{i=1}^{d} A_i^2}\sqrt{\sum_{i=1}^{d} B_i^2}} \\
CosineDist(A, B) &= 1 - CosineSim(A, B)
\end{aligned}
\tag{2.11}
$$

Where A and B are the two vectors (embeddings) being compared and d is their shared dimensionality. θ is the angle between the vectors. Figure 2.1 shows a visualization of these distances.

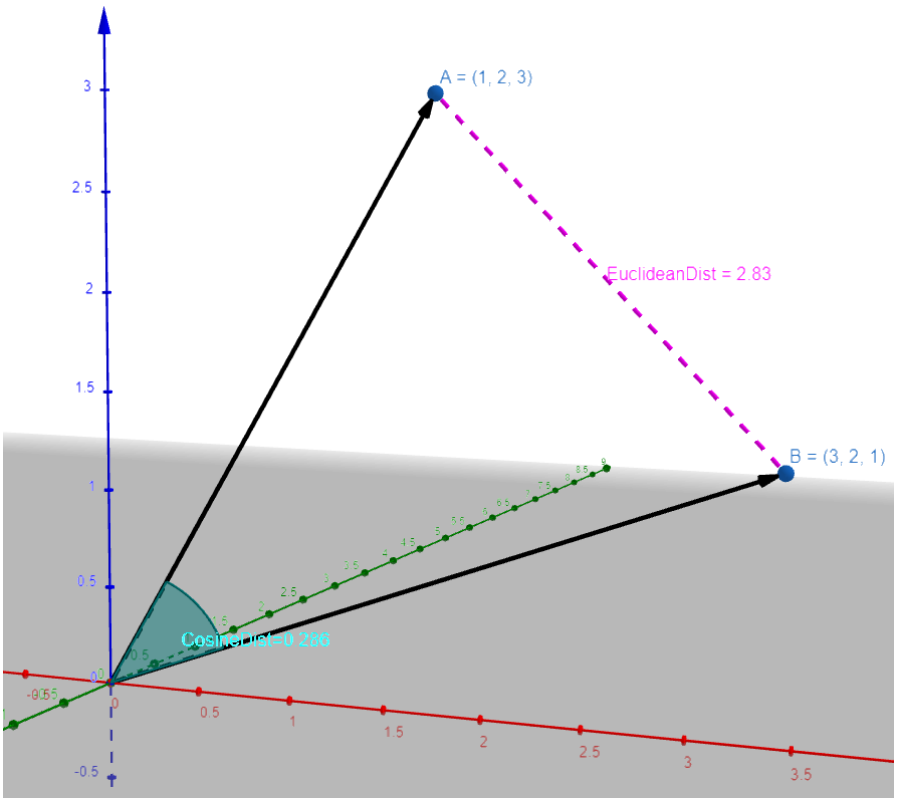


Figure 2.1: A visualization of distances between two points. The dot product has no visual representation, but would be 10. Figure created with GeoGebra.

2.3 Machine Learning

In recent years, there has been a resurgence of machine learning, and particularly deep learning, due to increased computational power and more research. With this comes new opportunities for many fields, including document retrieval. With more and more powerful text-understanding models, the potential is hard to overstate. This section serves as a build-up and introduction to the techniques used for retrieval in modern research.

2.3.1 Perceptron

The most basic type of neural network is a perceptron, also known as linear, dense, or fully connected. If the input is a vector $x_{n\times 1}$, then the output $y_{m\times 1}$ is simply a linear combination of these values, using a weight matrix $W_{m\times n}$, as well as a bias b_n. The subscript here denotes their dimensionality.

$$y_{m\times 1} = W_{m\times n} \cdot x_{n\times 1} + b_{n\times 1} \tag{2.12}$$

These models can be stacked several times, allowing for more complex operations. The model learns by adjusting W and b by taking small steps in the direction of the model's gradient as determined by the loss function (subsection 2.3.2) with backpropagation through all the layers.

In addition, it is common to use activation functions to modify the outputs. Some of the most common ones include:

ReLU is short for Rectified Linear Unit, which turns any negative values into zeros. Commonly used in the hidden layers of the network due to proven performance improvements.

$$\text{ReLU}(x) = \max(0, x) \tag{2.13}$$

Sigmoid is a function that restricts the output to a value between 0 and 1, commonly used for binary classification.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

Softmax is a generalization of sigmoid for any number of classes which restricts the sum of the entire result to 1. This is often used to get probabilities for each class.

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2.15}$$

2.3.2 Loss

In machine learning, the loss is a measure of the performance of a model. Typically, this is some real value that one seeks to minimize. For instance, a simple loss function could be the mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|\hat{y} - y| \tag{2.16}$$

With $\hat{y}$ being the model prediction and y the correct value.

A good loss function is essential for machine learning, as it will determine what the model will be good at. For the field of neural IR, there are several popular loss functions:

Cross-entropy is typically used for classification, which in the case of ranking comes down to picking the relevant document out of a selection.

$$CE = -\sum_{x \in X} p(x) \log q(x) \tag{2.17}$$

Where X is all the class outputs, $p(x)$ is the probability of the class being correct (typically just 1 or 0), and $q(x)$ is the model's predicted probability that the class is correct. For one-hot labels, this is simply the negated log probability of the correct answer.

Ranking loss is used to change embeddings based on some distance metric. Generally, one can give a model some input to compare with, called the anchor. The anchor is then compared with positive (somehow related) samples and negative (not related) samples. The model is tuned to place positive samples closer and negative samples further away in the embedding space. This is commonly used for facial recognition, where doing extensive training for each face is unfeasible. For the case of one anchor, one positive and one negative, this is commonly called triplet loss:

$$TL(a, p, n) = \max(0, m + \text{dist}(a, p) - \text{dist}(a, n)) \tag{2.18}$$

Where m is a margin to separate the positives and negatives by, and the dist function is typically cosine or euclidean distance. One problem to tackle with ranking loss is *hard negative mining* - finding samples that are hard for the model to guess since separating between random samples and relevant samples is fairly easy in most cases after little training.

2.3.3 Convolutions

Convolutional neural networks (CNNs) are a type of network commonly used in image processing. It bases itself on the assumption that local interactions are more important than distant interactions, meaning that connections between every neuron is not necessary. Instead, CNNs use a mathematical operation called convolutions, sliding a so-called

kernel (a weighting of values for each region) across the input. Typically, there are many kernels for each layer, each producing a different view of the input.

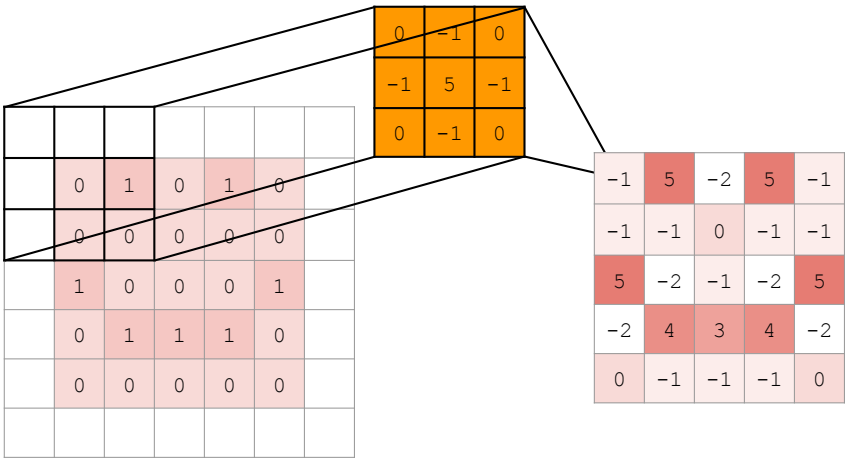


Figure 2.2: A 2D convolution. Every element in the current selection is multiplied with its corresponding value in the kernel, then all these values are added up. This specific kernel is designed to sharpen the image, but in neural networks it is free to do whichever operation is most useful to the end task. Note that the image is padded so its size remains the same.

2.3.4 Recurrent Networks

A recurrent neural network (RNN) is a type of neural network designed to work on sequential data. It does this by feeding the result of previous computations into itself along with the next data point in the sequence. There are multiple different kinds of RNN, like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit). RNNs are commonly used for time series data and were for a long time the best choice for Natural Language Processing (NLP) tasks, but have recently been surpassed by Transformers.

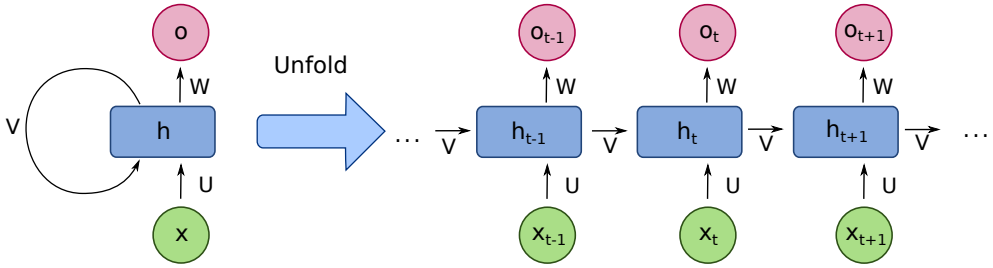


Figure 2.3: A recurrent neural network. U, V, and W are different operations, and x, h, and o are the input, hidden state, and output sequences respectively. Taken from `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`, licensed under the Creative Commons Attribution-Share Alike 4.0 International license.

2.3.5 Attention

A significant drawback with recurrent networks is their poor long-term recollection. While this is fine for tasks where the importance of previous inputs decreases over time, for many language tasks that rely on these long-range dependencies, such as translation, this gives poor performance.

To mitigate this, a mechanism called attention was proposed. Instead of simply feeding the hidden state from the last recurrent output at every step, the hidden state is run through an attention mechanism, which "asks" every element in the input sequence for relevance. This means that the recurrent part of the network is free to remember more high-level information, rather than needing to compress all the words that have appeared so far into a single vector.

A common way to do attention is the so-called scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.19}$$

Where Q, K, V are known as the query, key, and value vectors, respectively, and d_k is the dimensionality of the vectors. In essence, queries are supposed to represent what information it wishes to receive, the key tells us what information the value contains, and the value is the information content itself. In the case of RNNs, the queries typically come from the outputs of the decoder, while keys and values come from hidden states of the input sequence.

2.3.6 Transformers

While attention mechanisms helped RNN performance by mitigating long-term dependencies, a big drawback was still the need to feed RNN outputs into themselves recursively. By doing away with the RNN part completely and using only attention, Vaswani et al. (2017) were able to get state-of-the-art performance on various NLP tasks, owing mostly to the speedup acquired by increased parallelization as well as the aforementioned attention mechanism reducing path lengths for the embeddings. In addition, since attention contains no information about the order of the sequence, they introduce a positional encoding which is added to the initial word embedding.

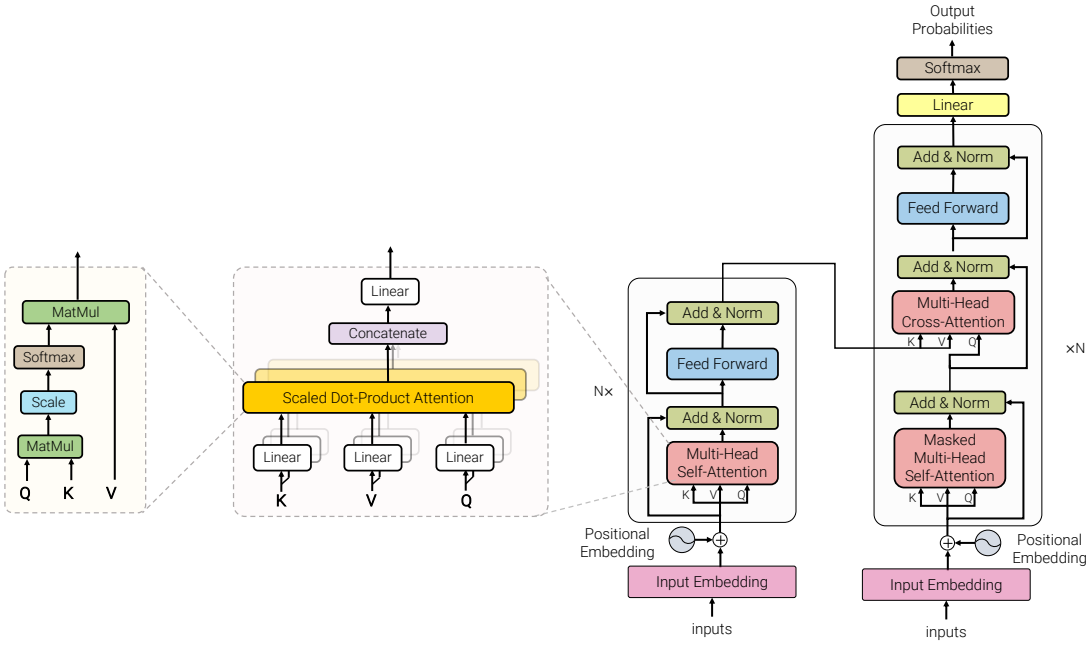


Figure 2.4: The Transformer architecture. Figure from Tay et al. (2020), with permission from Mostafa Dehghani, with modifications.

The architecture as presented by Vaswani et al. consists of an encoder and a decoder. The encoder is meant to "read" the text and produce contextual embeddings for each token, while the decoder "asks" the embeddings produced by the encoder about relevant information. To generate text, the decoder reads the current output embeddings - shifted right and masked to prevent reading ahead - and produces queries which it can use on the keys and values of the encoder's embeddings. The positional encoding is a special vector that uses sin and cos to contain a binary-like representation of where in the sequence the token is, which is added to the token embedding.

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.20}$$

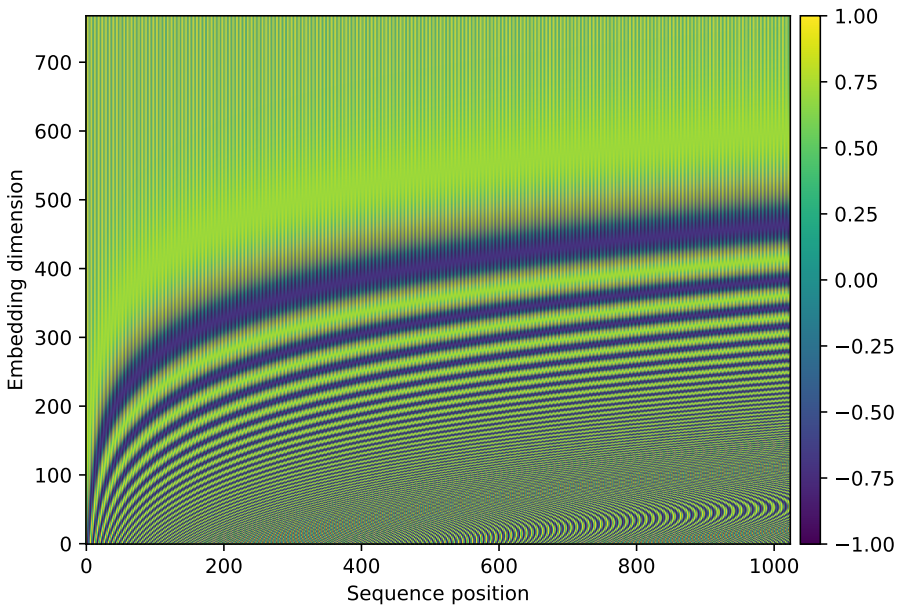


Figure 2.5: A plot of the positional encoding up to 1024 tokens, using a typical Transformer embedding size of 768. Figure created with Matplotlib.

As the name implies, multi-head attention uses multiple stacked "heads" of attention with separate weights. In the Transformer, each token's embedding is split into h different keys, values, and queries via dense layers, each of size $\frac{d}{h}$. These are then fed into the attention heads, concatenated to keep the same dimensionality, and fed into a linear layer to produce a new contextual representation of the token.

Reducing complexity

A big obstacle with Transformers is their $O(n^2)$ space and memory complexity for the sequence length. Reducing this is a big area of research, and many methods have been proposed with lower complexity, as shown in Figure 2.6. These are known as *efficient Transformers*. Efforts have been made to compare them directly (Tay et al., 2021), but there is no conclusive best solution as of yet. This is particularly relevant for document retrieval, as most documents are longer than the typical max length of around 256 tokens used in many solutions.

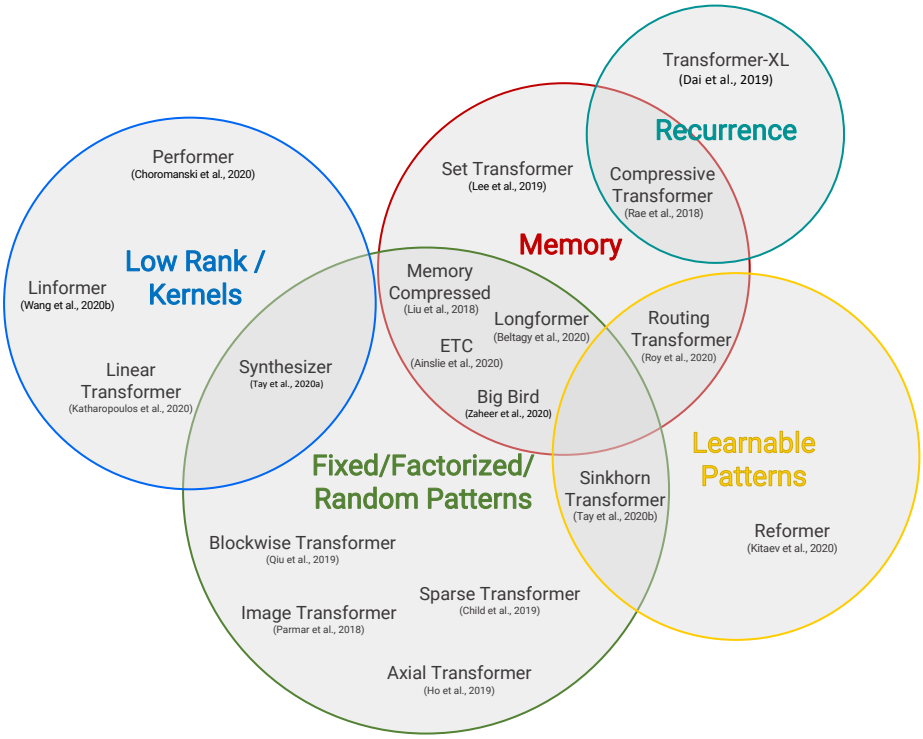


Figure 2.6: Taxonomy of efficient Transformer architectures. Taken from Tay et al. (2020), with permission from Mostafa Dehghani.

BERT

BERT (Bidirectional Encoder Representations from Transformers, Devlin et al. 2019) is a very popular Transformer architecture. Instead of training the language model to predict next words, they introduced two new unsupervised training methods in the pre-training, removing the decoder part of the Transformer entirely:

- Masked LM, in which a random token embedding is replaced, and the model is supposed to predict the missing word. They replace around 15% of tokens, of which 80% are replaced with a [MASK] token, 10% with a random token, and 10% remain unchanged but still need to be predicted.
- Next sentence prediction, in which two sentences are used as input, separated by a special [SEP] character. The model uses a [CLS] token, which captures information across the entire input sequence to predict whether or not the two sentences follow each other in the text.

These simple methods have proven to be a massive benefit, enabling BERT to be trained in a self-supervised manner, learning the "basics" of language, and later fine-tuned to do specific tasks such as classification, translation or question answering. Many different

pre-trained models are available, including multilingual BERT (mBERT), trained on many languages. In addition, BERT spawned a plethora of Transformer architectures trained in similar manners. For the purposes of this Thesis, the most important are:

RoBERTa (Robustly optimized BERT approach, Liu et al. 2019), which is a refinement of the BERT pre-training approach that significantly improves performance and consistency. Specifically, they use a different tokenizer which is less prone to OOV tokens, gradient accumulation to simulate larger batch sizes and thus enabling training on longer sequences, training on more data, changing the word masks dynamically during training, as well as removing the next sentence prediction objective.

BigBird (Zaheer et al., 2020) is a modified Transformer for long sequences which instead of the normal attention uses different variations and combinations of: *random attention*, where each token attends to a set number of random tokens in the sequence; *window attention*, where each token attends to the local context around itself; and *global attention*, where some tokens attends to all the tokens in the sequence.

Longformer (Beltagy et al., 2020) is similar to BigBird, but without random attention, and with *dilated sliding window attention*, which attends to every n-th word in an area around the token. Following the findings of RoBERTa, it also foregoes the NSP task.

3 Related Work

This chapter covers past work that is relevant to neural IR. It is split into three parts. The first part provides an overview of typical approaches to neural retrieval models in recent years. The second part covers the specific research area of zero-shot retrieval. Finally, the third part describes some commonly used datasets used for retrieval.

3.1 Traditional neural retrieval models

Within neural IR, there are three main ways of producing relevance scores, as outlined by Yates et al. (2021): **representation**-, **interaction**- and **Transformer**-based. In this section, a few of the most relevant methods within each area and their techniques are presented.

3.1.1 Representation

Representation-based models produce a single vector to represent the entire query or document, as opposed to their constituent term vectors. These can then be compared directly to produce a relevance score. Representation models are generally faster than the other types at retrieval time, and traditional IR models like TF-IDF (subsection 2.2.2) and BM25 (subsection 2.2.3) can be thought of as representation models using sparse vectors, and LSI and Doc2Vec (subsection 2.2.4) with dense. Due to the improved speed, representation-based models can be used for initial retrieval instead of only re-ranking top documents, which is typical for interaction and Transformer-based models.

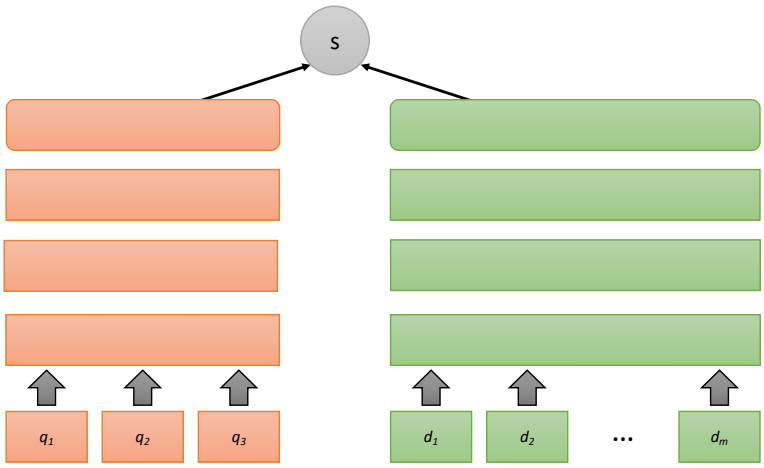


Figure 3.1: A representation model. Taken from Yates et al. (2021), with permission from Andrew Yates.

Adjacent tasks

Although the main concern for this Thesis is document ranking, it is useful to see the techniques used in related areas.

Sentence similarity is one of the pre-training tasks used for models like BERT and RoBERTa (section 2.3.6), for which they set state-of-the-art results. A clear bottleneck in using these models is that both sentences need to be input into the model every time, slowing down inference. To improve upon this, Sentence-BERT (Reimers and Gurevych, 2019) is instead trained to produce an embedding for each sentence, which can be compared much faster. In their paper, they measure the time to find the two most similar sentences in a corpus, reducing the inference time from 65 hours to 5 seconds, with similar accuracy. In addition, their findings show that BERT's [CLS] embedding output performs poorly with common similarity metrics like dot product or cosine similarity without specific training.

Question answering is another related task, where the query is replaced with a natural language question, and the model is supposed to find documents or passages in a knowledge corpus. Models like Dense Passage Retriever (DPR, Karpukhin et al. 2020) show that encoding the question and passage with two separate BERT models, and comparing using dot product can work well (beating BM25) with as little as 1000 training examples. A recent study by Ma et al. (2021c) replicating the results from the DPR paper has shown that BM25 is better than the original paper suggests, and using them in conjunction boosts the results even further. Retrieval-Augmented Language Model Pre-Training (REALM, Guu et al. 2020) uses an integrated retrieval system to improve language model results, concatenating the text with an automatically retrieved passage through latent search, aiding the model through additional concrete information, all done

in an unsupervised fashion with end-to-end training.

DSSM

One of the earliest application of deep neural nets in neural IR was DSSM, or Deep Semantic Similarity Model (Huang et al., 2013). To reduce input dimensionality, they used a bag-of-n-grams (as opposed to bag-of-words) array as input into a simple fully connected network, producing document vectors of size 128, which could then be compared using cosine similarity to produce a relevance score. Instead of basing themselves on text data alone, like earlier approaches, they used click-through data to train their model to recognize relevance as determined by humans. This was later built upon by methods like C-DSSM (Shen et al., 2014a) or CLSM (Shen et al., 2014b), which are quite similar with small adjustments to the architecture.

ANCE

A problem with representation based systems, as identified by Xiong et al. (2021), is that negative samples (documents) used for training are often uninformative later on in the training process, due to being too easy to identify as negatives. For example, DPR (section 3.1.1) uses negatives sampled from random, BM25 or positive passages from other questions to train the model. To make negatives harder to identify, ANCE (Approximate nearest neighbor Negative Contrastive Learning) was proposed, which samples negatives using ANN (subsection 2.2.4) to find documents that are close to the query (rated more relevant by the model), providing a more difficult challenge for the model. The ANN index is refreshed regularly, such that samples are not based on outdated embeddings. They use a fairly simple BERT-Siamese model, and show that using their ANN sampled negatives boosts performance compared to other sampling methods, including that of DPR.

3.1.2 Interaction

Interaction based models use interactions between query terms and document terms as basis. For a query of size q, and document of size d, a $q \times d$ matrix is produced, usually by computing similarities between term vectors. This matrix is then further processed to produce a final relevance score. A key advantage of this type of model is that any type of term vector can be used, potentially enabling more unsupervised learning.

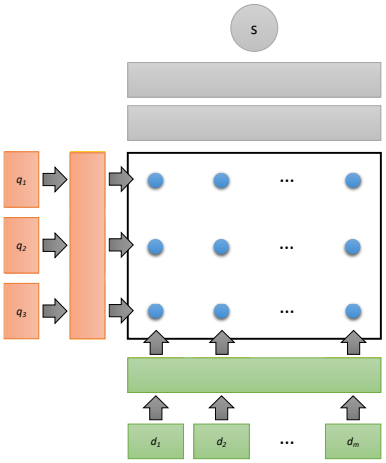


Figure 3.2: An interaction model. Taken from Yates et al. (2021), with permission from Andrew Yates.

ARC-II

The interaction based approach was originally proposed by Hu et al. (2014), presenting two different convolutional models for sentence matching: ARC-I, using a traditional representation based approach, and ARC-II, using an interaction based approach. Instead of simply calculating similarity between individual words in the sentences, ARC-II uses a sliding window, concatenating the word vectors from the windows of each sentence, and using 1D convolutions on these concatenations to produce values in a 2D matrix, which is then run through multiple layers of 2D pooling and convolution. For their objective function, they used a ranking-based triplet loss to train the network such that negative samples (non-similar sentences) are ranked as less similar than positive samples (similar sentences).

Building upon this idea, and taking inspiration from image recognition, Pang et al. (2016) used word-word interactions (i.e. cosine similarity or dot product) on two sentences to create an 'image' of their interactions. The model, called MatchPyramid, then uses traditional image processing techniques like pooling and convolution to process the image to produce a matching score for the two sentences.

DRMM

Building further on the ideas that ARC-II and MatchPyramid introduced, Guo et al. (2016) presented a different way of handling the matching, with more focus on the ad-hoc retrieval task. For their Deep Relevance Matching Model (DRMM), instead of using convolutions across the entire similarity matrix, they produce a histogram of document similarity values for each query term using pre-trained Word2Vec vectors. These histograms are used as input to a model individually, producing a similarity score

for each query term. The scores are aggregated using a term gating network to place more importance on certain terms. Different methods for producing the histograms are compared, including count-based, normalized and log-count. The term gating network can either be trained or use IDF to weight the query terms. Their best results are obtained using log-counts for histograms, and IDF for term gating. A potential disadvantage of using histograms specifically is that they are non-differentiable, meaning end-to-end training of the model is made difficult. This is addressed by later models.

K-NRM

To counter the problem of histograms being non-differentiable, Xiong et al. (2017) introduced K-NRM (Kernel based Neural Ranking Model). The main innovation they made was replacing the histograms with a special kernel called the RBF kernel. The formula for the RBF kernel is:

$$K_k(M_i) = \sum_j \exp\left(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}\right)$$

where k represents the index of the kernel, and μ_k and σ_k are adjustable weights, while M_i is the row of the i-th query term from the similarity matrix. The kernel enables the model to 'count' the number of occurences within in a certain similarity region, akin to the histogram of DRMM (section 3.1.2), but in a differentiable way. These kernels are aggregated, and a sum of logs for each kernel is computed, then weighted to produce the matching score.

PACRR

PACRR (Position-Aware Convolutional-Recurrent Relevance Matching, Hui et al. 2017) is another interaction model, whose main innovation was to use a recurrent layer at the end of the processing, and using varying kernel sizes to capture interactions. This enables the network to account for the position of query terms when producing the final relevance score. They also introduced Co-PACRR (Hui et al., 2018), which uses a sliding window of the query over the document to add additional context information to the network.

3.1.3 Transformer-based

Due to the way Transformers are constructed, a new way of comparing queries and document was made possible. By inputting the query and document, separated by a [SEP] token, the Transformer alone can produce an advanced relevance score informed by complex contextual interactions between every term in each.

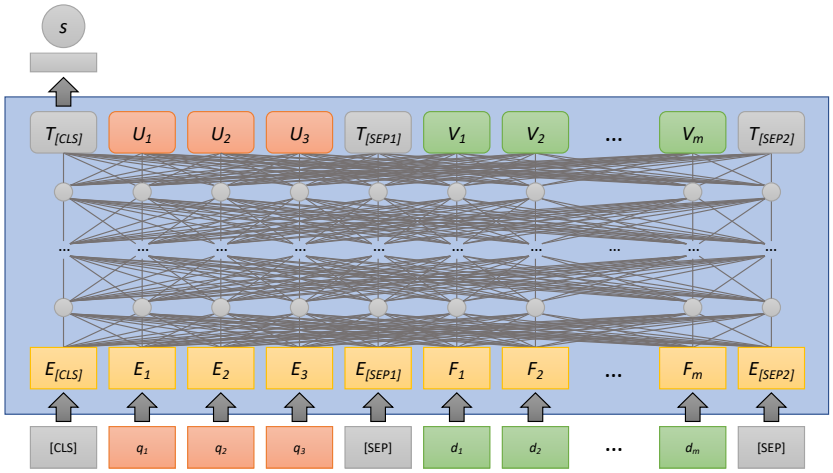


Figure 3.3: A Transformer-based re-ranking model. Taken from Yates et al. (2021), with permission from Andrew Yates.

Re-ranking with BERT

The seminal paper on Transformer-based retrieval models came from Nogueira and Cho (2019). They built a Transformer (BERT) based passage re-ranking architecture, basing itself on BERT's ability to identify question answer pairs, as shown in the original BERT paper. They analyzed performance on MS MARCO as a function of the number of training examples (question-passage pairs), and showed that the performance improved up to around 10 million samples.

Further developments came from Yang et al. (2019b), later built upon by Birch (Akkalyoncu Yilmaz et al., 2019), which extended the passage re-ranking concept to ad-hoc document retrieval. This was done by simply performing the relevance judgement on each sentence, then aggregating the results by weighting scores for a set number of top sentences, along with the ranking score from initial retrieval as such:

$$s_{final} = a \cdot s_{initial} + (1 - a) \cdot \sum_{i=1}^{n} w_i \cdot s_i \tag{3.1}$$

Where a is a tunable parameters, s_i is the score of the i-th highest scoring sentence and w_i its associated weighting, also tunable.

In an effort to further improve ad-hoc ranking performance, Nogueira et al. (2019) introduced a new stage to the re-ranking procedure: duoBERT (while also retroactively dubbing the regular Transformer re-ranker monoBERT). Their architecture uses BM25 for initial retrieval, passing each document along with the query into monoBERT for a deeper ranking score, then finally passing each pair of two documents (still along with

the query) into duoBERT to determine which one is most relevant, sorting the final list produced by monoBERT.

ColBERT

Due to the high computational cost of running both query and document in large Transformer models, a new method dubbed ColBERT was proposed by Khattab and Zaharia (2020). ColBERT's innovation comes mainly from replacing early interactions - sending both query and document into the model - with late interactions, where the query and document are processed by a Transformer model separately, then compared using a maximum similarity (MaxSim) operation for each query term across the document, summed to produce a final relevance score. Since the document embeddings can be pre-computed and stored, latency is substantially reduced (by around an order of magnitude in their testing), and their results show that the accuracy is similar to using early interaction.

CEDR

Taking inspiration from both representation models and Transformer models, CEDR (MacAvaney et al., 2019b) used a Transformer (BERT) to produce contextual embeddings for both query and document terms, and compared them using different interaction models like DRMM, K-NRM and PACRR. They introduced a way to aggregate outputs of multiple chunks of text by averaging their relevance score, as well as averaging [CLS] embeddings from the BERT output of the text chunks to use as additional input when calculating the final relevance score. They studied the effect of limiting the number of attention layers on speed and accuracy, finding that using more than five layers offered very little additional accuracy, but with a big slowdown. Their best results were obtained using BERT embeddings and K-NRM with [CLS] vector incorporation.

PARADE

Most Transformer models do their passage aggregation by traditional methods such as averaging [CLS] embeddings, or evaluating each passage for relevance individually and taking the max or mean. Instead of this, PARADE (Passage Representation Aggregation for Document Reranking, Li et al. 2020) places a Transformer or attention layer at the end, which looks at the [CLS] token embedding of all the passages in order to produce the final relevance score. They tested both using BERT and another language model called ELECTRA (Clark et al., 2020) as their passage models.

MonoT5

As seen with PARADE, the choice of pre-trained model can have significant impact on the performance of the final architecture. Whenever newer models show promise on other

NLP tasks, one can expect that they will be tried on the retrieval/re-ranking task. This is exactly what Nogueira et al. (2020) did, replacing the BERT model of monoBERT with a model called T5 (Raffel et al., 2019), designed for sequence-to-sequence outputs. T5 differs from other models in that during its pre-training, it is given "commands" of what to do. This can look something like 'translate English to German: That is good.', for which the model would generate 'Das ist gut.'. In fact, the T5 model is also trained on sentence similarity, paraphrasing and question answering during pre-training, likely providing a boost for retrieval/re-ranking tasks. In monoT5's case, they feed the model with the following template: 'Query: q Document: d Relevant: ', for which the model will outputs some probability of generating tokens 'true' and 'false', which are used as the ranking score. Interestingly, they evaluated their model in a zero-shot transfer setting, training on MS MARCO and evaluating on Robust04, obtaining very good results. Like monoBERT to duoBERT, monoT5 has also been used in a pairwise ranking fashion (Zhang et al., 2020), aptly named duoT5.

3.1.4 Criticism

In spite of seemingly good results, many neural ranking systems have come under scrutiny for using weak baselines to support their results. Yang et al. (2019a) tested several neural models and showed that most neural re-rankers do not significantly improve results from a strong BM25+RM3 initial retriever. In fact, only DRMM showed statistically significant improvement.

More recent results by Yates et al. (2020) have confirmed these results, although they also found that Birch (section 3.1.3) and CEDR (section 3.1.3) - both Transformer-based models which were not tested by Yang et al. - show significant improvements. They theorize that the small size of the Robust04 dataset (subsection 3.3.2) - only 250 queries - could be preventing models from generalizing. Birch and CEDR, by virtue of using Transformers, could be benefitting from their general NLP pre-training, only requiring a small amount of fine-tuning on Robust04.

3.2 Zero-shot

In terms of the zero-shot retrieval setting in particular, there is limited research.

MacAvaney et al. (2019a) tested the performance of different modern methods on Arabic, Mandarin and Spanish datasets, showing that multilingual BERT performed well on the zero-shot re-ranking task (top 100 documents from BM25). Providing a few labeled samples from the target language seemed to slightly improve the results overall, particularly on the Spanish dataset. They also tested PACRR and KNRM using mBERT's word embedding outputs with less success, performing substantially worse than the BM25 baseline in some cases.

Some solutions have been proposed for zero-shot between English datasets by generating

queries to train on (Ma et al., 2021a; Liang et al., 2020). This requires a language specific model trained to generate queries, which is not available for most languages.

As mentioned in section 3.1.3, monoT5 exhibits zero-shot capabilities when trained on MS MARCO and transferred to other English datasets. Birch (section 3.1.3) showed similar results with the standard BERT model, training on tweets and evaluating on newswire articles.

REALM (section 3.1.1) has proven potential on zero-shot tasks, as it uses unsupervised information to train a retrieval model, however adapting the model to document retrieval instead of question answering could prove difficult. Other unsupervised pre-training approaches designed to improve retrieval exist, such as PROP (Ma et al., 2021b), or ICT, BFS, WLP described by Chang et al. (2020), though none seem to have been applied to other languages as of yet.

Litschko et al. (2021) performed a systematic evaluation of multilingual encoders, e.g. multilingual BERT and Facebook's XLM (Lample and Conneau, 2019) as applied to cross-language information retrieval (CLIR). They show that these encoders fail to outperform systems based on cross-language word embeddings for documents, although for sentence-level retrieval the results are more promising. CLIR is a slightly different task in that it attempts to find documents in different languages than the query, but the results are notable nonetheless.

3.3 Datasets

This section will showcase the most used datasets from the research field and datasets relevant for this project. All of the datasets have advantages and disadvantages because they differ widely in size, how they are made and the language of the documents and queries. Choosing a correct dataset for the Thesis relies on considering those factors in relation to the experiments conducted. It does not exist many freely available non-English datasets for ad-hoc retrieval. Most of them are either behind a pay wall or they are to small to be useful.

3.3.1 MS MARCO

MS MARCO (short for Microsoft MAchine Reading COmprehension) is a collection of datasets introduced by Nguyen et al. (2016) in an effort to create a common benchmark for retrieval and question answering tasks. It consists of anonymized questions from Bing's search logs, split into multiple tasks. Namely, they have tasks for document and passage ranking, both full ranking and re-ranking. There are 1,010,916 queries in the form of questions from anonymized Bing users. The questions were put through filters and human editing to only leave answerable questions in the dataset. For each question there exists on average 10 relevant passages and every question has zero or more answers. The passages are collected from Bing search engine top results and each

passage is labeled relevant or not relevant by humans. Each passage is extracted from a document, those documents are the document collection where documents with relevant passages are considered relevant. In total there is 8.8 million passages and 3.2 million documents.

3.3.2 TREC 2004 Robust track

TREC 2004 Robust is a dataset created for the Text Retrieval Conferences (TRECs) in 2004, organized by the U.S. National Institute for Standards and Technology(NIST). Robust 04 is one most researched datasets for ad-hoc retrieval, due being over a decade old with and having high quality labels. Yates et al. (2021) The corpus of documents comes from TREC disk 4 & 5 excluding the congressional documents. Disk 4 & 5 consists of news articles from the Financial Times Limited, the Foreign Broadcast Information Service, and the Los Angeles Times with 528K documents. There are 249 topics and each topic have relevant documents, these topics are used as queries. Compared to MS MARCOs 1 million queries, 249 topics is not that much. What Robust 04 can offer though is a lot of good relevance feedback. When the dataset was labeled, all documents in the corpus was considired. This is reduces false negatives in the training dataset.

3.3.3 TREC Spanish

TREC Spanish was created by the Linguistic Data Consortium and consists of Spanish newswire data from Agence France Presse and El Norte. The corpus, same as Robust 04 was used for TREC in their Spanish task. The topics for this corpus is split in two, one for TREC 3 with 25 topics and topic descriptions and TREC 4 with only 25 topic descriptions. In this dataset there is only 58K documents, not a lot compared to the others.

4 Architecture

As there is no specific new architecture being introduced, this section will instead reason about why different approaches are good/bad for the task at hand. It also covers tools and libraries used to implement the solutions.

4.1 Strengths and weaknesses of different models

When choosing which model to use, it is important to consider many facets of performance. Latency, effectiveness, ease of use, ease of training, as well as knowledge and language transfer all have to be considered and tested in a final retrieval system. This section will outline the advantages and disadvantages of choosing each particular type of model.

4.1.1 Representation

In an ideal world, representation models are the obvious choice due to their simplicity and speed. Unfortunately, we do not live in an ideal world, and thus representation models still face a number of challenges, particularly in the zero-/few-shot retrieval setting.

Non-specific training

Training from the ground up is undesirable in the zero-shot setting, which means some sort of pre-training or unsupervised solution is required. Multilingual word vectors are an alternative, but requires a robust aggregation method to produce document-level embeddings. Fortunately, pre-trained multilingual Transformers like mBERT exist, meaning a solution similar to DPR (section 3.1.1) - simply using the [CLS] embedding output of the BERT model as the dense representation - could be viable. This is not ideal, since it has been shown that BERT's [CLS] embedding has poor performance "out of the box" (Yates et al., 2021, p. 114). The DPR authors do however show that it does not need many samples to start performing.

Iterative improvement

Since all the document vectors need to be pre-computed and stored to be effective, updating the model as new data comes through can be tedious. ANCE (section 3.1.1) uses an asynchronous update to keep the index refreshed during training, always retrieving

with an outdated document vector. Re-rankers have the benefit that further training of the model is simple, since updating vectors for every document in the corpus is not needed.

Performance limits

It is unclear how far representation models can take the retrieval performance. Most of the best results are currently obtained using multi-stage rankers, spending more time and computing power on evaluating the most relevant documents. This means that even if a good representation model solution is found, it might not be enough to beat a simple BM25+Reranker architecture. It could quite simply be that any reasonable vector dimensionality is insufficient to convey the entire information content of the document. Initial retrieval is still worth improving as it would boost the overall performance of the system, and is a good target for dense retrieval systems since their latency is comparable to classical retrieval methods.

4.1.2 Interaction

For transfer in the zero-shot setting, interaction models are seemingly promising due to their simplicity and ability to detect nuanced similarity interactions. However in practice they also have some significant drawbacks. The following text will outline these.

Pre-trained potential

Since the neural net part of the system can be detached from the term embeddings and only look at their similarities, training term embeddings in an unsupervised manner on a relevant corpus and using a pre-trained interaction model to get the relevance score could be viable with minimal training. The key question here is whether or not languages are different enough in their structure that it makes a significant impact on the retrieval performance. In theory, English and Norwegian are both Germanic languages, and should therefore share much of the same structure.

Performance limits

The fact that term embeddings are detached can also be a detriment to the performance. A simple term-to-term similarity matrix may not be adequate to capture the relevance of a document, and more context and language understanding may be required. As mentioned in subsection 3.1.4, interaction based methods were shown to have weaker results against BM25 than originally presented. This isn't to say they don't work, but they might require more (pre-)training before transfer.

4.1.3 Transformer-based

With such promising results in most other NLP fields, it is tempting to start using Transformers as soon as possible for retrieval as well. However one has to be careful to avoid a few pitfalls for retrieval applications.

Direct transfer limitations

Despite the existence of pre-trained models on general cross-language corpora, these models are not designed with query-document input in mind, and therefore need to be trained further. Since these models are trained end-to-end, it seems that training them on the right language and domain is a requirement for optimal results, and that zero-shot transfer across language and domain could prove difficult.

Computational cost

While accuracy is high, inference time is generally much higher on Transformers than for simpler neural net alternatives, or traditional IR methods. In part due to the large inference time, and the fact that Transformer-based models don't produce a single dense representation for every document, Transformer-based models are generally only viable for re-ranking the top documents after some initial retrieval.

Length limitations

Long documents are a big obstacle for Transformers since there is a quadratic dependency ($O(n^2)$) on the document length. In practice, this means that there is a max length restriction of around 256 tokens. This means an aggregation solution is required, or an alternative to Transformers with lower complexity (see subsection 2.3.6). For instance, monoT5 (section 3.1.3) uses passage aggregation with a width of 10 sentences and a stride of 5 - which has been shown to improve performance (Wang et al., 2019) - and PARADE uses an additional Transformer on top to handle passage aggregation.

Diversity

Transformers are incredibly diverse. In retrieval applications alone, they have been used to produce dense representations, produce term embeddings for interaction models, for direct relevance scoring as well as for choosing which of two documents is more relevant. It is not unthinkable that a single model could aid every step of the retrieval process. Although it has yet to come into fruition, there is potential for specialized Transformers (presumably with a lot of training) to dominate nearly all text retrieval tasks.

4.1.4 Takeaways

Due to the lack of pre-trained models and unsupervised methods without significant time and resource investment, along with a lack of good results on even English datasets, it seems fruitless to pursue dense and interaction models much further for the purposes of this Thesis. Taking a pre-trained interaction model and using new embeddings could be an interesting test, though it seems doubtful it would lead to noteworthy results. Dense models seem to have potential, but struggle to outperform a decent BM25 baseline consistently. There is a large number of pre-trained Transformer models available trained on many languages, as well as proven zero-shot multilingual transfer performance. For this reason the focus will mainly be on these.

4.2 Tools and Libraries

Creating tools to work with are time consuming tasks. Thanks to open source libraries a lot of the work is already done by others. This section outlines the tools used to obtain the results in this Thesis.

4.2.1 Python

Python is one of the most used programming languages and is widely used for machine learning. It is an object oriented programming language with a design philosophy focused on readability. With a large standard and third-party library developed by the community. Pytorch and Tensorflow, the most used deep learning libraries created by Facebook and Google respectively, is made on python. All the programming for this Thesis is done in Python, with pure Python and libraries for Python.

4.2.2 Jupyter notebook

Jupyter notebook is an open source web application which makes it possible to run code in the web browser. It is created and maintained by Project Jupyter a community driven project, it is availible for Python, Julia and R. It is a relatively new tool created in 2014 and it has become one of the most used tools for data exploration by data scientists. Advantages of using the notebook is being able to easily explore data and new libraries by having code cells to work in. Each code cell is a part of the code and can be ran in any order the user wants. This allows for a lot of trial and error as it is possible to just run the crashed code again after do small tweaks, without having to run the whole program over again. The notebook also allows you to easily and fast visualise data. All of the experimentation will be done in the notebooks as it allows exploration of new libraries and easy manipulation of the data used for the project.

4.2.3 Pyserini

Pyserini is a python toolkit created to recreate information retrieval research. It gives an interface to easily search through the most used datasets in information retrieval. The library uses Lucene to search and requires Java, which can create some issues. This is an important tool as preparing and indexing the large datasets can take a lot of time. The creators are researchers in the field which makes the library easy to use for experimenting. Their research is also implemented for easy reproduction and the basic search engine uses BM25 (subsection 2.2.3) with optimal settings, which gives good baselines. New datasets like the multilingual ones are not implemented, but it is easy to index new datasets for search.

4.2.4 PyTorch

PyTorch is an open source machine learning library created by Facebook AI research lab. It is one of the most used deep learning libraries along with Tensorflow. PyTorch offers efficient computing for machine learning models and an easy to use interface. Most of the research with neural networks in information retrieval is either made with PyTorch or TensorFlow. In this project PyTorch is used because of familiarity and most of the relevant research have used it.

4.2.5 Huggingface

Huggingface is an organization that offers open source libraries for Transformers and NLP. They also offer a model hub, a place where everyone can upload and share their trained models. This makes it easy to build and continue on other researchers work. The most used architectures of Transformers are implemented, newly published researched is also usually quickly added. Thanks to their implementation of Transformers, most of the PyTorch code is abstracted. In this project only the data loading and some model tweaking has to be written in native PyTorch.

4.2.6 h5py

H5py is a Pythonic interface to the HDF5 data format. This library is used because of the large datasets in information retrieval. With this library large files can be stored on disk, and quickly retrieved while training models. The datasets used for this Thesis is large and requires a lot of preprocessing for training, this makes reading and tokenizing the data a bottleneck for training. To remove the bottleneck the data is preprocessed once and saved to disk, because the data is to large to hold in memory and takes a lot of time to preprocess. Reading the data from disk is faster than processing it while training. h5py is used to save the data to disk and then retrieving it while training, which removes data processing as a bottleneck while training.

4.2.7 TrecTools

TrecTools is an open-source Python library used for TREC evaluation. TREC has its own evaluation tool, but it is written in c++ and can be difficult to use. The advantage of using TrecTools is having an interface to use while working in Python. Results from reranking are written to file when doing inference on the evaluation dataset. The file is then read by TrecTools and compared with the query-relevance file to calculate the metrics used for the results chapter.

4.2.8 Matplotlib

Matplotlib is an open source library for creating visualizations in Python. The library is easy to use and has many tools to visualize data. Exploring and plotting data can be an important step in developing models for different tasks. It is also great for creating figures for the report.

5 Experiments and Results

In this chapter the most interesting results of the research will be displayed, as well as how the research has been conducted. The setup and specific settings, needed to reproduce the results can also be found here.

5.1 Experimental Plan

The experimental plan is split in two parts. Firstly, the plan for the work environment, which explains what is needed to start the exploring of different methods. Secondly, the experimental approach, which goes through the main ideas behind the experiments.

5.1.1 Creating work environment

When experimenting with different methods on different datasets, it is important to create a code environment where it is easy to explore and evaluate new ideas. Having datasets with different properties prepared and easily available for testing will be crucial for fast experimenting. Changing models, parameters, training data and other variables needs to be easy and safe to make sure that when different aspects are evaluated nothing else than what is intentionally changed affect results. To do this the code needs to be clean, modular and easy to read.

Datasets

When selecting datasets to experiment on it is advantageous to use the same datasets as other researchers since it allows direct comparisons of results on data with known properties and good quality. Datasets in neural IR can be very large and time consuming to work with. For initial testing a small and widely used dataset will be sufficient, because this will allow baselines to be established quickly which can be compared to past results. Robust04 (see section subsection 3.3.2) is a relatively small dataset with only 250 queries, widely used by other researchers and supported by most libraries. Creating a baseline with BM25 (subsection 2.2.3) on this dataset and comparing it with others will show that the setup is correct.

After initial setup and testing, a dataset from a different language is needed. TREC-Spanish(subsection 3.3.3) is a dataset consisting of Spanish news articles and queries.

This is a dataset used by other researchers in the field to test zero-shot learning, and will be used for comparison. The datasets need to be easy to use for training and evaluation. This requires a good interface to the datasets, and the the data retrieval needs to be consistent for reliable comparisons. By using Pyserini (see subsection 4.2.3) the initial retrieval from BM25 will be done with correct configurations. To ensure the evaluation of the models is done in a correct way TrecTools (see subsection 4.2.7) library will be used by writing results from testing to files and using TrecTools to read the results and evaluate them.

Training models

Transformers are large and usually require many hours of training. IDUN (Själander et al. (2019)) from the High Performance Computing lab at NTNU offers a large amount of computing power. To share the computing power between users a queue system is used, where users can create a program job to be executed once resources are available. There are many users and the queue time can be from days to hours. The HPC lab will be very useful for training because they offer GPUs with a large amount of memory, which is necessary for the memory hungry Transformers, particularly when training on long documents where input length is important. Due to the queue time on IDUN early experimenting will be performed on personal computers with GPUs (Nvidia RTX2080). Using home computers to experiment will make it easier to find promising methods which can be scaled up and used on IDUN.

5.1.2 Experimental approach

The experimental approach will be split in three parts. Firstly, reproducing the results from related work on the most relevant datasets. Secondly, using the most common methods for zero-shot and few-show learning between English and foreign (Spanish) language datasets. Thirdly, iterative training of models with different parameters to find optimal settings.

Reproducing results from related work

Implementing and using the code of related work will be helpful to learn about their methods and serve as an inspiration for making improvements or new methods. By replicating results from their papers, one can ensure that the implementation works as intended. Additionally, this gives an opportunity to confirm their results.

Some libraries include the most common methods and are easy to use, which is helpful when implementing. Trying to save time will be important and using the path of least resistance will be prioritized when replicating results. When choosing methods to reproduce, the ones with available code and pre-trained models, as well as potential for few-shot and zero-shot learning will be preferred.

Zero-shot learning between English and foreign languages

Because the original Norconsult dataset does not have labels, the experimental approach needs to simulate transferring a model without training on any labels from the testing dataset, only using labels for evaluation. The evaluation dataset is TREC Spanish and will be used to compare the different experiments. If there is time to test, an extra evaluation dataset will be in English to confirm that the performance of the model is still acceptable after trying to implement it for other languages.

With zero-shot learning, data from the evaluation dataset can not be used for training. This means all the training has to be done on different datasets, preferably as similar to the evaluation dataset as possible. One of the most promising methods here is training a multilingual Transformer for relevance classification on one of the large English relevance datasets, then using that model on the evaluation datasets. The intuition here is that the model "learns" about relevance scores in English, but can transfer that knowledge to other languages it is trained on as well.

Tune parameters

After finding models that work for zero-shot learning, tuning them with the correct parameters is important to reach their full potential. Different models may need different tuning depending on their architecture. Parameters which may affect performance include learning rate, batch size and input lengths. To get the most out of the models, they will be tested with different parameters and compared with each other.

5.2 Experimental Setup

The setup is important to make exploring new ideas easy, as well as getting consistent results. In this section everything needed to reproduce the results can be found.

5.2.1 Corpus setup

The first step when dealing with a new project is taking a look at the data. For this project Pyserini was used for indexing and search of all datasets, see section 4.2.3. The most used datasets have available indexes for search, but not TREC Spanish. In this subsection, how to find the indexes and creating the index for TREC Spanish is explained.

Robust 04 and MS MARCO Document and Passage

For TREC Robust 04 and MS MARCO Document and MS MARCO Passage pre-built indexes from Pyserini were used. The indexes are found at the University of Waterloo's

Gitlab[1]. Queries/Topics and relevance assessments is also available directly from the library.

TREC Spanish

The TREC Spanish corpus is not freely available to use and has to be requested from the Linguistic Data Consortium.[2] The queries and relevance assessments can be downloaded from TREC's website. [3] [4]

The corpus is saved as XML an has to be parsed. Each document can be parsed by splitting to full document, ID, Article number, headline and document text like shown below.

```
<DOC>Full document</DOC>
<DOCNO>ID</DOCNO>
<ARTNUM>Article number</ARTNUM>
<HEADLINE>Headline</HEADLINE>
<TEXT>Document text</TEXT>
```

For topics it is also XML and it can be parsed by splitting into Number, description, topic, narr. The topic file is a bit harder to parse. See below for regex expressions.

```
First split each topic with: </top>
<num>\s*Number: ([\s\S]*)<title>
<desc> Description:([\s\S]*)<narr>
<title>  Topic:([\s\S]*)<desc>
<narr>([\s\S]*)
```

The documents then indexed with the following setting:

```
!python -m pyserini.index -collection JsonCollection \
-generator DefaultLuceneDocumentGenerator \
-threads 4 -input <input_file> \
-index <output_folder> -storePositions -storeDocvectors -storeRaw
```

5.2.2 Training models

Transformers requires large amounts of data, and they usually scale well with more data. MS MARCO Documents, shown in subsection 3.3.1, is the largest dataset with 3.2 million documents and is probably the best dataset to train with. Working with too large datasets requires more time for training and pre-processing of data. MS MARCO Passage is smaller than Documents, but the token size of its documents is around 100 in

[1] Waterloo Gitlab: `https://git.uwaterloo.ca/jimmylin/anserini-indexes`
[2] LDC:`https://catalog.ldc.upenn.edu/LDC2000T51`
[3] Queries: `https://trec.nist.gov/data/topics_noneng/index.html`
[4] Relevance assessments: `https://trec.nist.gov/data/qrels_noneng/index.html`

average as can be seen in Figure 5.1. Due to the problems working with large datasets and the token sizes of MS MARCO Passage, Robust 04(subsection 3.3.2) was used for initial training.

Creating training data

All that is available is a indexed corpus with documents and a list of queries and their relevant document. To use that for training the data has to be transformed to be used for Transformer classification tasks. Each query has relevant label for every document, because that is impractal for training only a subset of documents is used for every query. For each of the 250 queries in Robust 04 the top 100 relevant documents are retrieved with BM25 and RM3 subsection 2.2.3. The hyperparamets used for BM25 and RM3 are k1=0.9 and b=0.4 and for RM3 it is terms=10, docs=10 and query weight=0.5 as recommended by Yates et al. (2021).

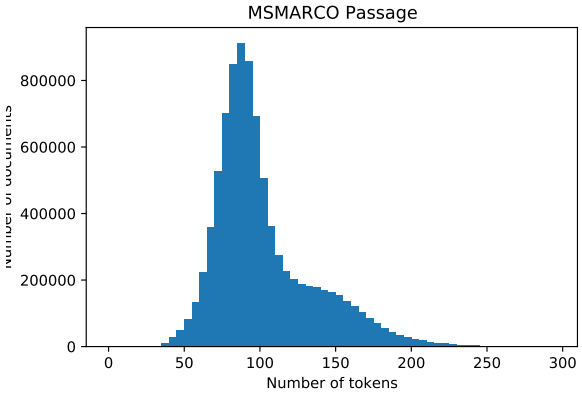
For each training sample a query and document are paired together and tokenized with a [CLS] and [SEP] token as shown here.

```
[CLS]query text[sep] document text
```

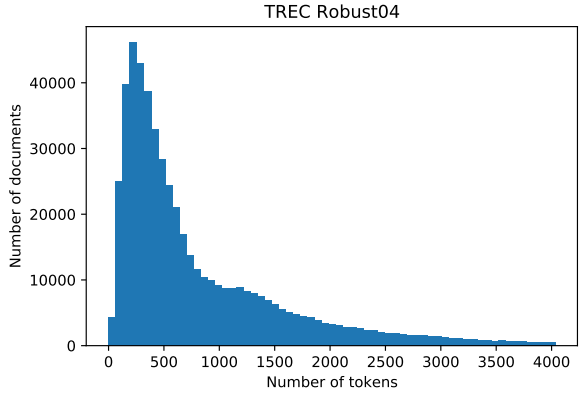
It is tokenized with Huggingface tokenizer with varying max lengths and with a stride of half of the max length. If the document and query together are longer than max length, then the query is kept in full and the document is split up. Retrieving the documents and tokenizing the document and query takes a lot of time and is a huge bottleneck while training. To reduce the time the training samples are pre-computed and saved to disk. Reading from disk can also be a bottleneck and to improve the retrieving of training samples from disk H5py is used, explained in chapter subsection 4.2.6.

For Robust04 a 90/10 split is used on the queries, of the 250 topics 225 are used for training and 25 for validation. The high ratio of training versus validation, and not doing five fold or similar validation is done because the evaluation is done mainly on other datasets. Metrics used for validation of test set are Validation Loss, F1 Score, Precision, Accuracy and Recall.

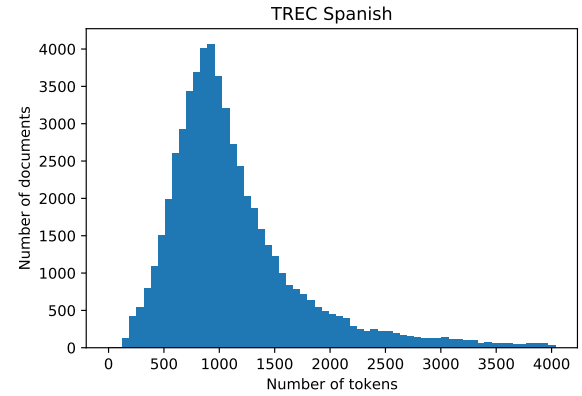
In figure Figure 5.1 a histogram showing number of tokens per document in the different datasets is shown. As seen, most documents contain less than 4000 tokens. For MS MARCO Passage almost all documents contains less than 250 tokens, but for Robust04 and TREC Spanish most documents are longer. When creating a model for these documents, max length of input tokens needs to be considered. Shorter max length is more practical for training, but could reduce the performance of the models. Tokens were created by Huggingface BERT multilingual uncased tokenizer and represent the amount of tokens used for training with BERT multilingual. Different models use different tokenizers and number of tokens per document may vary, depending on tokenization method.



(a) MS MARCO Passage, contains 8.8 million passages/documents.



(b) TREC Robust04, contains 528K documents



(c) TREC Spanish, contains 58K documents

Figure 5.1: Histograms of number of tokens per document for relevant datasets. Tokens created with Huggingface BERT multilingual uncased tokenizer. Figure created with Matplotlib

Initial testing

For initial testing, relatively inexpensive hardware from home is used for training, an NVIDIA RTX 2080 with 8GB of VRAM. Four different models were tested: BERT multilingual, XLM Roberta, Longformer and Big Bird. They were trained using Huggingface Transformers and PyTorch dataset module. Hyperparameters for the models shown in Table 5.1

Model	Epochs	Batch size	Warmup steps	Learning rate
mBERT	3	12	500	2e-5
XLM-RoBERTa	3	8	500	2e-5
Longformer	3	4	500	2e-5
Big bird	3	4	500	2e-5

Table 5.1: Hyperparameters for initial training

HPC trained models

Continued testing was performed on IDUN (Själander et al. (2019)). Due to the better hardware, max length of input could be increased, training could be done faster and the batch sizes increased. For training different GPUs were used depending on what was available on the cluster, they differed from NVIDIA V100 with 16GB or 32GB and NVIDIA P100 16GB. To simulate consistent batch sizes for the efficient Transformers, accumulated gradients was used.

Model	Max length	Epochs	Batch size	Learning rate	Gradient accumulation
mBERT	256	5	32	2e-5	1
mBERT	512	10	12	3e-6	4
XLM-RoBERTa	256	5	32	2e-5	1
XLM-RoBERTa	512	10	8	3e-6	4
BigBird	1024	5	4	3e-6	4
BigBird	2048	5	4	3e-6	4
Longformer	2048	5	4	3e-6	4
Longformer	4096	5	2	3e-6	8

Table 5.2: Hyperparameters for HPC training with IDUN.

5.2.3 Evaluate models

The evaluation is done on the TREC-Spanish Dataset, a dataset the trained Transformers never have seen before. Metrics used are, Precision@30, Precision@20, mean average

precision, Reciprocal rank 100/10 and nDCG@20 (see subsection 2.2.1 for metrics). To calculate the metrics, the Trec Tools library is used.

For each query in the dataset BM25 is used to retrieve the top 100 documents. Then each document is tokenized together with the query and sent to the relevance Transformer models. Each document gets a relevance score to the document. The scores, document ids and query ids are then written to file.

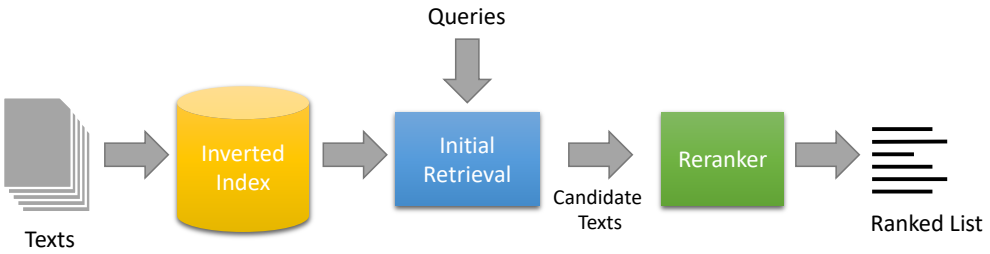


Figure 5.2: Diagram explaining a multi-stage retrieval system. In this Thesis, the initial retrieval is done using BM25, which retrieves the top 100 candidates for each query. Reranker is the model being tested. Diagram from Yates et al. (2021), with permission from Andrew Yates.

With Trec Tools, the query relevance file which contains the labels and file with reranked score is read. The library then compares the files and calculated the metrics. This is done with all the different Transformers to compare results.

5.3 Experimental Results

This section contains the interesting results from the research, which will be discussed later in the report. The results are split into different tables, and the sections with tables are ordered from the progression of the experiments.

5.3.1 Models list

The results shows different models, with long names and different features. Here is a list of the different models and their key features.

BM25 This is the baseline model. It uses BM25 + RM3, see subsection 2.2.3, this is a light weight model and uses a fraction of the time compared to the Transformer based models. This is also the model which is used as initial retrieval for all the other models. Models scoring worse than BM25 have actually just made the initial retrieval worse.

mBERT Cased multilingual BERT, same architecture as regular BERT (section 2.3.6) except it is pre-trained on a multilingual corpus. The different versions are trained on Robust04 for this task.

XLM-RoBERTa XLM-RoBERTa is the multilingual version of the RoBERTa Transformer architecture (section 2.3.6). The different versions of this model are trained on Robust04 for this task.

BigBird BigBird is a Transformer architecture created for longer input lengths then regular Transformers. This model is not multilingual as the pre-training is done only on English documents. The different versions of this model are trained on Robust04 for this task.

Longformer As BigBird, longformer is also created for longer inputs, and is not multilingual. It uses a different approach and uses generally less GPU memory than BigBird. The different versions of this model are trained on Robust04 for this task.

New Dog Results from Teaching a New Dog Old Tricks: Resurrecting Multilingual Retrieval Using Zero-shot Learning by MacAvaney et al. (2019a). It uses a multilingual BERT and is trained on Robust04. This is the only model which was made for testing on the same dataset as this paper, TREC Spanish. The scores are taken from their paper and not reproduced by us.

Rethink A reranker created by Gao et al. (2021). Their model uses regular BERT which is not multilingual. The model is trained on the MS MARCO Document dataset (subsection 3.3.1).

Passage uncased This model uses uncased BERT for reranking. It is created for NBoost, a search engine library, by Thienes and Pertschuk (2021). MS MARCO Passage (subsection 3.3.1) is used for training.

Passage large Same as Passage uncased, except it uses a larger version of BERT, which requires more GPU memory.

Passage mBERT Similar to the other Passage models, except it uses Multilingual BERT. The model is trained on MS MARCO Passage (subsection 3.3.1) by Reissel and Manaj (2021) and uses the approach explained in Nogueira and Cho (2019).

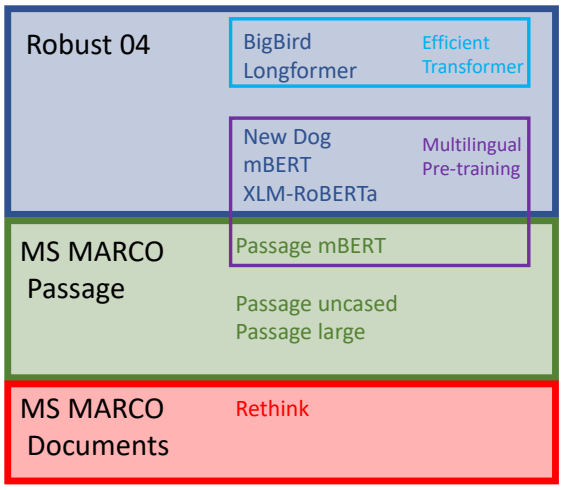


Figure 5.3: Taxonomy of models tested. Outlines which dataset the models were trained for ranking, and marks special properties.

5.3.2 Initial training

In Table 5.3 the results from the initial training is shown. This is done on a home computer with limited hardware. The models are trained with batch sizes which uses highest possible amount of GPU memory, without getting an out of memory error. The setup for the training can be found in Table 5.2.2.

Model	Train Parameters		Metrics			
	Max len	Batch size	MAP	P@20	MRR@100	nDCG@20
BM25 Baseline			0.208	0.650	0.782	0.658
mBERT	256	12	**0.211**	**0.584**	**0.752**	**0.589**
XLM-RoBERTa	256	8	0.203	0.562	0.682	0.570
Longformer	512	4	0.205	0.554	0.638	0.550
BigBird	512	4	0.208	0.566	0.719	0.583

Table 5.3: TREC Spanish initial testing.

As seen in Table 5.3, no model performs better than the BM25 baseline. The best performing model in all metrics is mBERT, with a max length for tokens half of what Longformer and BigBird has. The batch size of mBERT is larger than the other ones, 3 times larger than Longformer and BigBird. Of the models with larger max length, BigBird outperformed Longformer by a substantial margin on nDCG@20 and MRR@100. XLM RoBERTa scored lower than mBERT on all metrics, batch size being the only difference in parameters.

5.3.3 Increasing batch size

In Table 5.4 mBERT and XLM RoBERTA is compared with different batch sizes. The models with larger batch sizes are trained on the HPC IDUN.

Model	Train Parameters		Metrics			
	Max len	Batch size	MAP	P@20	MRR@100	nDCG@20
BM25 Baseline			0.208	0.650	0.782	0.658
mBERT	256	12	0.211	**0.584**	**0.752**	**0.589**
mBERT	256	32	**0.213**	0.578	0.675	0.581
XLM-RoBERTa	256	8	0.203	0.562	0.682	0.570
XLM-RoBERTa	256	32	0.209	0.580	0.612	0.563

Table 5.4: TREC Spanish, increase batch size.

As can be seen in 5.4 mBERT with a batch size of 12 scores highest in all metrics except for MAP. Both mBERT and XLM-RoBERTa does not score any better with larger batch sizes. In the end though, no model outperforms the BM25 Baseline, except for in MAP.

5.3.4 Increasing max length

In Table 5.5 the effects of increasing max length of input tokens is presented. The table is split in two parts, first mBERT and XLM RoBERTA are shown. Those Transformers are not viable with max lengths over 512. In the second part, BigBird and Longformer are shown, they can have considerably longer input lengths. Longformer is shown with up to 4096 in input length, BigBird is only up to 2048 due to memory constraints.

Model	Train Parameters		Metrics			
	Max len	Batch size	MAP	P@20	MRR@100	nDCG@20
BM25 Baseline			0.208	0.650	0.782	0.658
mBERT	256	12	0.211	0.584	0.752	0.589
mBERT	512	12	**0.226**	**0.654**	**0.830**	**0.669**
XLM-RoBERTa	256	8	0.203	0.562	0.682	0.570
XLM-RoBERTa	512	12	0.209	0.576	0.764	0.608
XLM-RoBERTa *	512	48	0.211	0.578	0.764	0.591
BigBird	512	4	0.208	0.566	0.719	0.583
BigBird *	1024	16	0.211	0.600	0.645	0.596
BigBird *	2048	16	**0.217**	**0.618**	**0.870**	**0.645**
Longformer	512	4	0.205	0.554	0.638	0.550
Longformer *	2048	16	0.208	0.570	0.721	0.581
Longformer *	4096	16	0.207	0.576	0.667	0.572

Table 5.5: TREC Spanish, different max lengths. For models marked with *, accumulated gradients is used to reach batch size. The multilingual models are split from the models with English training data only and best performing metrics in each section is bolded.

As shown in Table 5.5, mBERT with a max length for input tokens of 512, is the best scoring model in all metrics by a large margin. XLM-RoBERTa with and max lenght of 512 also scores better than the one with length of 256. For the models with longer input lengths, BigBird with input length of 2048 is the best performing one. All the models improve their metric scores for every increase in max length, except for Longformer with max length of 4096, which performs slightly worse than the 2048 one.

5.3.5 Comparison with other models

In Table 5.6 the models trained for this paper are compared with other publicly available models from Huggingface model zoo. All models are tested by us on the dataset, except for the New Dog model, where the metrics are taken from the paper MacAvaney et al. (2019a). In the first part of the table, models trained by us are shown. The second part of the table contains models found from Huggingface model zoo and the New Dog model. These models use BERT or Multilingual BERT with an input length of 512, with the main difference between them being training methods. Passage models use a larger amount of training data from MS MARCO. New Dog uses Robust04 for training, see subsection 5.3.1 for more information about each model.

Model	Max length	Metrics			
		MAP	P@20	MRR@100	nDCG@20
BM25 Baseline		0.208	0.650	0.782	0.658
mBERT	512	0.226	0.654	0.830	0.669
BigBird	2048	0.217	0.618	**0.870**	0.645
New Dog	512	**0.262**	0.640		0.667
Rethink	512	0.226	0.622	0.763	0.640
Passage uncased	512	0.231	0.676	0.774	0.690
Passage large	512	0.236	0.686	0.842	0.710
Passage mBERT	512	0.247	**0.724**	0.841	**0.739**

Table 5.6: TREC Spanish evaluation results. Models in the first section are trained for this Thesis, models in the other section are trained by others. All results are evaluated for this paper, except for New Dog which are taken from their paper (MacAvaney et al. (2019a))

As can be seen in table Table 5.6, Passage mBERT is the best scoring model in nDCG@20 and P@20 by substantial amount. For MRR@100 BigBird is the best and in MAP New Dog is performing a lot better than all the other models. Passage large sees some improvement over its smaller counterpart Passage uncased. mBERT and New Dog which are trained similarly are performing almost equally, except for in MAP, where New Dog outperforms greatly.

6 Evaluation and Discussion

Results do not speak for themselves, they need to be evaluated and discussed to give any value. This chapter is split in two sections, evaluation and discussion. In the evaluation section, the results from chapter 5 are discussed in depth, and then evaluated with consideration for the goal and research questions. Further down in the discussion section a more general analysis about the work, results and what is learned is presented.

6.1 Evaluation

This section contains the evaluation of the results and the goals and research questions for this Thesis. First the results are evaluated in depth for each table presented in the previous chapter, the progression of the subsections follows the progression of the experiments done for the Thesis. Later in this chapter the goal and research questions are answered with considerations of the results.

6.1.1 Results

The results discussed here can be found in section 5.3. Before evaluating the results, here is a short explanation of the metrics to give some intuition about what they measure. Mean average precision, or MAP for short, is a metric that considers the average precision on all relevant documents, even those not retrieved by the initial retriever. The metric is primarily a test of the initial retriever as there is no top n cut-off for the calculation. Precision at 20 (P@20) is the precision of the top 20 documents from the initial retriever. Mean reciprocal rank (MRR@100) shows how high up the first relevant document appears on average; if the first document always is relevant, the score is 1. Normalized Discounted Cumulative Gain (nDCG@20) considers the top 20 documents; the amount of relevant documents is balanced with how high the relevant documents are. For this evaluation, nDCG@20 is used as the preferred metric for comparisons.

Robust 04 was used to train the models, a relatively small dataset compared to MS Marco Passage. This was done to save time to increase the number of different models and methods explored. Not training on as much data probably reduces the performance compared to models trained on datasets that are orders of magnitude larger, as can be seen in the results. The models trained on smaller datasets did not perform as well, but the results are still valid for comparing methods, which can be used for training models

when scaling up the training data. Although it is not a given that the best methods for small and big datasets are the same, it should be a good indication.

The models are evaluated on the TREC Spanish dataset, a dataset with Spanish news articles. This is a dataset with a decent amount of documents but only contains 25 queries. In the Spanish dataset, most of the documents' token length was under 4000 tokens, with the bulk of them being no more than 1000 tokens long. Ideally, there should have been another dataset with another non-English language. CLEF (Cross-Language Evaluation Forum) had an ad-hoc dataset with different languages, but it was hard to get and was behind a paywall. There was an attempt to get a dataset, but it was not available in time for evaluation as hoped. Not having another dataset to validate reduces the validity of the experiments for zero-shot transferability between languages. The results only show how much is transferred between English and Spanish. Many European languages are similar, and without going in-depth about the similarities between different European languages, one can assume that if it is possible between English and Spanish, English and, for example, Norwegian should not be considerably harder.

Initial training

The initial tests were done with home equipment with sub-optimal hardware because it was easier to use than High-Performance Computing systems (HPC). Two different multilingual models and Transformers designed for long input lengths (Efficient Transformers) were tested. As there are no available multilingual Transformers for long length input, it was interesting to see if input length was more important than having a multilingual model.

As the results showed in Table 5.3, mBERT had the best performance for all metrics, even though the input length of tokens was only half compared to the non-multilingual models. BigBird, an efficient Transformer made for longer input lengths, performed slightly worse than mBERT, which has multilingual pre-training. XLM-RoBERTa and Longformer underperformed slightly, but it is hard to say how much was because of randomness from training. XLM-RoBERTa also had a smaller batch size which could have changed the performance. Another reason why they underperformed could be their pre-training. Neither XLM-RoBERTA or Longformer had next sentence prediction in their pre-training. Next sentence prediction could be a vital step to relevance prediction, as predicting the chance of the following sentence is not too different from predicting the chance of a query and document being relevant.

For continued exploration, it was decided that all models should be investigated further. The Efficient Transformers were only tested with input lengths of 512 due to hardware restrictions, but both were explicitly created for input lengths over 512, which is the soft limit of BERT. The Efficient Transformers had proved work for re-ranking, and even shown performance similar to the multilingual models, which was not a given. It would be interesting to see how they would perform with higher input lengths. The multilingual

models did not differ that much, and it would be interesting to see how they would perform with longer input lengths and larger batch sizes.

Increasing batch size

Before increasing max input length, it was interesting to see if increasing the batch size of the multilingual models would change the performance. In the initial testing, mBERT performed better than XLM-RoBERTa, but mBERT was trained with larger batch sizes. To test this, both models were trained with a batch size of 32, four times larger than the initial testing.

The results from Table 5.4 show that both XLM-RoBERTa and mBERT perform worse in all metrics when using larger batch sizes, except for mBERT in MAP. The drop-off in performance is high for mBERT's MRR@100, but it is not considerably lower for everything else. It is hard to say with minor differences if the batch size impacted or just randomness from training. Also, in the following table (Table 5.5), XLM-RoBERTa was trained with a batch size of 12 and a batch size of 48 (using accumulated gradients). The one with 12 performed slightly better.

Another advantage of a larger batch size can be faster training, depending on what the limitation is. If the limitation is the compute time for the GPU, computing more samples at once will speed up training considerably. This proved to be the case for the setup used for the experiments here, without having data to prove it. This meant that it was probably safe to use as large a batch size as possible without reducing the scores of the results too much, which shortened the time to train models and meant more time for different experiments.

Increasing max length

When looking at the token count for different datasets as shown in Figure 5.1, it can be seen that most documents (not passages) contain more than 256 tokens. For TREC Spanish, the average is over 1000 tokens per document. The question is then, how much does the amount of tokens the Transformer can compute at once affect the metrics?

In Table 5.5 models are compared with different input lengths. The multilingual models, mBERT and XLM-RoBERTa are tested with max lengths of 256 and 512. Going higher is not viable due to memory constraints with these models. BigBird is tested with a max length of 512, 1024, and 2048. Going higher was not possible without GPU memory issues. The Longformer was tested with a max length of 512, 2048, and 4096 and generally required less memory to train than BigBird. Of the multilingual models, mBERT clearly had the best performance beating all models in all metrics except for BigBird in MRR@100. BigBird did the best of the efficient Transformers, beating the Longformer by a considerable amount. The smallest model BigBird 512 had similar results to the best performing Longformer 2048.

The mBERT model with an input length of 512 was the best performing model, with BigBird not far behind. The mBERT model has the advantage of having full attention and being multilingual, but the input length is only a fourth of what BigBird uses. The increase in nDCG@20 between BigBird 512 and BigBird 2048 is a decent amount, and for mBERT, from 256 to 512, it is even greater. Increasing the input length of mBERT is impossible without using a tremendous amount of memory, but it is possible to train the BigBird model on a multilingual dataset. It seems like nDCG@20 increases when using longer input lengths for all models, except for Longformer when using 4096. The data shows longer input lengths are good until at least 2048, but having a multilingual BERT with a longer input length is still better than going for long input lengths with English models.

Comparison with other models.

All the models trained for this paper are trained with the Robust04 dataset, this is because it is a relatively small dataset, which makes it easy to try different models and methods because the training time is reduced. Huggingface provides a public model library with pre-trained models, which is trained by the community. There are some models for re-ranking trained on MS MARCO Passage in the model library, a huge dataset with a large number of queries, and short documents called passages. It is considerably larger than Robust04, but the passages are shorter, about 100 tokens long per document. Compared to TREC Spanish and Robust 04, which is about 500-1500 on average. Robust 04 has 250 queries, while there are just over 1 million queries for MS MARCO Passage. Increasing training data is a standard method to increase performance for models, in Table 5.6 public available models are compared to the models trained for this paper, as well as results taken from the MacAvaney et al. (2019a) paper. This paper uses a multilingual BERT model (New Dog in the table) on the same dataset of TREC Spanish.

The New Dog model, which is the best model found for the TREC Spanish dataset, was outperformed by the models trained on MS MARCO Passage and barely by mBERT, which is trained similarly, except for MAP where New Dog had the best result. In the New Dog paper MacAvaney et al. (2019a) it is not mentioned how they calculated MAP. When calculating MAP, all the relevant documents are considered, and how many documents used for initial retrieval changes the MAP score a lot. For this Thesis, 100 documents were used for initial retrieval, which leaves some of the relevant documents out. Their paper also reports a considerably higher MAP for their BM25 baseline than this paper reports, but when tested with 1000 documents for initial retrieval instead, a similar MAP was achieved. The difference in MAP can probably be attributed to the number of documents used in initial retrieval for re-ranking.

All the Passage models performed great on the TREC Spanish dataset, even those which were not multilingual. This shows that having a large amount of training data is crucial for performance, more important than training on long documents and using multilingual

pre-trained models. The Passage large model performs a bit better than its counterpart Passage uncased, which has fewer parameters. When comparing Passage uncased to Passage mBERT, its multilingual counterpart, the results improve a lot. This implies that scaling up the model parameters is not as important as using a multilingual model. The Rethink model, which is trained on MS MARCO document ranking dataset instead of MS MARCO Passage, scores worse than all the other models. This may imply that MS MARCO Passage is a better dataset to train on than MS MARCO Documents. Further testing on the MS MARCO document dataset would need to be done.

BigBird has the best MRR@100, which is unusual considering it lacks far behind in the other metrics. MRR@100 is a score that measures how high up in the ranking the first relevant documents are on average. This means the BigBird model is better at putting a relevant document at the top of the results but does not have as many relevant documents at the top 20 as the models with high nDCG@20. This can be because BigBird considers more text at once than the other models, but from Table 5.5 the models with longer input length do not have any higher MRR@100. Except for the models with 2048 as input length. There could be something about that exact input length and the TREC Spanish dataset, which is unique. It is hard to say anything conclusive about it without testing on another dataset.

6.1.2 Research Question and Goals

When evaluating results, it is important to return to the original objectives of the Master's Thesis so that one can put them in terms of the goal and research questions.

Research question 1 *What are the zero-shot capabilities of neural document ranking models?*

Most neural models have not shown much in the way of zero-shot performance until very recently with the introduction of Transformer models. Research has shown that many Transformer-based models like monoT5 perform well when transferring between English datasets, and our results show that with enough training specifically on the retrieval task, even English-only models can also be used for zero-shot retrieval on multilingual datasets.

Research question 2 *How can neural document ranking models be adapted to work for new languages?*

There are a few ways to adapt models for new languages. Particularly, the choice of pre-trained model seems to be very important, with models trained on multiple languages providing a boost in performance, although it seems language is less important than being good at the retrieval task in general. Getting the initial word embedding correct might be the most important part, which could also be tried on interaction models, although it was not tested in this Thesis. Dense retrieval seems to be a harder nut to crack, though there is potential for a smarter solution using unsupervised/self-supervised training.

Goal *Find a document ranking model that gives good results across languages using zero-shot learning.*

Although "good results" is quite a vague aim, it seems this has been accomplished under most definitions. The best models tested have outperformed a strong BM25+RM3 baseline on datasets with absolutely no labeled examples. These models are also easy to obtain and use, requiring no training before being implemented, although it is an option for potential further improvement of performance.

6.2 Discussion

Central to scientific research is being able to interpret results, and explain their importance and relevance. Specifically, this section will explore and deduce the role of input length, multilingual models and retrieval-specific training, as well as discuss limitations with the research and other things to consider.

6.2.1 Input length

The length restriction of Transformers is brought up several times in this Thesis, so it is interesting to see whether or not it has any effect. The results show that generally, it seems that increasing max length does improve performance, with the increase from 256 to 512 being quite noticeable. Since the tests are run with strided windows, one might think the effect is mitigated, but it seems looking at as much of the document as possible at once is beneficial. The documents in our datasets are almost always shorter than 2048 tokens, at which the effect of adding to the max length tapers off (Longformer). Computing power limitations meant that testing a standard Transformer like BERT with a reasonable batch size at lengths longer than 512 was infeasible, and IDUN was required to even push it above 256.

Interestingly, models trained on MS MARCO Passage, for which each passage is generally around 100 tokens, are the best performers in testing. This could be because the passage dataset is more specific in its relevance judgement. Passages are shorter and are therefore more clear in their information content, as opposed to documents for which there might only be a small part of the document that is relevant, with the rest adding noise. This is further evidenced by the fact that the Rethink model, which is trained in a similar manner on MS MARCO Documents instead, performs worse than those trained on MS MARCO Passage. It also seems that using the models trained on passages with larger max lengths than they are trained on is not a detriment to their performance.

6.2.2 Multilingual models

The effect of using multilingual models instead of English-only is positive, although smaller than one might expect. Presumably, the most important part of multilingual

models is getting the initial word embedding (meaning) right, since the relationships between words later on are quite similar between languages in the same family. Spanish and English are not all that different, and the massive pre-training involved with these Transformer models likely already contains some Spanish just by chance from the web-crawls used to create training data, making Spanish not entirely unfamiliar. Fine-tuning with semi-supervised pre-training objectives on the retrieval corpus (without relevance labels) could have a similar effect to using a multilingual, but was not tested.

A big problem with training multilingual retrieval models is the lack of standard datasets. Experiments in this Thesis were performed on only the TREC Spanish dataset, which as the name implies only contains a single language. It was used mainly due to its inclusion in MacAvaney et al. (2019a). There are also Arabic and Mandarin equivalents, but these are separate datasets, and as such the actual *multi*lingual capabilities are somewhat of an assumption based on good results when transferring from English to Spanish. The CLEF (Cross-Language Evaluation Forum) Initiative aims to improve access to multilingual datasets, although their datasets also seem to be separated by language. There were plans to include their datasets in testing, but due to unforeseen delays this fell through. Like TREC Spanish, CLEF datasets are behind a paywall, which is a notable barrier to entry and might discourage usage.

Different datasets are also produced in different ways, with some being very thoroughly scanned for relevant documents (Robust04, TREC Spanish) and some take shortcuts by having humans label data from an existing retrieval model (MS MARCO). None of the most popular datasets have graded relevance, which could add some needed nuance to the ranking quality. The datasets used can have significant implications for training as well as for evaluation.

6.2.3 Retrieval specific training

While multilingual training did have a measurable impact, having a model specifically trained for the retrieval task seemed to make the biggest difference. The relevance scoring itself is a more specific task, although the BERT pre-training does also involve next sentence prediction, which is similar. The fact that RoBERTa and Longformer, which have removed the next sentence prediction task from pre-traning, perform worse than their counterparts which include it, indicates that this is an important task to include if one wants a good model for retrieval. Some even more specific retrieval pre-training, such as REALM or PROP (section 3.2) might be the most promising direction for future research, as there are limits to the amount of labeled data one can produce.

There are also many opportunities to try different training procedures. Currently, the training just uses top 100 documents from BM25 for training, however many relevant documents that is. Balancing the number of relevant documents and irrelevant documents in different ways, or sampling from different sources could have performance effects for the final re-ranker. The availability of relevant samples is also affected by how the dataset is

made. From experience, another pitfall to avoid is using accuracy/loss to validate, leading to not stopping soon enough and performance degrading on the retrieval metrics.

Another aspect is that the entire document is labeled as relevant instead of specific sections, adding noise. In question answering datasets, answers are often labeled using spans, which could be an (albeit time consuming to create) alternative, which might also be used to produce text extracts for results. There are also different methods for aggregating results for smaller sequences into full document results which could be tried.

6.2.4 Limitations

Although the results shown are interesting, they are in no way conclusive. As there was limited time, care was used when choosing models to test. Although speeding up training, needing to use IDUN slowed down the iteration process considerably, as the queue system delayed initial startup. Testing more models, on more datasets, several times would have to be done to determine performance conclusively. Constructing ablation studies determining the effect of individual components in model performance would be ideal, instead of inferring them from serendipitous differences/similarities between different models. Another factor is that the improved results compared to MacAvaney et al. (2019a) may be partly due to a slightly improved initial retriever, although there is a large enough difference that this is probably not a significant factor.

6.2.5 Additional considerations

Latency is an aspect of retrieval with re-rankers which is rarely discussed or tested. On inference, the slowest model (BigBird) used up to 20 seconds to re-rank top 100 documents. This is unacceptable in any practical application, and likely invalidates any ranking accuracy benefit if it can not be sped up significantly.

Furthermore, the hardware requirements for large language models are a big obstacle. Having a GPU cluster to train/run on is very rare, and these GPUs consume a lot of electricity, meaning there is an environmental impact as well. Weighing all this up, the slight improvement in ranking performance provided by increasing max length may not be worth it. Finding lighter alternatives to language models should definitely be considered.

Bias in search engines could be a large problem, especially when queries are political or about disputed topics. How the models rank controversial documents is difficult, if not impossible to predict. The results would be affected by the labels from the training data, first from any language model pre-training, then by the retrieval specific training data. This problem is not considered in this Thesis but should be examined by everyone who provides a public search engine.

7 Conclusion and Future Work

Neural retrieval seems to be going through a transition, with new models showing substantial improvements for the first time in years. Promising results are sure to drive and inspire further research with renewed optimism. Zero-shot and multilingual retrieval are important aspects of this that warrant further examination. It is clear that this is just the beginning of the neural retrieval success story, and that the coming years will have lots of new and interesting solutions.

Currently, neural retrieval in practice is only available to a select few organizations with the resources to support it. Going forward, work on multilingual, zero-shot, low-resource and publicly available models can make neural retrieval freely available to the masses. Hopefully, this Thesis serves as a step towards the democratization of neural retrieval models and their application to diverse datasets with multiple languages.

Following is an overview of this Thesis's contributions to the field, as well as proposed directions for future work.

7.1 Contributions

This Master's Thesis provides an overview and evaluation of different methods for multilingual zero-shot neural retrieval. Several pre-trained Transformer models are tested on the TREC Spanish dataset, with the best results showing state-of-the-art performance.

Traditional neural retrieval methods are explained and discussed, providing a rationale for focusing on Transformer-based re-rankers; BM25 is a sufficient initial retriever, and interaction-based re-rankers do not seem to be good enough.

It is shown that for multilingual retrieval using transferred Transformer-based models, training more on the retrieval task even with English-only datasets has a larger impact than using a multilingual model. Using both in conjunction provided the best performance. Increasing max length also leads to improvements, and models pre-trained with the next sentence prediction objective seemed to have an advantage.

In addition, this Thesis seems to be the first application of efficient Transformer architectures (Longformer and BigBird) on the document re-ranking task. However performance on more common English datasets are not examined.

7.2 Future Work

It is quite clear that there is massive potential for neural retrieval methods. Evidently, the largest obstacle is getting enough labeled training data, although clever engineering could mitigate this. There are several avenues for future research.

More data

This might be the most obvious solution, but the availability of labeled data is likely what puts large search companies ahead when it comes to applying neural models to retrieval. The largest available dataset currently is MS MARCO, which is English-only and contains 3.2M documents, only a tiny fraction of Bing's total data quantity. Constructing larger datasets, with more diversity in both language and domain, and higher quality (perhaps through graded relevance judgments) would enable higher quality models which can handle more use-cases.

Relatedly, the lack of test datasets in different languages is a detriment. This Thesis only tested on TREC Spanish due to its availability and similarity to Norwegian (compared to the other TREC alternatives of Mandarin and Arabic). Obtaining and making available similar datasets for additional languages would enable development of more robust methods and ensure that models can perform consistently. The CLEF (Cross-Language Evaluation Forum) Initiative is a step in this direction, but like TREC Spanish, it has a barrier of entry in the form of a pay-wall.

Smarter solutions

Methods like REALM and PROP (section 3.2) seem to be the kind of solutions that are needed more of in the future. Creating self-supervised pre-training tasks which transfer to retrieval - beyond the already useful language modeling in Transformer models - could be exactly what is needed to push retrieval performance to the next level. Another alternative is augmenting the data by translating documents and queries using another language model, a task which they excel at.

Going "back to the roots" of IR, a more mathematically grounded formulation of the document ranking task may be possible. It seems this has been to some extent lost in recent research, focusing on getting some working model by just feeding it data, rather than building it up from first principles.

Custom architecture

Transformer-based models are ill-suited to the retrieval task in many ways, with length restrictions, speed limitations and possibly excessive model size for the retrieval task. A custom architecture along with a good training methodology tailored for retrieval could overcome these limitations. It does not have to be revolutionary, perhaps as simple

as multilingual pre-training on an efficient Transformer with global attention on the query.

As an example, using an architecture similar to the Perceiver (Jaegle et al., 2021) one could use the query text to generate query vectors for attention, and all the document text to produce keys and values. This could be done multiple times over to create a contextualized representation of the query with minimal compute, as complexity is reduced from $O((q+d)^2)$ to $O(q\,d)$. At the end one could use the embeddings to create a relevance score.

Dense retrieval

While this Thesis did not explore dense retrieval much due to BM25's already good performance, it is still an interesting direction which is worth further exploration. Being able to beat the performance of a strong BM25 initial ranker using a dense retriever - particularly without supervised training - would be a big step forward, especially since there has been little progress in performance since BM25 was introduced in the 90s. A better initial ranker may even be able to boost the performance of the re-ranker beyond the sum of their parts.

Bibliography

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1352. URL https://www.aclweb.org/anthology/D19-1352.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkg-mA4FDr.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. doi: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9. URL https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of BERT rerankers in multi-stage retrieval pipeline, 2021. URL https://arxiv.org/abs/2101.08751.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, Oct 2016. doi: 10.1145/2983323.2983769. URL `http://dx.doi.org/10.1145/2983323.2983769`.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020. URL `https://arxiv.org/abs/2002.08909`.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050, 2014. URL `https://proceedings.neurips.cc/paper/2014/hash/b9d487a30398d42ecff55c228ed5652b-Abstract.html`.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450322638. doi: 10.1145/2505515.2505665. URL `https://doi.org/10.1145/2505515.2505665`.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1110. URL `https://www.aclweb.org/anthology/D17-1110`.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek, editors, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 279–287. ACM, 2018. doi: 10.1145/3159652.3159689. URL `https://doi.org/10.1145/3159652.3159689`.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. *CoRR*, abs/2103.03206, 2021. URL `https://arxiv.org/abs/2103.03206`.

Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. Which BM25 do you mean? A large-scale reproducibility study of scoring variants. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 28–34. Springer, 2020. doi:

10.1007/978-3-030-45442-5_4. URL https://doi.org/10.1007/978-3-030-45442-5_4.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020. URL https://arxiv.org/abs/2004.04906.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *CoRR*, abs/2004.12832, 2020. URL https://arxiv.org/abs/2004.12832.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291, 2019. URL http://arxiv.org/abs/1901.07291.

Quoc V. Le and Tomás Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL http://arxiv.org/abs/1405.4053.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. PARADE: passage representation aggregation for document reranking. *CoRR*, abs/2008.09093, 2020. URL https://arxiv.org/abs/2008.09093.

Davis Liang, Peng Xu, Siamak Shakeri, Cícero Nogueira dos Santos, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. Embedding-based zero-shot retrieval through query generation. *CoRR*, abs/2009.10270, 2020. URL https://arxiv.org/abs/2009.10270.

Robert Litschko, Ivan Vulic, Simone Paolo Ponzetto, and Goran Glavas. Evaluating multilingual text encoders for unsupervised cross-lingual retrieval. *CoRR*, abs/2101.08370, 2021. URL https://arxiv.org/abs/2101.08370.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1075–1088, Online, April 2021a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.eacl-main.92.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. Prop: Pre-training with representative words prediction for ad-hoc retrieval. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, Mar 2021b. doi: 10.1145/3437963.3441777. URL http://dx.doi.org/10.1145/3437963.3441777.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. A replication study of dense passage retriever. *CoRR*, abs/2104.05740, 2021c. URL https://arxiv.org/abs/2104.05740.

Sean MacAvaney, Luca Soldaini, and Nazli Goharian. Teaching a new dog old tricks: Resurrecting multilingual retrieval using zero-shot learning. *CoRR*, abs/1912.13080, 2019a. URL http://arxiv.org/abs/1912.13080.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: contextualized embeddings for document ranking. *CoRR*, abs/1904.07094, 2019b. URL http://arxiv.org/abs/1904.07094.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL http://arxiv.org/abs/1310.4546.

Pandu Nayak. Understanding searches better than ever before. https://www.blog.google/products/search/search-language-understanding-bert/, Oct 2019. (Accessed on 05/31/2021).

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. URL http://arxiv.org/abs/1901.04085.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424, 2019. URL http://arxiv.org/abs/1910.14424.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL https://www.aclweb.org/anthology/2020.findings-emnlp.63.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng.

Text matching as image recognition. *CoRR*, abs/1602.06359, 2016. URL http://arxiv.org/abs/1602.06359.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL https://doi.org/10.3115/v1/d14-1162.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Prabhakar Raghavan. How AI is powering a more helpful google. https://blog.google/products/search/search-on/, Oct 2020. (Accessed on 06/03/2021).

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Philipp X Reissel and Igli Manaj. amberoad. https://huggingface.co/amberoad/bert-multilingual-passage-reranking-msmarco, 2021. (Accessed on 06/10/2021).

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, January 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 101–110. ACM, 2014a. doi: 10.1145/2661829.2661935. URL https://doi.org/10.1145/2661829.2661935.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. *CIKM 2014 - Proceedings of the 2014 ACM International Conference on Information and Knowledge Management*, pages 101–110, 11 2014b. doi: 10.1145/2661829.2661935.

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

Karen Sparck Jones. A statistical interpretation of term specificity and its application

in retrieval. *Journal of Documentation*, 28(1):11–21, Jan 1972. ISSN 0022-0418. doi: 10.1108/eb026526. URL https://doi.org/10.1108/eb026526.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *CoRR*, abs/2009.06732, 2020. URL https://arxiv.org/abs/2009.06732.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Philip Pham Dara Bahri, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Cole Thienes and Jack Pertschuk. Nboost: Neural boosting search results. https://github.com/koursaros-ai/nboost, 2021. (Accessed on 06/10/2021).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. *CoRR*, abs/1908.08167, 2019. URL http://arxiv.org/abs/1908.08167.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. *CoRR*, abs/1706.06613, 2017. URL http://arxiv.org/abs/1706.06613.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. *CoRR*, abs/1904.09171, 2019a. URL http://arxiv.org/abs/1904.09171.

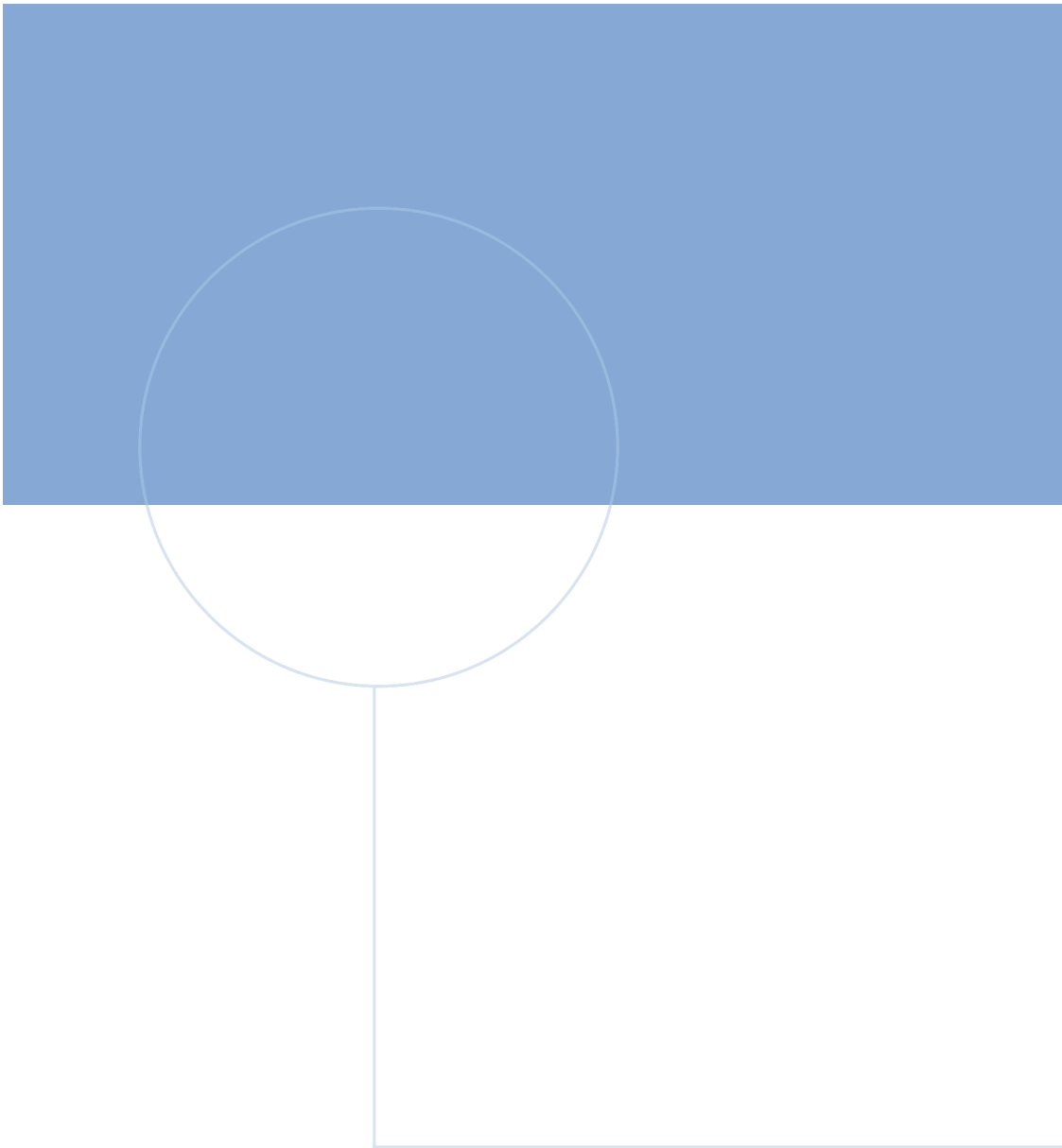
Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of BERT for ad hoc document retrieval. *CoRR*, abs/1903.10972, 2019b. URL http://arxiv.org/abs/1903.10972.

Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 861–864, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368223. doi: 10.1145/3336191.3371868. URL https://doi.org/10.1145/3336191.3371868.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: BERT and beyond. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 1154–1156, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441667. URL https://doi.org/10.1145/3437963.3441667.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Edwin Zhang, Nikhil Gupta, Raphael Tang, Xiao Han, Ronak Pradeep, Kuang Lu, Yue Zhang, Rodrigo Nogueira, Kyunghyun Cho, Hui Fang, and Jimmy Lin. Covidex: Neural ranking models and keyword search infrastructure for the COVID-19 open research dataset. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 31–41, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sdp-1.5. URL https://www.aclweb.org/anthology/2020.sdp-1.5.



NTNU

Norwegian University of
Science and Technology