DESIGNING AN INFORMATION FRAMEWORK FOR SEMANTIC SEARCH

(SEMANTİK ARAMA İÇİN BİLGİ ÇERÇEVESİ TASARLANMASI)

by

ALPER MITINCIK, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

FEBRUARY 2022

ACKNOWLEDGMENTS

When the work got harder, I kept asking myself why I decided to apply to this programme. I could find a comfortable job and have fun at my leisure time instead and have a less stressful last year. This work was optional and I had already worked as a software engineer and it seemed kind of unnecessary. Looking back now, I think it was one of the best decisions I have made during my professional career. It has become a milestone for my further career. I feel very lucky. Finally, I am proud to be a part of Galatasaray University.

TABLE OF CONTENTS

LIST OF SYMBOLS

CNN	: Convolutional Neural Network
IR	: Information Retrieval
LSTM	: Long Short-Term Memory
MAP	: Mean Average Precision
NDCG	: Normalized Discounted Cumulative Gain
NLP	: Natural Language Processing
QA	: Question Answering
RNN	: Recurrent Neural Network

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

New generation information retrieval procedures provide complex tools to remodel the design of search engines. Even though semantic analysis is gradually adopted by corporations, complex behavior of knowledge behind the information entails subsequent data learning models. Text models are currently in use through lexical features. Search engines with lexical methods lack contextual and semantic information. This barrier has been overcome with the development of deep learning methods. More accurate results can be retrieved by obtaining contextual information of different types of content such as text, image, video with neural models. In this study, a broad perspective of search engines was considered through lexical and semantic features. Semantic search methods were experimented then compared with lexical methods in data sets consisting of scientific documents. Since scientific documents are relatively well-formatted datasets and do not contain irrelevant content, the focus was on comparing semantic search methods and neural models throughout the study, without dealing with out-of-context data and semantic conflicts. As a result, semantic search methods performed better than lexical search. We conclude that current search-retrieval tasks require new perspectives in semantics where multimodal information is handled with deep learning strategies.

Keywords : Information retrieval, Semantic search, Deep learning, Re-ranking, Dense retrieval

RÉSUMÉ

Les procédures de recherche d'informations de nouvelle génération fournissent des outils complexes pour remodeler la conception des moteurs de recherche. Même si l'analyse sémantique est progressivement adoptée par les entreprises, le comportement complexe des connaissances derrière l'information entraîne des modèles d'apprentissage des données ultérieurs. Les modèles de texte sont actuellement utilisés par le biais de fonctionnalités lexicales. Les moteurs de recherche avec des méthodes lexicales manquent d'informations contextuelles et sémantiques. Cette barrière a été surmontée avec le développement de méthodes d'apprentissage en profondeur. Des résultats plus précis peuvent être récupérés en obtenant des informations contextuelles sur différents types de contenu tels que du texte, une image, une vidéo avec des modèles neuronaux. Dans cette étude, une large perspective des moteurs de recherche a été considérée à travers des caractéristiques lexicales et sémantiques. Des méthodes de recherche sémantique ont été expérimentées puis comparées à des méthodes lexicales dans des jeux de données constitués de documents scientifiques. Étant donné que les documents scientifiques sont des ensembles de données relativement bien formatés et ne contiennent pas de contenu non pertinent, l'accent a été mis sur la comparaison des méthodes de recherche sémantique et des modèles neuronaux tout au long de l'étude, sans traiter de données hors contexte et de conflits sémantiques. En conséquence, les méthodes de recherche sémantique ont donné de meilleurs résultats que la recherche lexicale. Nous concluons que les tâches de recherche-récupération actuelles nécessitent de nouvelles perspectives en sémantique où les informations multimodales sont traitées avec des stratégies d'apprentissage en profondeur.

Mots Clés : Recherche d'informations, Recherche sémantique, Deep learning, Reranking, Recherche dense

ÖZET

Yeni nesil bilgi arama prosedürleri, arama motorlarının tasarımını yeniden şekillendirmede karmaşık araçlar sağlamaktadır. Anlam tabanlı analiz profesyonel uygulamalarda kademeli olarak benimsense dahi, bilginin arkasındaki karmaşık birikimin davranışı, kademeli veri öğrenme modellerini gerektirmektedir. Metin modelleri sözlük tabanlı özelliklere dayalı olarak kullanılmaktadır. Sözlüksel yöntemlere sahip arama motorları, bağlamsal ve anlamsal bilgilerden yoksundur. Bu engel derin öğrenme yöntemlerinin geliştirilmesiyle aşılmaktadır. Metin, resim, video gibi farklı içerik türlerinin bağlamsal bilgileri sinir ağı modelleriyle elde edilerek daha doğru sonuçlara ulaşılabilir. Bu çalışmada, sözlüksel ve anlamsal özellikler üzerinden arama motorlarına geniş bir perspektiften bakılmıştır. Anlamsal arama yöntemleri denenmiş ve bilimsel dokümanlardan oluşan veri setlerinde sözlüksel yöntemlerle karşılaştırılmıştır. Bilimsel belgeler nispeten iyi biçimlendirilmiş veri kümeleri olduğundan bağlam dışı veriler ve anlamsal çatışmalarla uğraşmadan, çalışma boyunca anlamsal arama yöntemlerini ve sinir modellerini karşılaştırmaya odaklanıldı. Böylelikle, anlamsal aramanın sözcüksel aramadan daha iyi performans gösterdiği gözlenmektedir. Mevcut bilgi arama-bulma görevlerinin, çok modlu veri kümelerinin derin öğrenme stratejileriyle işlendiği anlambilimde yeni bakış açıları gerektirdiği sonucuna varılmıştır.

Anahtar Kelimeler : Bilgi çıkarımı, Semantik arama, Derin öğrenme, Tekrar sıralama, Yoğun çıkarım

1 INTRODUCTION

Modern information technologies are characterized through the adaptive search-retrieval tasks. The dynamics of these procedures are coupled with the relevance between the information representations and the insights of the dataset. Even though the dataset is defined with a structured data map between the documents and the queries, there are several issues during the information design. In a textual search-retrieval task, each word in a collection of documents is counted to find the connectivity which is also known as the frequency between the documents and the search query. If words in a query appear several times in a document, it is assumed that this document becomes more relevant to the query. However, word counts tend to increase if document size becomes longer. Therefore, document lengths should be computed in search-retrieval task. The optimization procedures are generally applied to generate appropriate indexes to reduce time issues. An inverted index containing the frequencies and positions of words is considered as an efficient option to perform the lexical search. Unfortunately, this mechanism does not provide contextual information of text. Lexical search might lead to retrieve irrelevant documents with polysemic properties.

In recent years, the academic scope and the corporate point of view have started to boost the number of complex approaches in semantic search-retrieval tasks. Semantic search focuses on the contextual meaning of the documents rather than the conventional lexical matching. Semantic search seeks to improve the search accuracy by understanding the content of the search query. The procedure is applied using the embeddings over all entries in the corpus into a vector space. The formalism of the query design is also embedded into the same vector space and the closest embedding.

In a nutshell, three different methods are formulated for semantic search. Firstly, the sparse retrieval uses neural models where the learning function in search-retrieval lies on the token-level contextualized representations. The indexing procedure takes a long time and high storage space is the main challenge during the implementation. Secondly, the dense retrieval consists of neural model where the set of queries and document datasets are encoded in a dense vector space. Transformer based bi-encoder models are capable of encoding contexualized representations fast and efficiently. Finally, the

re-ranking method leads to two stages in retrieval pipeline. The first stage is the lexical retrieval from conventional index and the latter is re-ranking the retrieved documents by using cross-encoder models. It is required high computational overhead.

Information retrieval (IR) can be applied in different media such as image, video, news, document, product. In this study, scientific documents are selected as datasets. Scientific documents are generally more structured, well-formatted for search-retrieval tasks. The Out-of-context issue is also less frequent in scientific datasets. Moreover, the information retrieval from the scientific documents can be useful in many different fields such as fact checking in the insights during the postprocessing of relevant documents through the queries.

In this study, we have used four scientific documents datasets from different domains. In addition, two non-scientific documents datasets are added to give an idea about the results in other textual experiments. In order to compare the results with the conventional approach, BM25 scoring is applied as a lexical search method. 16 different neural models are used, including one model in sparse retrieval, 12 models in dense retrieval, and 3 models in re-ranking. For each dataset, the total number of documents, the total number of terms, the total number of unique terms, the total number of sentences, the average number of terms per sentences, the average number of terms per documents, the total number of queries are calculated. The lexical search by using BM25 and semantic search methods such as dense retrieval, re-ranking are explained in the following chapter. Embedding, attention mechanism, transformer architecture concepts are detailed in methods. All experiments are evaluated by using different metrics in the results chapter.

The aim of the analysis was to reveal the promising effects of neural information retrieval in a comparative study. We have compared the procedures through several parameters to underline the complexity of the insights in query-document pairs. The study has been presented as follows. The second chapter details the perspective of our methodology where the datasets, the search-retrieval tasks, the evaluation procedures and the framework have been described. The following chapter addresses the results in a comparative hierarchy. The best results have been highlighted in the tables. As a result, ranking success increased 4% on average in all datasets in the re-ranking method. Between the models used in dense retrieval, pooling and scoring are compared in the same base models. Mean pooling has performed better on 3 of 4 scientific datasets.

Cosine similarity performed better for the same base models. Finally, we have conluded our study by comparing lexical and semantic approaches in a broad perspective in the final chapter where the future steps were discussed.

2 LITERATURE REVIEW

BM25 algorithm is considered as one of the most common scoring functions in the lexical search. Robertson and Zaragoza have developed the most generic retrieval model using BM25 where the similarities and differences with other retrieval frameworks have been analyzed. They have given an overview of optimization techniques by tuning the different parameters in the models (Robertson and Zaragoza, 2009).

The attention mechanism is a promising milestone for the semantic search. Vaswani et al. have introduced a semantic information architecture where the transformer model had an eschewing recurrence to allow significantly more parallelization (Vaswani et al., 2017). This mechanism had achieved a new state in machine translation quality. Experiments on two machine translation tasks have shown these models to be superior according to semantic relevance while being more parallelizable and requiring less training time. Attention models catch on in semantic information tasks where the sequential procedures and transduction tasks are allowing to analyze query-document pairs through their dependencies on input-output similarity distances.

Furthermore, Devlin et al. have introduced a new language representation model in attention-based information analysis (Devlin et al., 2019). Bidirectional Encoder Representations from Transformers (BERT) became one of the most pioneering models using the transformer architecture in the semantical field. BERT was originally designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model became popular in IR and natural language processing (NLP) to handle high level issues related to data semantics. Even if the computational complexity of BERT is higher, promising results have been shown recently in question answering (QA) and IR without substantial task-specific architecture modifications.

In a similar way, Reimers and Gurevych have presented Sentence-BERT, a modification of the pre-trained BERT for Siamese language onto a triplet network structure (Reimers and Gurevych, 2019). They have applied the cosine similarity to derive the semantical proximity in the sentences with the embeddings. Hofstätter et al. have proposed a cross-architecture training procedure that adapted knowledge distillation

to the varying score output distributions of different BERT and non-BERT passage ranking architectures (Hofstätter et al., 2021). They have evaluated the procedure of distilling knowledge from state-of-the-art concatenated BERT models to four different efficient architectures. Macdonald and Tonellotto have proposed a framework called PyTerrier which allowed a complex series of retrieval flowchart. The framework was suitable to optimize automatically the retrieval pipelines to increase their accuracy scores (Macdonald and Tonellotto, 2020).

Thakur et al. have introduced Benchmarking-IR, a robust and heterogeneous evaluation benchmark for the information retrieval (Thakur et al., 2021). They have leveraged a careful selection of 18 publicly available datasets from diverse text retrieval tasks and domains and evaluated 10 state-of-the-art retrieval systems including lexical, sparse, dense, late-interaction, and re-ranking architectures on the benchmark. The dense and sparse-retrieval models have shown efficient results regarding the computational power.

3 METHODOLOGY

The flowchart of IR consists of two major pipelines ; the indexing and the retrieval. In the indexing step, the values such as the frequency and the position are calculated on the collected data, then converted into an easy-to-find format and saved. In the the retrieval step, the relevant documents are found by expanding and analyzing the query. There are two common search methods in which these processes are applied : lexical and semantic approaches.

3.1 Lexical Search

In a simple search method, word counts in all documents is counted to find documents related to typed query. If words in query appear many times in some documents, it is assumed that those documents are related to query, then documents are ranked accordingly. It is not a semantic approach. Also, words counts tends to increase if document is too long. Therefore, document lengths should be used in calculation of score. If calculation is done at query time, it can take a very long time. Pre-calculation should be done to increase efficiency. Proper data structure should be used. Using an inverted index is a good solution.

Inverted index is an efficient method of information retrieval from large collections. It keeps all necessary statistics for ad-hoc information retrieval such as document frequency, term frequency per document, document length, average document length. Metadata such as name and location for document can also be saved in inverted index.

There are some essential natural language processes such as tokenization, normalization, stemming, filtering stop words for each word. Surely, these should apply while indexing. In tokenization step, documents are split into words, punctuation marks and numerical values. In normalization step, case folding and correction of abbreviations are applied. Lemmatization and stemming are used to reduce to a common base form. All these linguistic models are language dependent. For example, the sentence "Information retrieval is the process of obtaining information system resources that are relevant to an information need from a collection of those resources" turns into these tokens :

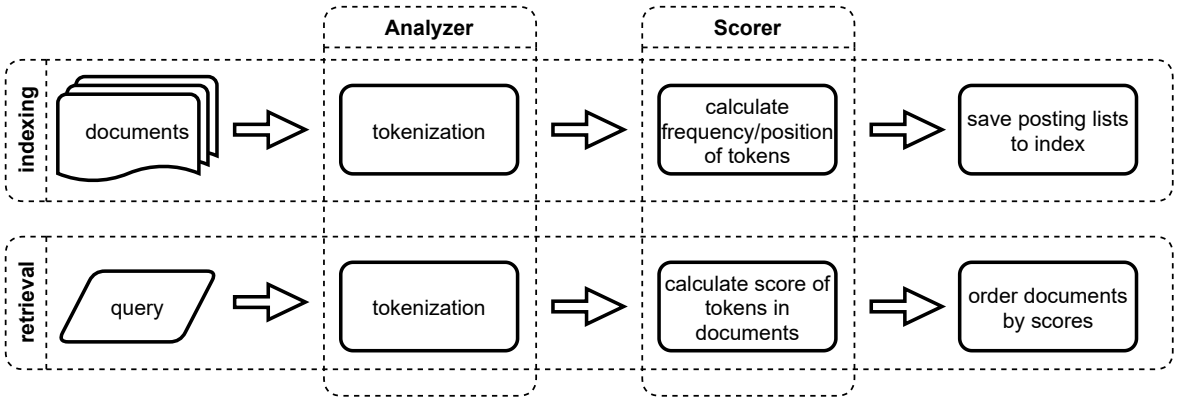


FIGURE 3.1 – Process of Lexical Search

information, retrieval, process, obtaining, system, resource, that, relevant, need, from, collection, those. At the end of process, a term list is occurred then these are added to dictionary and posting list. This is inverted index. There are positions and frequency values of terms in index. It is not necessary to read all documents in querying time, relevant documents can be found.

Frequency values and positional information are used for scoring. The most common function for score calculation is Okapi BM25 which includes TF-IDF. TF-IDF (term frequency-inverse document frequency) is count based function. Term counts for TF are calculated with logarithm function, so changes of term frequency become getting smoother. It also applies to IDF. Then, sum over all query terms, that are in index.

$$\text{TF-IDF}(d,q) = \sum_{t \in T_d \cap T_q} log(1 + tf_{t,d}) \cdot log(\frac{|D|}{df_t}) \qquad (3.1)$$

In a document, both term frequencies and document frequencies with term positions are used. An example document and term statistics for **cell** are in below.

```
{
  "doc" : "The visualization of biologically relevant molecules and
  activities inside living cells continues to transform
  cell biology into a truly quantitative science..."
 }
```

```
"cell" : {
  "doc_freq" : 2878,
```

```
  "ttf" : 11645,
  "term_freq" : 3,
  "tokens" : [
    {
      "position" : 10,
      "start_offset" : 82,
      "end_offset" : 87
    },
    {
      "position" : 14,
      "start_offset" : 111,
      "end_offset" : 115
    },
    {
      "position" : 30,
      "start_offset" : 222,
      "end_offset" : 226
    }
  ]
}
```


BM25 is based on probabilistic information retrieval, uses TF-IDF. First defined in 1994 by Robertson et al., it was used as default scoring model in Apache Lucene in 2015. In addition to TF-IDF, hyperparameters k and b are used. k determines term frequency scaling. 0 is binary model, greater is raw term frequency. b determines document length normalization. 0 means no normalization. These hyperparameters are used in that ranges commonly : $0.5 \leqslant b \leqslant 0.8$ and $1.2 \leqslant k \leqslant 2$. A saturation function is used to optimize change in score values. BM25F, calculate score for multiple fields (or streams) in a document. For example, 3 streams may be used per document : title/abstract/body. Each stream can have different weights. Saturation function is also implemented at stream level. In BM25 formula, $|d|$ is used as document length in word, $avgdl$ is used as average document length in index. The score of a document d given a query q which contains the words $q_1, ..., q_n$ is given by :

$$\text{BM25}(d,q) = \sum_{i=1}^{n} \frac{IDF(q_i) \cdot TF(q_i,d) \cdot (k_1+1)}{TF(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})} \quad (3.2)$$

As shown in Figure 3.1, in the indexing step, all documents are analyzed as language-dependent or language-independent, and terms are retrieved. Terms are counted, average document lengths are calculated for use in BM25 function, documents are counted. Posting lists are created with terms and position data. All values are saved in the lexical index. In the retrieval step, the query is analyzed similarly. BM25 scores are calculated using the terms in the query and all documents containing the terms, and the documents are ranked accordingly.

3.2 Semantic Search

The semantic information retrieval considers the contextual basis. Neural models are capable of finding contextual information. The embeddings are generated for both queries and documents via neural models. The similarity between these embeddings is calculated and ranked. The global mechanism of semantic search is given in Figure 3.2.

Finding more relevant document for query depends on semantic understanding of query and texts. For example, to find physical proximity between two people, distance of their location in space is calculated. Assume corpus as space and texts as person. It is necessary to define position of each text. This can be demonstrated by vector representation. Vector representation describer is an embedding. Embeddings can be determined using different levels. Word embedding, char n-gram embedding, sentence embedding, document embedding etc. Different neural models use different levels of input representation and learning representation. Word embedding is commonly used. Word embeddings allow to apply arithmetic properties.

In word embeddings, vector representations of words are provided. In vector space, dimensions are abstract, so it can not be inferred as if the 25th dimension means. Vector space allows math operations. Vectors use with a simple data structure : Dictionary<string, float[]>. It can be used as 1-word-1-vector (Word2Vec, GloVe), 1-word-1-vector+char-n-grams (FastText) (Bojanowski et al., 2017). Usually distance between two embeddings is calculated, especially cosine similarity is widely used. The closest neighbors are also semantically relevant.

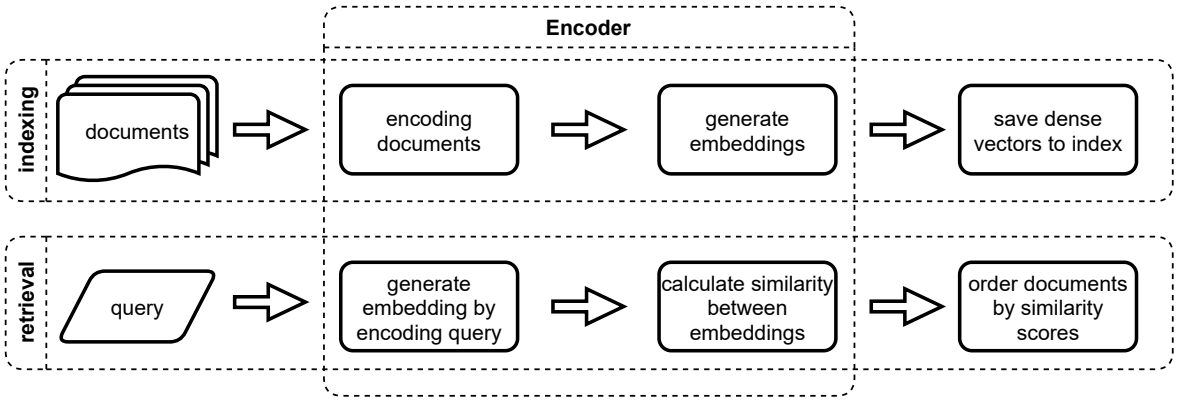
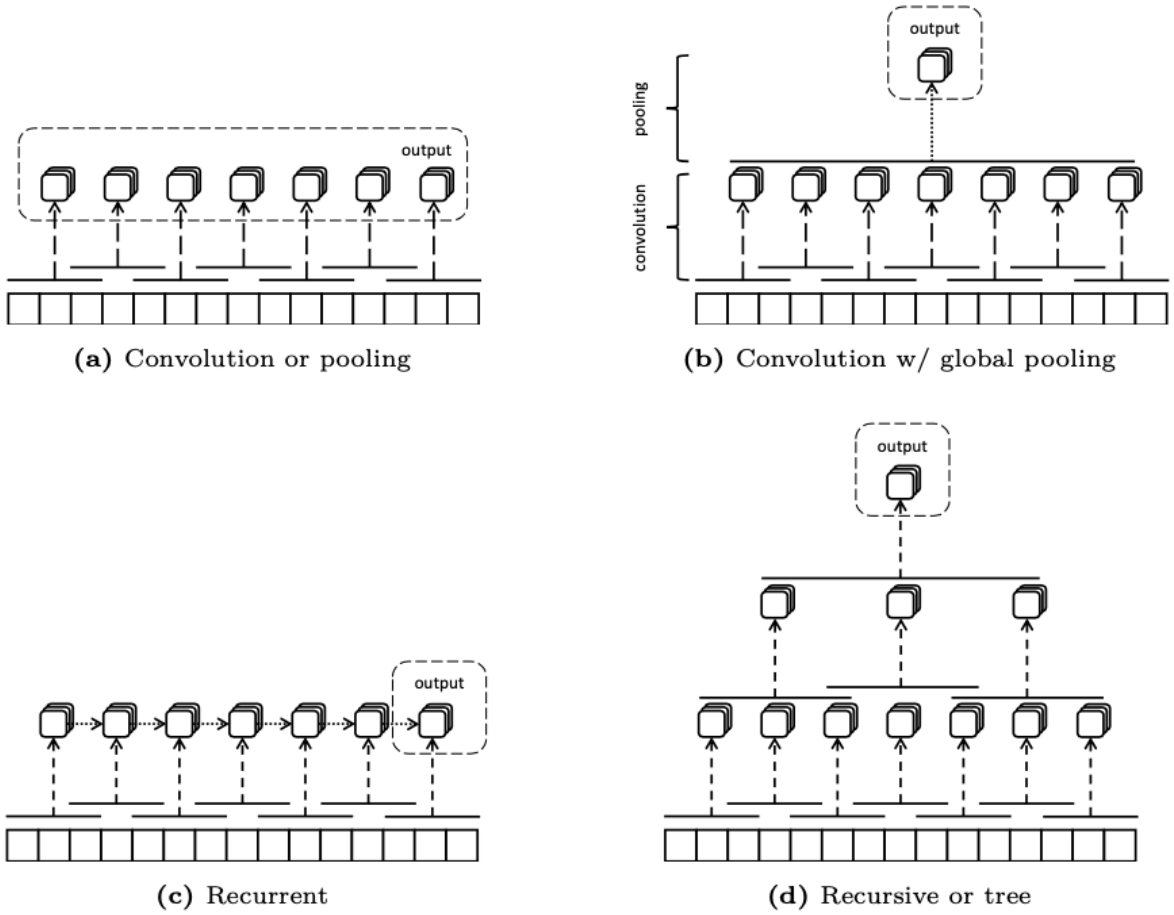


FIGURE 3.2 – Process of Semantic Search

Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are commonly used. Neural networks can be used together. CNN can be used for n-gram modeling and character embedding. It is widely used in computer vision field. CNN apply a filter with a sliding windows over input data. Values in the filter region are merged in to an output value. Filter parameters are learned during training. 1D CNN has a 1 dimensional filter and operate on 1 dimensional input. N of N-grams equals filter size. Pooling layer is commonly applied after CNN layers. There are different types of pooling. Max-pooling retains only the strongest feature per filter region. Average-pooling retains the average over each feature per filter region. Pooling does not train additional weigths. Output size depends on input size. Adaptive pooling can switch from variable length to fixed length. It defines akernel. Kernel size is a function of input size, so that result is a fixed output size. CNNs produce character n-gram representations. It can be used with or without tokenization.

RNN models global patterns in sequence data by operating words in corpus through sequences. It allows arbitrarily sized inputs. RNN model can be trained with backpropagation and gradient descent. It has ability to condition the next output on an entire words history. RNN is widely used in text data. RNN takes the ordering of elements into account. But simple RNNs forgets information from a few dozen steps ago. Long Short-Term Memory (LSTM) can help to solve this problem. LSTM introduces gated memory access. RNN including LSTM can be used as bidirectional or stacked to encode/decode input effectively. Figure 3.3 shows the processes of convolutional and recurrent neural networks (Mitra and Craswell, 2018).

Word representations are encoded in RNN layers and decoded in other RNN layers. Output is created with functions such as softmax probability via vocabulary. The fixed-



(a) Convolution or pooling

(b) Convolution w/ global pooling

(c) Recurrent

(d) Recursive or tree

FIGURE 3.3 – Neural Networks

length context vector can be a bottleneck. Attention mechanism helps to solve this problem. It allows to find relevant parts of the input. It creates a weighted average context vector. Weights are based on a softmax, sum up to 1. Attention is parameterized and trained end-to-end with the model. It provides some interpretability. It can show which words have more impact for which output. Each new decoder state produces a new context vector. The endocer states is read-only. These states are combined. Fully connected layer contains learned parameters and non-linear activations. After softmax normalization, each attention weight is multiplied with corresponding encoder state, then these are summed up for the context vector.

The input involves queries and keys of dimension d_v, and values of dimension d_v. The dot products of the query with all keys are computed, these are divided each by $\sqrt{d_k}$, and applied a softmax function to obtain the weights on the values. The attention function are computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. The matrix of outputs are computed below.

$$\text{attention}(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V \tag{3.3}$$

The two most commonly used attention functions are additive attention, and dot-product (multiplicative) attention. Dot-product attention is identical to the algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

The transformer architecture is not task specific as CNNs and RNNs. The improvement over RNN in Transformer is that it does not require recurrence. For a tensor, attention is calculated with a series of matrix multiplications over the sequence. This is an efficient way computationally. It operates on sequences of vectors. Pre-trained Transformers is popular in NLP and IR studies.

Transformers are stacked with multiple layers. In the first layer, term representations are encoded. In the second layer, contextualized representations are encoded by learning connections between terms without using any knowledge base. Transformers contextua-

lize with multi-head self-attention. Every subword token attends to every other token. Positional encodings are added to subword tokens embeddings. In each transformer layer, each vector are projected with 3 linear layer to Query, Key, Value. These projections are transformed to another multi-head dimension. Query and Key matrices are multiplied. Query and Key attention is calculated via softmax. Then, attention is multiplied by Values and return output.

BERT stands for Bidirectional Encoder Representations from Transformers. BERT has wordpiece tokenization and embedding. It covers infrequest terms in small vocabulary. It has many dimensions and layers, base version has 12 layers and 768 dimensions. Pre-training can takes weeks on 1 GPU. Special tokens are used. $[CLS]$ is classification token, used as pooling operator to get a single vector per sequence. $[MASK]$ is used in the masked language model, to predict words. $[SEP]$ is used to indicate a second sentence or additional sequence encodings. Sentence pairs are packed together into a single sequence. The sentences are differentiated in two ways. First, separatation with a special token, second, adding a learned embedding to every token indicating whether it belongs to sentence A or sentence B. BERT adds trained position embeddings and sequence embeddings. Token embeddings can be word pieces. Position embedings indicate the position of tokens. BERT consists of n layers of stacked transformers. It using LayerNorm, GelU activations. Every transformer layer receives as input the output of the previous one.

TABLE 3.1 – Sequence Embeddings (E)

Input	[CLS]	the	first	sentence	[SEP]	the	latter	one	[SEP]
Token	$E_{[CLS]}$	E_{the}	E_{first}	$E_{sentence}$	$E_{[SEP]}$	E_{the}	E_{latter}	E_{one}	$E_{[SEP]}$
Segment	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B
Position	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8

There are many BERT variants for many languages, in many domains, with different architectures. Different architectures provide larger models, more efficient outputs, longer sequences. Pre-training requires lots of configuration and computation time. Model and tokenizer can be defined with ready-to-use pre-trained models. The passage is given as input and encoded. In QA models, query and passage are given and the position containing the answer to the query is marked in the passage. Models trained for QA can be used for re-ranking. First, a candidate list is retrieved from an index. The passages in this candidate list are combined and included in the QA process so that the most relevant position is marked. In Figure 3.4, the illustrations of fine-tuning of BERT on

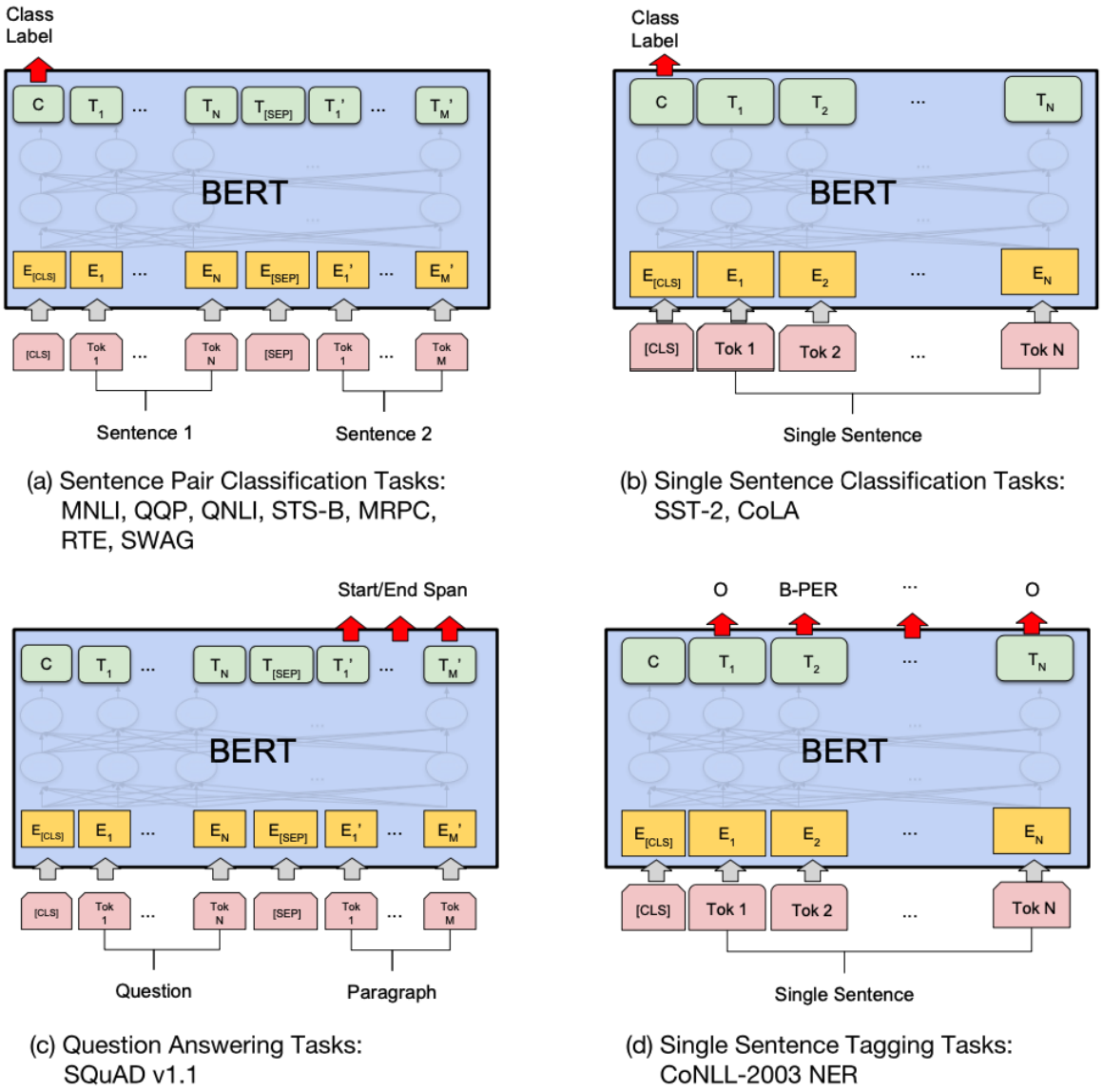


FIGURE 3.4 – BERT on Different Tasks

different tasks are showed (Devlin et al., 2019)

Bi-encoders perform self-attention over the query and candidate document separately, map them to a dense vector space, and then combine them at the end for a final representation. Therefore, bi-encoders are able to index the encoded candidates and compare these representations for each input resulting in fast prediction times.

Cross-encoders perform cross self-attention over a given input and candidate document and tend to attain much higher accuracies than their counterparts. However, it must recompute the encoding for each input and label ; as a result, they are not possible to retrieval end-to-end information cause they do not yield independent representations for the inputs and is prohibitively slow at test time.

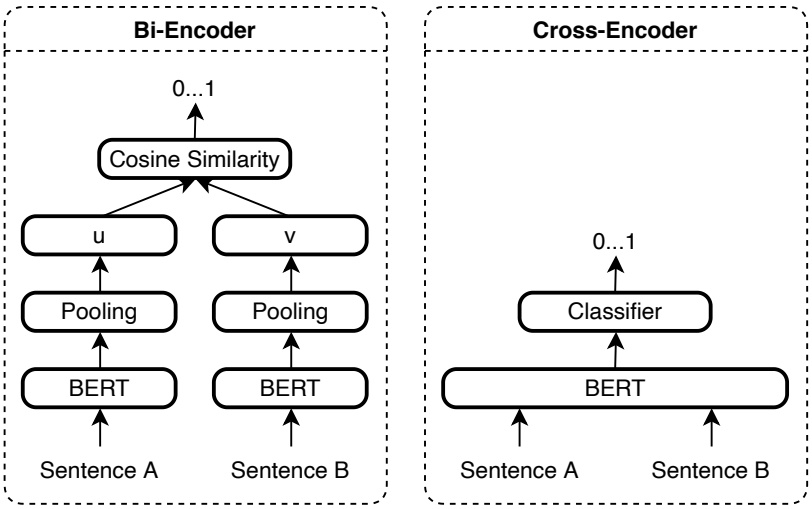


FIGURE 3.5 – Encoder Models

Bi-encoders method usually achieves lower performance compared with cross-encoders method and requires a large amount of training data. The reason is cross-encoders can compare both inputs simultaneously, while bi-encoders have to independently map inputs to a meaningful vector space which requires a sufficient amount of training examples for fine-tuning. Therefore, bi-encoders are used in dense retrieval for all documents, cross-encoders are used in re-ranking with candidate documents coming from first stage retrieval. Encoder models are shown in Figure 3.5 (Reimers and Gurevych, 2019)

3.3 Corpus

In this thesis, all datasets have been fetched from the Benchmark IR repository. The corpus and query sets are enlisted for different search and retrieval purposes. In a nutshell, there are three formats of data : queries, corpus, query links to relevant documents. In general, a typical query set contains query-id and query text information. The corpus set contains document-id, document-title, text and optional metadata. Query links to relevant documents register query-id and document-id pairs and the relevance score.

TABLE 3.2 – Query Schema

Name	Type	Description
query-id	string	Generated query ID
text	string	Query text

Table 3.3 – Corpus Schema

Name	Type	Description
document-id	string	Document's S2ORC ID
title	string	Title of article
text	string	Abstract of article
metadata	string	Authors, references, year etc.

Table 3.4 – Query-Relevance Schema

Name	Type	Description
query-id	string	Generated query ID
document-id	string	Document's S2ORC ID
score	number	Relevance grade

At a glance, we have created multi scale dataset with 4 scientific and 4 non-scientific datasets given in Table 3.5. For each dataset, total number of terms, documents, sentences and unique terms, have been computed as it is shown in Table 3.6. Moreover, average number of terms per documents, total number of queries have been calculated.

3.3.1 Scientific Documents

TREC-COVID dataset is introduced in "TREC-COVID : Constructing a Pandemic Information Retrieval Test Collection" paper that authored by Voorhees et al. TREC-COVID is a community evaluation designed to build a test collection that captures the information needs of biomedical researchers using the scientific literature during a pandemic. One of the key characteristics of pandemic search is the accelerated rate of change : the topics of interest evolve as the pandemic progresses and the scientific literature in the area explodes (Voorhees et al., 2020).

SciFact dataset is published by Allen Institute, is used in "Fact or Fiction : Verifying Scientific Claims" paper that authored by Wadden et al. SciFact study introduces task of scientific claim verification to evaluate the veracity of scientific claims against a scientific corpus. SciFact is an expert-annotated dataset of 1,409 scientific claims accompanied by a corpus of 5,183 abstracts that support or refute each claim, and annotated with rationales justifying each supports or refutes decision. This corpus is consist of randomly sample articles from a manually curated collection of journals spanning domains from basic science to clinical medicine. This corpus has mostly medical subjects such as humans, mice, animals, female, male, middle aged, adult, cell line,

signal transduction, RNA, receptors, gene expression regulation, models, risk factors, cells, cell differentiation, mutation (Wadden et al., 2020).

SciDocs dataset is published by Allen Institute, is used in the paper "SPECTER : Document-level Representation Learning using Citation-informed Transformers" that authored by Cohan et al. SciDocs dataset is larger and more diverse from other scientific datasets. The corpus allows different tasks : document classification such as Medical Subject Headings (MeSH) or Microsoft Academic Graph (MAG), citation prediction, recomendation and many others. SciDocs is consist of many articles about art, biology, business, chemistry, computer science, economics, engineering, environmental science, geography, geology, history, materials science, mathematics, medicine, philosophy, physics, political science, psychology, sociology (Cohan et al., 2020).

NFCorpus dataset is gathered from NutritionFacts.org website and is introduced by Vera Boteva et al. in the paper "A Full-Text Learning to Rank Dataset for Medical Information Retrieval". NFCorpus dataset is released for IR purposes. It consists of thousands of text queries that are linked to thousands of research articles in medical domain. Queries are taken from health topics on NutritionFacts.org website, relevance links are extracted from direct and indirect links of queries to research articles on PubMed. NFCorpus textual content is extracted from titles and descriptions of videos, blog articles, QA and topic pages (Boteva et al., 2016).

3.3.2 Non-Scientific Documents

FiQA means Financial Opinion Mining and Question Answering. FiQA dataset is gathered from microblogs, reports, news and the paper authored by Maia et al. In FiQA study, two tasks is defined to use this corpus. The first task is aspect-based financial sentiment analysist assigns that detects the target aspects which are mentioned in text and predict sentiment score for each of the mentioned targets. The latter is opinion-based question-answering over financial data. It assings answering natural language questions in financial domain. The challenging part is some questions are opinionated, targeting mined opinions and their respective entities, aspects, sentiment polarity and opinion holder. These tasks are required expertise in both of an information retrieval and a question answering perspective (Maia et al., 2018).

ArguAna dataset has argument and counterargument pairs that is extracted from de-

bates on idebate.org. A diverse corpus is published by Henning Wachsmuth et al. In ArguAna study, a large corpus is published for studying multiple counter-argument retrieval tasks, a topic-independent approach is provided to find the best counterargument to any argument. The authors claim that many counterarguments can be found without topic knowledge. ArguAna corpus has non-scientific and diverse documents in these domains : culture, digital freedoms, economy, education, environment, free speech debate, health, international, law, philosophy, politics, religion, science, society, sport (Wachsmuth et al., 2018).

Touché provides a focused collection of arguments and some socially important and controversial topics, including arguments that could help an individual form an opinion on the topic or arguments that support/challenge their existing stance. Touché is explained in Overview of Touché 2020 : Argument Retrieval authored by Bondarenko et al. (Bondarenko et al., 2020)

Quora dataset gives an opportunity to train and test models of semantic equivalence, based on actual Quora data. The dataset consists of over 400,000 lines of potential question duplicate pairs. Each line contains IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair. This dataset can be used for QA or IR purposes. Sharma et al. introduce the dataset in Natural Language Understanding with the Quora Question Pairs Dataset. (Sharma et al., 2019)

3.3.3 Dataset Analysis

The main focus of this study is to examine scientific documents. However, two non-scientific datasets are added to show averages and initial accuracy values in other datasets of the models used in experiments. 4 scientific and 4 non-scientific datasets are used. Numbers of terms, number of unique terms, number of sentences of datasets are important in indexing and retrieval processes. This data is not available in the papers. Therefore, it is developed to retrieve these figures in all datasets using NLTK.

There are two categories : scientific and non-scientific. There are 8 datasets in total. For each dataset, total number of documents, total number of terms, total number of unique terms, total number of sentences, average number of terms per sentences, average number of terms per documents, total number of queries are calculated. All

Table 3.5 – Numbers of Queries/Documents of Datasets

Category	Dataset	Number of Queries	Number of Documents
Scientific	TREC-COVID	50	171332
Scientific	SciFact	1109	5183
Scientific	SciDocs	1000	25657
Scientific	NFCorpus	3237	3633
Non-Scientific	FiQA	6648	57638
Non-Scientific	ArguAna	1406	8674
Non-Scientific	Touché	49	382545
Non-Scientific	Quora	15000	522931

dataset statistics are given in Table 3.5 and Table 3.6

Table 3.6 – Numbers of Terms of Datasets

Model	Terms	Unique Terms	Sentences	Terms per Sentence	Terms per Document
TREC-COVID	16053536	185434	1126955	14.245	93.698
SciFact	658662	32591	47121	13.978	127.081
SciDocs	2686311	71106	195663	13.729	104.700
NFCorpus	504134	24388	35130	14.350	138.765
FiQA	4187377	66790	419639	9.978	72.649
ArguAna	822246	33063	65730	12.509	94.794
Touché	60059064	367881	6143992	9.775	156.998
Quora	3631243	81308	593800	6.115	6.944

3.4 Evaluation

Feedbacks and labeled datasets can be used to improve information retrieval system. With these data, learning-to-rank methods are applied in lexical search, and models are retrained in semantic search. Evaluation metrics are required to implement these methods. Relevance of documents in resultset can be binary or graded. Evaluation metric should be chosen accordingly. Basic evaluation metrics precision and recall can be used. However, in information retrieval systems, only the number of relevant documents in resultset is not sufficient. It is also important to have relevant documents at the top. Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) are more commonly used, where ranking is involved in calculation. NDCG supports graded relevance while MAP uses binary relevance values.

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances. Recall (also called sensitivity) is the fraction of rele-

vant instances that were retrieved. Both precision and recall are therefore based on relevance. Precision asks how many selected documents are relevant. Recall asks how many relevant documents are selected.

$$\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \quad (3.4)$$

$$\text{Recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \quad (3.5)$$

Precision is equal to the number of related documents retrieved divided by the number of retrieved documents. Recall is equal to the number of retrieved related documents divided by the total number of related documents.

Mean Average Precision for a set of queries is mean of average precision scores for each query. In this calculation, it becomes important to have relevant documents at the top. @k notation is used for top k results.

$$\text{MAP(Q)} = \frac{1}{|Q|} \cdot \sum_{q \in Q} \frac{\sum_{i=1}^{k} P(q)_{@i} \cdot rel(q)_i}{|rel(q)|} \quad (3.6)$$

Q is query set, $P(q)_{@i}$ equals precision of query q after first i documents. $rel(q)_i$ equals binary relevance of document at position i. $|Q|$ and $|rel(q)|$ are shows numbers of queries and relevant documents. MAP is the mean of average precision over all the queries.

Normalized Discounted Cumulative Gain allows to use of grades. @10 is commonly used as evaluation metric. Grades of documents are divided into order of documents and DCG is calculated with their sum. Then DCG values in all queries are divided by sorted DCG values. Logarithm values are calculated. Usually four levels of perfectly relevant, highly relevant, relevant, irrelevant grades are used.

$$\text{DCG(Q)} = \sum_{d \in D, i=1} \frac{rel(d)}{log_2(i+1)} \quad (3.7)$$

$$\text{NDCG(Q)} = \frac{1}{|Q|} \cdot \sum_{q \in Q} \frac{DCG(q)}{DCG(sorted(rel(q)))} \quad (3.8)$$

Q and D are query and its result document sets, $|Q|$ shows number of queries. $rel(d)$ equals relevance grade for single query-document pair, and $rel(q)$ is list of all relevance grades for a query. $sorted()$ function returns graded documents by descending relevance.

4 RESULTS

Neural re-ranking approaches use the output of a first-stage retrieval system, often BM25, and re-ranks the documents to create a better comparison of the retrieved documents. Cross-encoder models based on pre-trained Sentence Transformers has shown strong performance.

Dense retrievers map queries and documents in a shared, dense vector space. This allowed the document representation to be pre-computed and indexed. Bi-encoder models based on pre-trained Transformers can be used.

In sparse retrieval, similarity scores between the non-contextualized query embeddings with the contextualized document embeddings are computed. These scores can be pre-computed for a given document, which results in a 30k dimensional sparse vector. It has a huge computational cost.

4.1 Experiments

16 different neural models are used in the experiments. Base models, training datasets, pooling strategies, similarity functions can be different model by model. Batch size equals 128 for each experiment. 12 models are used as bi-encoder for dense retrieval method, 3 models are used as cross-encoder for re-ranking method, and 1 model is used for sparse retrieval method.

4.1.1 Base Models

A large number of models with Transformer architecture are developed. It can be task-specific or general purpose. Many operations such as the number of attention heads, the number of hidden layers, tokenization method, sequence generation method, pooling strategy can be different. Model variants are derived from base models with fine-tuning and training. 6 base models are used in the experiments : MPNet, MiniLM, BERT, DistilBERT, RoBERTa, ELECTRA.

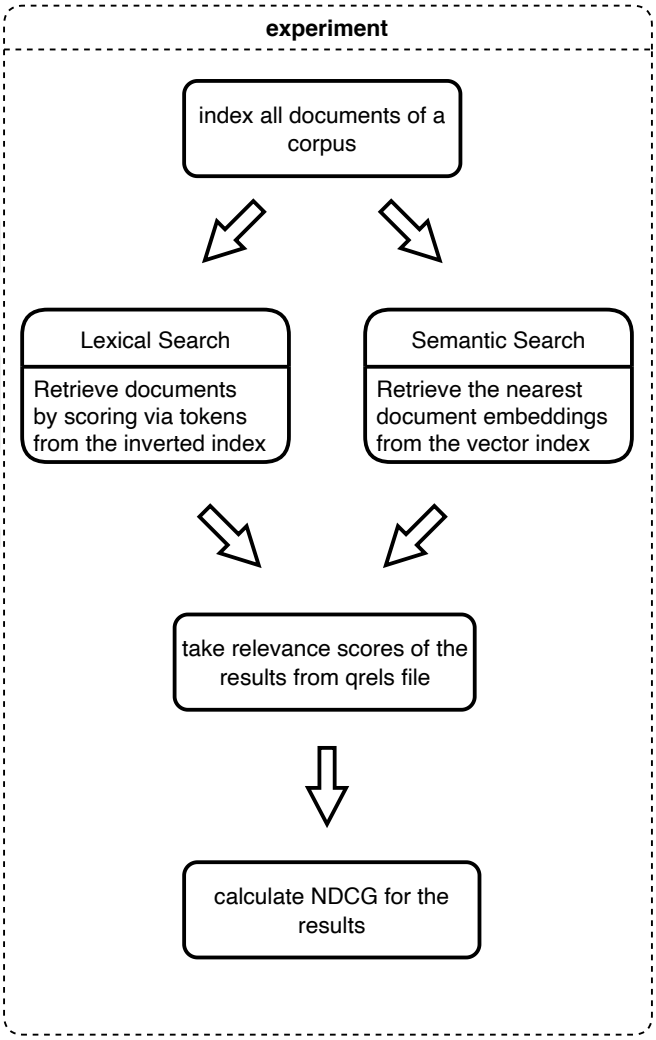


FIGURE 4.1 – Process of Experiment

BERT is a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus. DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark (Sanh et al., 2020). RoBERTa builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates (Liu et al., 2019).

MPNet adopts a novel pre-training method, named masked and permuted language modeling, to inherit the advantages of masked language modeling and permuted language modeling for natural language understanding (Song et al., 2020).

MiniLM has simple and effective approach to compress large Transformer based pre-trained models, termed as deep self-attention distillation. The small model (also called student) is trained by deeply mimicking the self-attention module of the large model (also called teacher). In this model, distillation is applied for the self-attention module of the last Transformer layer of the teacher (Wang et al., 2020).

ELECTRA is a new pretraining approach which trains two transformer models : the generator and the discriminator. The generator's role is to replace tokens in a sequence, and is therefore trained as a masked language model. The discriminator, which is the model we're interested in, tries to identify which tokens were replaced by the generator in the sequence (Clark et al., 2020).

4.1.2 Training Datasets

Variety of datasets in which the models are trained prevents model from being biased. Having a large corpus of datasets also helps to reduce the problem of out of vocabulary. Training datasets contain query and document tuples. In the experiments, models are elected with 4 different dataset groups : ALL, MULTI-QA, MS-MARCO, Specter.

Specter dataset contains scientific texts. It has about 684K training tuples (Cohan et al., 2020).

MS-MARCO (Microsoft Machine Reading Comprehension) is a large scale dataset focused on machine reading comprehension, question answering, and passage ranking.

There are different sizes of dataset. Small version is used in selected models. It has 532,761 training tuples (Bajaj et al., 2018).

MULTI-QA is collected for QA purpose. It is transformed as suitable for IR tasks. It contains many different QA datasets such as WikiAnswers, Stack Exchange, MS-Marco, Amazon QA, Yahoo Answers etc. It has 214,988,242 training tuples.

ALL contains both QA and IR datasets such as Reddit comments, PAQ, S2ORC, Code Search, COCO Image Captions, Specter, SearchQA, Flickr Wikipedia, SQuAD etc. It has 1,170,060,424 training tuples.

4.1.3 Pooling and Scoring

Pooling operation allows to derive a fixed sized sentence embedding. There are three main pooling strategies : Using the output of the CLS-token, computing the mean of all output vectors (MEAN-strategy), and computing a max-over-time of the output vectors (MAX-strategy). For a sentence 5 tokens long with CLS-token, each token in the input sentence is embedded in a tensor and is represented in a vector space. After the pooling operation, sequence dimension is squashed, represents a pooled embedding of the input sequence.

In Lexical search, the terms in the query and the documents in the index are scored with the BM25 function. In semantic search, embeddings are represented as vectors. Cosine similarity or dot product is commonly used to find similarity between vectors. Cosine similarity cares about angle difference, while dot product cares about angle and magnitude. If normalization is applied to have the same magnitude, the two are indistinguishable. If magnitude plays a role, dot product would be better as a similarity measure.

The dot product of query (Q) and document (D) is defined as follows :

$$\vec{Q} \cdot \vec{D} = \|Q\|\|D\| \cos\theta \tag{4.1}$$

The cosine similarity for comparing the query and document vectors :

$$\cos(\vec{Q}, \vec{D}) = \frac{Q \cdot D}{\|Q\|\|D\|} = \frac{\sum_{i=1}^{n} Q_i D_i}{\sqrt{\sum_{i=1}^{n} (Q_i)^2}\sqrt{\sum_{i=1}^{n} (D_i)^2}} \tag{4.2}$$

4.2 Evaluation Metrics

To measure information retrieval effectiveness in standard way, a test collection consisting of three things is needed. A document collection, query collection and a set of relevance judgments. Relevance judgments may be binary or graded assessments.

Two of the scientific datasets used in this study, TREC-COVID and NFCorpus, have 3-level graded relevance judgments. SciFact, SciDocs, and non-scientific datasets have binary relevance judgments.

In a search engine, users want to see related documents at the top. Therefore, in information retrieval, ranking is as important as relevance. Even though the precision and recall metrics are widely used in other domains, they are not effective in information retrieval. Instead, MAP and NDCG metrics are preferred, which also use rankings in the result documents.

For the TREC-COVID dataset, evaluation results are given for the top 10-100-1000 results obtained using the all-mpnet-base-v2 model in dense retrieval method. Unlike Precision and recall, MAP and NDCG do not depend on the number of retrieved results. Regardless of the number of results, the relevancy or accuracy value increases if the relevant documents are at the top.

NDCG is calculated with graded relevance judgments. Therefore, it is an accurate and fair evaluation metric if more relevant results are at the top. For example, in the results of the TREC-COVID dataset, it can be seen that more relevant documents are not on the top in the first 100 results, while more relevant results are on top for the Top 10 results. MAP, on the other hand, shows the first 100 results as better than the first 10 results because it does not use grade. The different evaluation values are shown in Figure 4.1 and Figure 4.2. In a search engine, the number k is given as 10 because users usually want to find what they are looking for in the first place. NDCG@10 was used for the measurement of all methods for all datasets.

Table 4.1 – Evaluation Scores of Top k Results

Metric	10	100	1000
Precision	0.556	0.433	0.199
Recall	0.014	0.105	0.424
MAP	0.012	0.070	0.197
NDCG	**0.513**	**0.417**	**0.432**

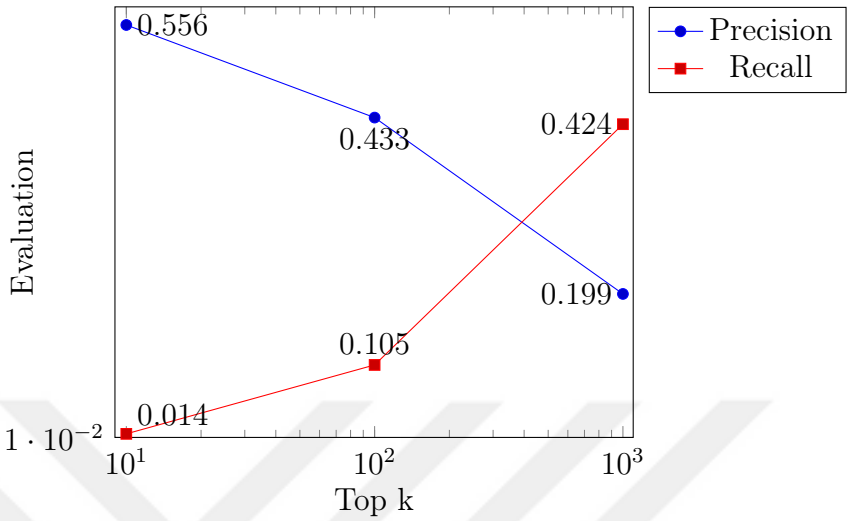


FIGURE 4.2 – Precision and Recall Metrics

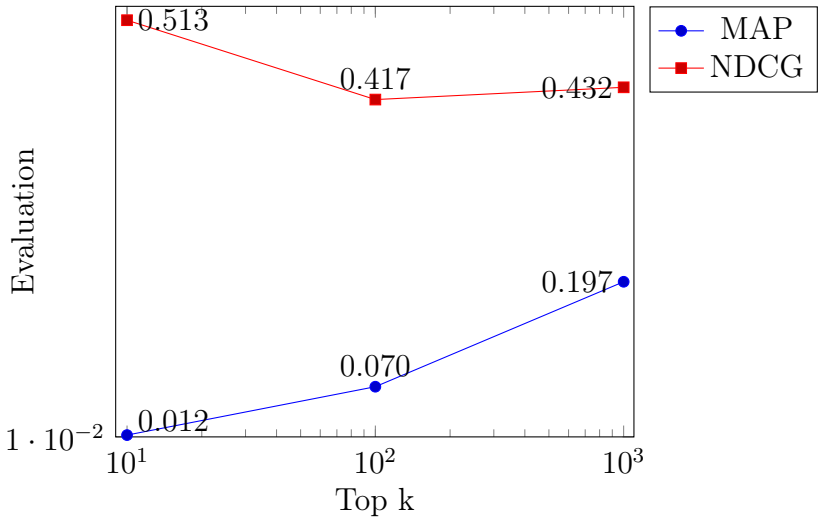


FIGURE 4.3 – NDCG and MAP Metrics

4.3 Implementation

There are 4 different steps for all experiments. Each experiment runs separately for each corpus, start to end. Firstly, index all documents for a corpus, then retrieve the result document set for all queries. For these two steps, Elasticsearch was used in the lexical method, Faiss was used in the semantic methods. Then, relevance scores of the results are taken from qrels file. Finally, NDCG and MAP values are calculated for the results. This process is shown in Figure 4.3.

Main development language is Python 3.7 for all experiments in the thesis. In dataset analysis, Natural Language Toolkit (NLTK) 3.2.5 and Pandas 1.1.5 were used. Elasticsearch 7.13.4 was used in lexical search experiments . Benchmark IR 0.2.3 (including Huggingface Transformers 4.12.5, Faiss 1.7.1) was used for semantic search methods. All experiments were run on Google CoLab Pro servers. Hardware specifications are 26G memory, 125G harddisk and GPU support with Tesla P100-PCIE-16GB, NVIDIA-SMI 495.44, driver version : 460.32.03, CUDA version : 11.2

TREC-COVID dataset is provided under publisher-specific licenses, declared in the thesis. SciFact is provided under the CC BY-NC 2.0 license. SCIDOCS is provided under the GNU General Public License v3.0 license. NFCorpus, FiQA, Quora do not report the dataset license in the thesis or a repository. ArguAna, Touché are provided under the CC BY 4.0 license. Neural models were fetched from HuggingFace (https ://huggingface.co). The neural models were adapted by using Python programming language. Implementation is referenced in Appendix A.1. All experiments are publicly available at the project repository (https ://github.com/alpers/semantic-search).

4.4 Summary

Table 4.2 contains the results of all three semantic search methods : dense retrieval, re-ranking, sparse retrieval with the NDCG@10 evaluation metric. Models are divided by four different training datasets in training phase : all, multi-qa, ms-marco, specter. Afterwards different pooling strategies (CLS, MEAN) and similarity functions (cosine similarity, dot product) are selected. Totally, 16 different neural models are used.

Sparse retrieval does not seem efficient in production due to disk size and computation

cost. The models that performed better in re-ranking and dense retrieval methods are marked in bold. **ms-marco-MiniLM-L-6-v2** as cross-encoder and **all-mpnet-base-v2** as bi-encoder are selected.

As a result, ranking success increased in all datasets in the re-ranking method. As seen in Table 4.3, in 4 scientific datasets, results are increased 10%, 1‰, 6‰, and 6% respectively. 4% on average. In dense retrieval, 3 out of 4 results have decreased, unfortunately. It has better results in only one scientific dataset. Between the models used in dense retrieval, pooling and scoring are compared in the same base models. Mean pooling has performed better on 3 of 4 scientific datasets. Cosine similarity performed better for the same base models. Looking at the training datasets, the models trained with more and diverse datasets performed better by far. For base models, MPNet and MiniLM seem more successful than others.

TABLE 4.2 – NDCG@10 Scores for Neural Models

Method	Model	Pooling Scoring	Scientific				Others			
			Trec-Covid	SciFact	SciDocs	NFCorpus	FiQA	ArguAna	Touché	Quora
Dense	**all-mpnet-base-v2**	mean/dot	0.513	**0.655**	**0.237**	**0.332**	**0.499**	0.465	0.199	0.874
	all-distilroberta-v1	mean/dot	0.528	0.631	0.216	0.292	0.394	0.479	0.195	0.867
	all-MiniLM-L12-v2	mean/dot	0.508	0.626	0.218	0.322	0.372	0.471	0.172	0.877
	multi-qa-mpnet-base-dot-v1	cls/dot	0.618	0.589	0.166	0.318	0.487	0.503	0.198	0.874
	multi-qa-MiniLM-L6-cos-v1	mean/cos	0.558	0.540	0.157	0.296	0.363	0.491	0.156	0.872
	multi-qa-distilbert-dot-v1	cls/dot	0.693	0.548	0.157	0.313	0.432	0.410	0.236	**0.888**
	multi-qa-distilbert-cos-v1	mean/cos	0.563	0.595	0.160	0.302	0.400	**0.510**	0.196	0.879
	msmarco-distilbert-base-tas-b	cls/dot	0.481	0.642	0.148	0.318	0.300	0.427	0.162	0.835
	msmarco-distilbert-dot-v5	mean/dot	0.663	0.594	0.140	0.298	0.286	0.348	0.224	0.832
	msmarco-bert-co-condensor	cls/dot	**0.726**	0.600	0.139	0.318	0.285	0.379	0.189	0.863
	msmarco-roberta-base-ance	cls/dot	0.653	0.511	0.121	0.235	0.294	0.418	**0.239**	0.852
	allenai-specter	cls/dot	0.358	0.506	0.142	0.185	0.061	0.320	0.102	0.639
Re-ranking	**ms-marco-MiniLM-L-6-v2**	-/cos	**0.757**	**0.686**	**0.165**	**0.365**	0.384	0.416	0.270	**0.830**
	ms-marco-TinyBERT-L-2-v2	-/cos	0.728	0.663	0.152	0.352	0.333	**0.418**	0.251	0.802
	ms-marco-electra-base	-/cos	0.697	0.673	0.153	0.344	**0.386**	0.400	**0.276**	0.783
Sparse	sparta-distilbert-base-v1	mean/cos	0.606	0.666	0.058	0.353	0.355	0.352	-	-

TABLE 4.3 – Lexical vs. Re-ranking vs. Dense Retrieval

Dataset	**Lexical** (BM25)	**Re-ranking** (BM25+Cross-Encoder)	**Dense Retrieval** (Bi-Encoder)
TREC-COVID	0.688	**0.757**	0.513
SciFact	0.685	**0.686**	0.655
SciDocs	0.164	0.165	**0.237**
NFCorpus	0.343	**0.365**	0.332
FiQA	0.253	0.348	**0.499**
ArguAna	**0.471**	0.416	0.465
Touché	**0.347**	0.270	0.199
Quora	0.807	0.830	**0.874**

5 DISCUSSION

In current information search-retrieval tasks, lexical search is mostly preferred due to its time complexity, hardware costs and the performance of relevance scores between query-document pairs. The main challenge in semantic search methods is to prepare a benchmark dataset in a broad perspective. Since scientific documents are well-formatted, data-related problems have been largely resolved. In this way, we have focused on the comparative analysis of semantic search methods with different neural models.

As a result, it was seen that semantic search has shown better results than lexical search. Neural models were differentiated into training dataset groups, base models, pooling strategy, and similarity function. The sparse retrieval method was not efficient due to disk size requirements and computational cost. Models that perform better in re-ranking and dense retrieval methods were selected and compared with lexical search.

In the re-ranking method, the ranking accuracy of all data sets increased by an average of 4 percent. In dense retrieval, the accuracy rate decreased in 3 out of 4 data sets. As a result, it is seen that semantic search performs better than lexical search in the re-ranking method.

6 CONCLUSION

New generation information retrieval procedures provide complex tools to remodel the design of search engines. Even though semantic analysis is gradually adopted by corporations, complex behavior of knowledge behind the information entails subsequent data learning models. Text models are currently in use through lexical features. Search engines with lexical methods lack contextual and semantic information. This barrier has been overcome with the development of deep learning methods. More accurate results can be retrieved by obtaining contextual information of different types of content such as text, image, video with neural models. In this thesis, a broad perspective of search engines was considered through lexical and semantic features. Semantic search methods were experimented then compared with lexical methods in data sets consisting of scientific documents.

The semantic information retrieval considers the contextual basis. Neural models are capable of finding contextual information. The embeddings are generated for both queries and documents via neural models. Finding more relevant document for query depends on semantic understanding of query and texts. It is necessary to define position of each text. This can be demonstrated by vector representation. In word embeddings, vector representations of words are provided.

Even if we note that our results are promising, there are shortcomings in the study. The results could be examined by grouping scientific documents by domain-specific or general. By detailing the data analysis, the data-centric success of the methods and models could be determined. All datasets contain English documents. The neural models used in the experiments do not perform well in others languages. To support other languages, multilingual neural models can be used.

In the next stages, the neural model comparison results can be used in model pre-training and model-tuning studies for IR purposes. It can be a guide for those who want to use semantic search in production.

Our future steps of semantic search-retrieval tasks will focus on the creation of a new generalized non-scientific dataset where the current lexical and semantical methods will

be dealt with. Furthermore, a multilingual framework will be also added to the current scheme.

6.1 Publications

— Our study was presented in RDCONF (International Conference on Design, Research and Development) on December 15-18, 2021. Our paper was published in EJOSAT (European Journal of Science and Technology).

— The paper is available at https ://doi.org/10.31590/ejosat.1043441

REFERENCES

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S. and Wang, T. (2018). Ms marco : A human generated machine reading comprehension dataset.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017). Enriching word vectors with subword information.

Bondarenko, A., Fröbe, M., Beloucif, M., Gienapp, L., Ajjour, Y., Panchenko, A., Biemann, C., Stein, B., Wachsmuth, H., Potthast, M. and Hagen, M. (2020). Overview of touché 2020 : Argument retrieval, pp. 384–395.

Boteva, V., Gholipour, D., Sokolov, A. and Riezler, S. (2016). A full-text learning to rank dataset for medical information retrieval.
URL: *http ://www.cl.uni-heidelberg.de/ riezler/publications/papers/ECIR2016.pdf*

Clark, K., Luong, M.-T., Le, Q. V. and Manning, C. D. (2020). Electra : Pre-training text encoders as discriminators rather than generators.

Cohan, A., Feldman, S., Beltagy, I., Downey, D. and Weld, D. S. (2020). Specter : Document-level representation learning using citation-informed transformers.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Bert : Pre-training of deep bidirectional transformers for language understanding.

Hofstätter, S., Althammer, S., Schröder, M., Sertkan, M. and Hanbury, A. (2021). Improving efficient neural ranking models with cross-architecture knowledge distillation.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). Roberta : A robustly optimized bert pretraining approach.

Macdonald, C. and Tonellotto, N. (2020). Declarative experimentation in information retrieval using pyterrier, p. 161–168.
URL: *https ://doi.org/10.1145/3409256.3409829*

Maia, M., Handschuh, S., Freitas, A., Davis, B., McDermott, R., Zarrouk, M. and Balahur, A. (2018). Www'18 open challenge : Financial opinion mining and question answering, *Companion Proceedings of the The Web Conference 2018*, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, p. 1941–1942.
URL: *https ://doi.org/10.1145/3184558.3192301*

Mitra, B. and Craswell, N. (2018). An introduction to neural information retrieval, *Foundations and Trends® in Information Retrieval* **13**(1) : 1–126.
URL: *http ://dx.doi.org/10.1561/1500000061*

Reimers, N. and Gurevych, I. (2019). Sentence-bert : Sentence embeddings using siamese bert-networks.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework : Bm25 and beyond, *Foundations and Trends in Information Retrieval* **3** : 333–389.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2020). Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter.

Sharma, L., Graesser, L., Nangia, N. and Evci, U. (2019). Natural language understanding with the quora question pairs dataset.

Song, K., Tan, X., Qin, T., Lu, J. and Liu, T.-Y. (2020). Mpnet : Masked and permuted pre-training for language understanding.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A. and Gurevych, I. (2021). Beir : A heterogenous benchmark for zero-shot evaluation of information retrieval models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). Attention is all you need.

Voorhees, E., Alam, T., Bedrick, S., Demner-Fushman, D., Hersh, W. R., Lo, K., Roberts, K., Soboroff, I. and Wang, L. L. (2020). Trec-covid : Constructing a pandemic information retrieval test collection.

Wachsmuth, H., Syed, S. and Stein, B. (2018). Retrieval of the best counterargument without prior topic knowledge, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, Association for Computational Linguistics, Melbourne, Australia, pp. 241–251.
URL: *https ://aclanthology.org/P18-102*

Wadden, D., Lin, S., Lo, K., Wang, L. L., van Zuylen, M., Cohan, A. and Hajishirzi, H. (2020). Fact or fiction : Verifying scientific claims.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. and Zhou, M. (2020). Minilm : Deep self-attention distillation for task-agnostic compression of pre-trained transformers.

APPENDIX A EXPERIMENTS

All code files are attached to the **semantic-search.zip** file.

1. Install Python 3 packages

   `install_packages.sh`

2. Install Elasticsearch

   `install_elasticsearch.sh`

3. Run experiments by using Python scripts:
   - `0_dataset_analysis.py`
   - `1_lexical_reranking.py`
   - `2_sparse_retrieval.py`
   - `3_dense_retrieval.py`

BIOGRAPHICAL SKETCH

Alper MITINCIK has a B.S. degree in Computer Engineering from Yildiz Technical University. He has started his professional career as a software engineer after graduating. In 2018, he has begun master's studies at Galatasaray University and has been working on information retrieval and natural language processing until today.