AN INFORMATION RETRIEVAL MODEL BASED ON INTERACTION FEATURES AND NEURAL NETWORKS

FADEL ALHASSAN

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

AN INFORMATION RETRIEVAL MODEL BASED ON INTERACTION FEATURES AND NEURAL NETWORKS

FADEL ALHASSAN

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

ABSTRACT

As the state-of-the-art for ad-hoc retrieval, the interaction-based approach represents the interaction between the query and the document through the semantic similarities of their words. The constructed interaction structure is then passed into a deep learning model for feature extraction which in turn are passed into another deep learning model for textual documents ranking.

As far as we know, no study has yet identified how relevance matches may appear in the interaction structure and what features reflect that matches. Instead, the majority of the proposed models are based on the hypothesis that relevance matches are following some fixed visual patterns in the interaction matrix. Therefore, most of them are utilizing deep learning techniques for visual pattern recognition for features extraction. This features extraction approach affects the proposed models' performance and simplicity.

This work starts with an analytical study to identify a set of features called the interaction features which reflect how relevance matches may appear in the interaction matrix. Accordingly, a new approach for features extraction and documents ranking is proposed.

Interestingly, the study found that the interaction features do not follow any specific visual pattern and therefore it suggests that deep learning techniques are not the most effective approach for the feature extraction task. Instead, a set of manually designed functions are proposed and a shallow neural ranking model was developed.

The experiments results confirm the previous finding and show that, though less complex and more efficient, our model was able to outperform two baselines and give a close performance to the state-of-the-art model even without using some important IR factors like term importance.

ABSTRAK

Status semasa berkaitan *Capaian adhoc* masa kini, di mana kaedah berasaskan interaksi telah mewakilkan interaksi di antara pertanyaan dan dokumen melalui persamaan semantik perkataan mereka. Struktur interaksi yang telah dibina ini akan disalurkan ke dalam model pembelajaran mendalam (MPM) untuk pengekstrakan ciri, yang seterusnya akan disalurkan pula ke dalam MPM yang lain untuk menentukan tahap kedudukan dokumen teks tersebut.

Sehingga kini, tiada lagi kajian yang telah mengenalpasti apakah ciri-ciri interaksi dan bagaimana mereka boleh berada dalam struktur interaksi tersebut. Sebaliknya, kebanyakan model yang dicadangkan, hanya menganggap yang ciri-ciri interaksi berkaitan adalah mengikuti beberapa corak visual yang tetap. Oleh itu, mereka akan menggunakan teknik-teknik MPM untuk pengecaman pola visual bagi pengekstrakan ciri. Andaian ini mengakibatkan prestasi model menjadi terjejas dan rumit.

Kajian ini adalah berbeza kerana ianya bermula dengan kajian analisa bagi mengenalpasti ciri-ciri interaksi yang paling penting dan bagaimana ianya boleh berada di dalam matriks interaksi yang berkaitan. Berdasarkan analisis tersebut, barulah cadangan kaedah baru untuk pengekstrakan ciri dan tahap kedudukannya diusulkan.

Menariknya, kajian ini mendapati bahawa ciri-ciri interaksi sebenarnya tidak mengikuti apa-apa pola visual yang tertentu. Oleh itu, ia mencadangkan bahawa teknik-teknik MPM tidak semestinya menjadi kaedah paling berkesan untuk tugasan pengekstrakan ciri-ciri. Sebaliknya, satu set fungsi buatan-tangan yang mudah adalah dicadangkan bagi tugasan pengekstrakan ciri dan model neural tahap kedudukan yang mudah telahpun dibangunkan.

Keputusan eksperimen yang dibandingkan dengan hasil keputusan terdahulu, mengesahkan bahawa model neural yang mudah dengan satu set fungsi buatan-tangan bagi pengekstrakan ciri-cirinya, telah pun dapat mengatasi prestasi keputusan dua model-asas yang amat kukuh. Ianya juga telah menghampiri prestasi kesemua model-model masa kini walaupun tanpa menggunakan faktor pengambilan maklumat yang penting seperti kepentingan terma.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to Dr. SRI DEVI RAVANA and Dr. ROHANA BINTI MAHMUD, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work.

Also, I wish to thank my wife Hadia for her endless support and encouragement throughout my study.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

With the dramatical increase of information size on the web, information retrieval applications become a gate for any person or service seeking this information. Typically, for retrieving certain information, the user should articulate his need as a sequence of query terms, thereby the information retrieval application understands the user's needs and returns the relevant documents.

Expressing information needs as a sequence of query terms is not always an easy task especially when the user is not familiar with domain terminology. Most of the traditional ad-hoc retrieval approaches like TF-IDF and BM25 (Robertson & Zaragoza, 2010) depend on the exact matching between the query terms and documents words which fails to retrieve relevant documents where few or no terms from the query are found. Even though models like Latent Space Analysis (LSA) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) and Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) were proposed to alleviate the previous problem by focusing more on the semantic similarity between the query and the document, those models are still not able to outperform the traditional models because they fail to preserve and utilize positional information.

Based on the previous argument, it is essential for any newly proposed information retrieval model to find an effective way to reflect and capture both the exact and the semantic match. Furthermore, it is necessary for any modern IR model to preserve positional information and use them to measure other important IR factors such as query coverage and term proximity.

In the last years, deep learning has led to a dramatic improvement in computer vision, speech recognition, and machine translation. Likewise, it is expected that deep learning will gradually be able to outperform traditional approaches in information retrieval and

that it may lead to a new breakthrough by providing a new deep learning model for information retrieval which is able to meet all the required factors (Mitra & Craswell, 2017a).

Driven by those expectations, multiple works have been proposed to use deep learning approaches for building a new information retrieval model. Some of these works focus on the semantic match while others focus on the combination between the exact and the semantic match. However, at the early stages, most of these models were not able to outperform the traditional IR models. In fact, most of these models were obtained from different domains (like computer vision, speech recognition, and NLP) and thus they were designed to solve different problems.

Recently, a number of more mature neural IR models were proposed and for the first time they were able to outperform the traditional IR approaches and to gain the state-of-the-art performance in the ad-hoc retrieval task (Guo, Fan, Ai, & Croft, 2016; Hui, Yates, Berberich, & de Melo, 2018; McDonald, Brokos, & Androutsopoulos, 2018; Pang et al., 2017a). The majority of these models were based on a new approach, called the interaction based approach, which introduces the interaction between the query and the document as a matrix of the semantic similarities of all corresponding query terms and document words (Pang, Lan, Guo, Xu, & Cheng, 2016). The interaction matrix is a rich representation structure which is able to reflect the exact and semantic matches and preserve all positional information. Hence, it is expected that by implementing a suitable deep learning model that takes the interaction structure as input, it is possible to capture and integrate many of the required factors to build an effective IR model.

The majority of the proposed deep learning models for the interaction approach are depending on visual patterns recognition techniques for implicitly extracting the required features from the interaction structure (Hui et al., 2018; McDonald et al., 2018; Pang et al., 2017a; Pang, Lan, Guo, Xu, & Cheng, 2016). However, there is a lack of studies that

identified the required features or provide tangible evidence that the visual pattern recognition techniques are suitable for capturing relevance matches. More important, since most of the proposed models are originally designed for computer vision tasks, they are either recognition or classification models whereas ad-hoc retrieval task is a combination of recognition and ranking. This lack of harmony between the IR requirements and the proposed models affects proposed models' performance and simplicity. More detailed information on this is covered in Chapter 2.

1.1 Problem Statement

Even though the interaction approach outperforms traditional IR models and gains the state-of-the-art performance for the neural ad-hoc retrieval, there is no clear understanding of, how relevance matches may appear in the interaction matrix, what features that we are looking for in order to measure relevance matches, and what is the most effective way to extract these features. Consequently, there is still a gap between IR factors, as a base for assessing relevance matches, and the proposed interaction-based models which significantly affects both models' performance and simplicity. This gap can be observed in the most recent interaction-based models in two places: the interaction structure and the feature extraction approaches.

In fact, in order to get better performance, some works suggested embedding other structure into the interaction matrix while others chose to use more sophisticated deep learning models for feature extraction. However, both approaches led to heavy and complex models that require lots of computational resources and that are not amenable for analysis or interpretation. More important, the fact that the proposed approaches for performance optimization are not based on robust analysis of the relationship between the required IR factors and the interaction structure indicates the possibility of getting equal or even better performance with less complex and more efficient approaches.

1.2 Research Objectives

The main purpose of this study is to identify the minimum set of features, called interaction features (see Section 4.2.6), required to asses relevance matches by reflecting how a set of IR factors appears in the interaction matrix and to build a new deep learning model for ad-hoc retrieval task based on these features.

To this end, some objectives can be formulated:

1. To identify the minimum set of the features that represent a set of selected IR factors in the interaction matrix.
2. To propose an effective way to extract the identified features from the interaction matrix.
3. To build an effective neural ranking model based on the extracted features.
4. To evaluate the proposed model in term of performance, efficiency, and simplicity on the ad-hoc retrieval task and to analyze the impact of the model's components.

1.3 Study Scope

This study only concerns with the ad-hoc retrieval task which retrieves the relevant documents from a predefined documents collection for a previously unseen query without using any query log or any form of user interaction.

Two criteria were used for selecting the set of IR factors that will be considered in this study (See Section 2.6):

1. Consider factors that are commonly used by different models.
2. Consider factors that do not need any structure or resources other than the interaction matrix.

Accordingly, the following four information retrieval factors are taken into account in the study in order to identify the required features and to design the ranking model (For more details on each factor see Section 2.1):

1. Exact match
2. Semantic Match
3. Query Coverage
4. Term proximity

The study follows the interaction approach where the semantic similarities between all query terms and document words are computed and stored in a structure called the interaction matrix and introduced as model input.

1.4 The significance of the Study

Since deep learning was able to dominate classical models in computer vision, NLP and speech recognition, most of the available studies in the field of neural IR were motivated by the expectation that deep learning will dominate IR domain as well. In fact, most of the available studies were competing for achieving better performance in ad-hoc retrieval depending on the available deep learning models and techniques at the expense of efficiency and simplicity. Consequently, this race led to, in most cases, heavy models which need lots of computational resources. In the same time, most of these models have complex structures which limit their contribution for growing our understanding of the relationship between deep learning and information retrieval.

The finding of this study would establish for a more simple and interpretable approach that is based on the analysis of the relation between the interaction matrix and the selected set of IR factors.

Furthermore, the findings of the study will allow for building an efficient yet effective neural IR model which is essential for incorporating neural IR models in real life IR applications.

1.5 Thesis Structure

In addition to the introduction, the thesis has five additional chapters. The next chapter reviews the utilization of deep learning techniques for the ad-hoc retrieval task. Chapter 3 and 4 detail the research method and explain the proposed methods, models, and techniques for achieving the study objectives. Chapter 5 lists the results of the experiments, contrast them with other baseline models and discuss the finding. Finally, Chapter 6 concludes the study and suggests future works.

Chapter 2: Literature Review

Deep learning has gained state-of-the-art performance in different domains like computer vision, voice recognition, and NLP. It is expected that over the next couples of years deep learning will be able to outperform traditional approaches in the field of information retrieval and gain the state-of-the-art performance as well (Zhang et al., 2016). However, despite large research efforts, some information retrieval experts start to show some doubts that deep learning models may not be even suitable for some IR tasks like ad-hoc retrieval (Zhang et al., 2016).

This review aims at exploring the most efficient deep learning models that have been designed for ad-hoc retrieval task and analyzing these models in order to find possible drawbacks and limitations. Furthermore, the review points out the underlining gaps that cause these problems which help in understanding the current state of the Neural IR and by taking these gaps as guidelines for the study, they may lead to significant performance optimization.

Since the use of deep learning in IR is still recent, most of the reviewed papers were collected between 2010 and 2018 and in order to include all the proposed models, the papers were chosen from the following domains: Neural information retrieval, deep learning for semantic matching, deep learning for NLP and information retrieval. Neural IR is a newborn domain which has been specifically established to connect and support all new researches using deep learning models to solve information retrieval problems. However, a good portion of important works is being published under other domains. Thus, the review included some important works form other related domains like Semantic Matching, NLP and IR.

In the beginning, more than one hundred papers were collected by searching in ISI journals and in the most known IR conferences like the Special Interest Group on

Information Retrieval (SIGIR). Afterward, the papers were filtered based on two criteria. While the first criterion excludes all papers that do not focus on long text matching which is the typical case for ad-hoc retrieval, the second one excludes all papers that do not use a neural model.

After applying the filtering criteria, only 25 papers remain and most of these papers where a conference paper published in the last two years which reflects the fact that deep learning in general and neural IR are still new domains.

The review is organized as follow: Section 1 provides a brief definition for some important IR terms that appear frequently in the study. Afterward, the most known traditional information retrieval approaches are briefly described in Section 2. The basic deep learning techniques that have been used in IR tasks are reviewed in Section 3. Section 4 review the main deep learning approaches for ad-hoc retrieval. Section 5 reviews the loss functions that have been used to train neural IR models. IR factors that have been considered in the different neural IR models are explored in Section 6. The review discussions and research gaps are provided in Section 7. Finally, Section 8 concludes the review.

2.1 IR Concepts

This section provides a brief definition of the basic IR terms and concepts that are used in the study.

a. Ad-hoc Retrieval

Ad-hoc retrieval is the task of retrieving the relevant documents for a previously unseen query from a static collection of documents. The returned document list is ranked and decreasingly ordered where the probability of relevance of a document in the list is considered independent from other documents that come before. The retrieval in this task does not depend on any query logs or user preferences and it

does not include any further interaction with the user (Collins-Thompson, Macdonald, Bennett, Diaz, & Voorhees, 2014).

For instance, given a collection of documents $C = \{d_1, d_2, \ldots, d_n\}$ from a certain domain, like political news, and a previously unseen query like q_1= *"US election meddling"*. In the ad-hoc retrieval task, our goal is to sort all the documents in C according to their relevance to q_1.

Typically, ad-hoc retrieval is performed on relatively long textual data which make it different from other IR tasks like question answering or from other retrieval tasks that depend on short texts such as titles and anchors.

b. Term Frequency

Term frequency represents the number of times a certain query term appeared in the given document (Manning, Raghavan, Schütze, & others, 2008, p. 96).

For example, given the document d_1 = *"The American president claimed his Russian counterpart was "extremely strong and powerful in his denial" of any election meddling*" and the query q_1= *"US election meddling"*, term frequency of word *"election"* and "*meddling*" equals 1.

Regardless of its simplicity, term frequency is one of the widely used IR features that represents the cornerstone of a wide range of IR models.

c. Term Importance

In the term frequency, as described above, all terms are considered equally important. This equality consideration is a critical problem for any IR model that depends only on term frequency. For example, in a query like "second world war" the word "second" does not hold the same discrimination power as the word "war". In fact, it turns out that terms which occur too often in a corpus are not as informative as rare terms. In this connection, term importance refers to any measure that is used by IR models to express query terms discriminative power

(Manning et al., 2008). For instance, the inverse of the number of documents in a collection that contain a certain term referred to as inverse document frequency IDF was used by multiple models as term importance measure.

d. Exact match

Exact match refers to the lexical match between a query term and a document word characters and it is used for counting term frequency. Typically, the query and the document texts are fed into a stemmer before searching for exact matches. The stemmer in this context is used to remove any suffix or prefix that may affect the lexical match (Manning et al., 2008, p. 29). For instance, in a document d_1 = "*Trump said that he actually accepts the intelligence findings that Russia meddled in the U.S. election*" we can observe and exact match of word stem "*elect*" and word stem "*meddl*".

e. Semantic match

Unlike the exact match, the semantic match intends to search for query meaning inside the document. In other words, semantic match measures to which extent the document is about the query topic. For example, even though there is no exact match, a document about "Kuala Lumpur" should be relevant to the query "Malaysia" (Mitra & Craswell, 2017b). Further, a phrase like *"political race"* gives the same meaning as *"election"*, although different words were used.

f. Query Coverage

Although the name is new, query coverage as a notion, to some extent, is reflected in most of the traditional IR models such as BM25 and LM. Query coverage means how many query terms have been covered in the document. For a multiple terms query, the intuition behind query coverage is to favor documents with high, or even low, terms frequencies and high query coverage over documents with high terms frequencies and low query coverage because in the latter case the document

is biased toward a subset of query's terms. For instance, a document with a high frequency of terms "Information" alone or term "retrieval" alone should not be more relevant to the query "information retrieval" than a document that has both terms (Hui, Yates, Berberich, & de Melo, 2017a).

g. Term Proximity

Term proximity reflects to which extent query terms co-occur close to each other in the document. For instance, the co-occurrence of query terms in the boundary of one sentence gives a higher indication of relevance than the co-occurrence in scattered places. Moreover, large portion of everyday queries are about certain concept (e.g. "Information Retrieval"), place name (e.g. "Time Square"), proper names or titles (e.g. "Taylor Swift" or "Game of Thrones") where the underlying meaning will change dramatically if distance increases between query terms (Rasolofo & Savoy, 2003).

2.2 Traditional IR Models

Traditional models in this context refer to any IR model that does not use a neural network. Even though this definition includes a large number of diverged models, all of these non-neural models can be classified into three main categories: Term Frequency, Latent or semantic Space and Language Models.

Since the review focuses on neural models, a brief description is provided for the most known version of each category.

2.2.1 Term Frequency (TF)

Term frequency refers to an old family of IR models which use a handcrafted equation to compute the relevance score between a query term and a document depending on the count of term occurrence in the document (known as term frequency or TF) and on how many documents in the collection contain that term (known as documents frequency or

DF). The following section reviews BM25 as one of the most effective term frequency-based IR models.

2.2.1.1 Okapi BM25

BM25 is one of the most effective and well-known traditional IR models which have been the state-of-the-art performance and the baseline for other IR models for a long time. It defines for each query term two components: the first one measure the local relevance and the second one measures the global term weight. Local relevance express document eliteness where elite documents are those that are about the concept represented by the term. Document eliteness is computed as a 2-Poisson distribution of the term frequency where it increases as the term frequency increase in the document. In order to consider document length, the local component is normalized using document length and the average document length in the corpus (Robertson & Zaragoza, 2010). In the other side, the global weight, referred to as term importance, indicates how much the occurrence of a certain term adds to the query relevance. This term importance can be measured using different equations depending on the available evidence (Robertson, 2004). Though effective, BM25 is considered as an exact match model. That is, it is only effective when the exact query words appear in the document. This limitation makes BM25 inadequate to meet the semantic match requirement and thus it is usually combined with other IR tasks like query expansion in order to alleviate this limitation. Moreover, the BM25 formula does not consider other important IR features like term proximity. As a result, documents with few terms that co-occur in a close distance may be outranked by documents with scattered terms but with higher frequency.

2.2.2 Latent Semantic

Latent semantic indexing refers to a family of indexing methods that tries to find a latent space of compact dimensions and project documents and queries as vectors or points in

that space. Consequently, the relevance between a document and a query is measured by the distance between their projections. The following section looks into LSA as one of the most known latent semantic methods.

2.2.2.1 Latent Semantic Analysis (LSA)

In 1990, Deerwster proposed the LSA model as a solution for the semantic match limitation in the term frequency approach. The main idea of this approach is that the relevance between a document and a query should not be based on the occurrence of query's terms because the same meaning can be expressed by different terms and one term may have different meanings in different contexts. Instead, the relevance relationship between documents and queries' terms is transferred into an implicit higher-order space known as the semantic latent space. The model takes as input the bag of words representation of all documents and queries' terms and returns the representation of each document and each term in the latent space. In order to find the best approximation for the latent space, Deerwster used the Single Value Decomposition (SVD) to divide the bag of words matrix into three new matrices: terms matrix, singular values matrix, and the documents matrix. Afterward, the singular values matrix is reduced into the top K values and therefore the other matrices are reduced as well. After construction, the latent space representations, new queries or new documents can be represented by computing the centroid of their constituent terms (Deerwester et al., 1990). Even though LSA is a powerful model which is able to preserve both exact match and semantic match signals, it suffers from several drawbacks and limitation as well. Beside its computation complexity, it is not clear when LSA needs to reconstruct its latent space representations after updating it with new terms and documents (Manning et al., 2008). More important, like BM25, LSA discards positional information which means that it is not able to detect other important features like query coverage and proximity. Furthermore, LSA latent space can be regarded as a local space which over fit the selected corpus at the

construction time, thereby it is obvious that new documents that may belong to external domains will not be represented accurately.

2.2.3 Language Models (LM)

Language models are a family of information retrieval models that represent each document as a language model and a query as a random sample that could be generated from the document model. From the probabilistic view, a language model can be considered a function that assigns a probability for each string that can be selected form some vocabulary. The simplest language model used for IR is the unigram model where each term (string) is estimated independently. To rank a document, the model calculates the probability that the given query is generated from the document which is done by multiplying the probability of each query term. Term probability is computed as a percentage of term frequency to the document length. The classical problem of such ranking model is that if only one query term does not appear in the document the rank will be zero. In order to alleviate that problem, the model adds another term weighting factor known as model smoothing. That is, instead of depending only on term frequency over the document to estimate term probability, the model considers also term frequency over the whole documents collection which is very similar to the inverse term frequency in the term frequency models (Manning et al., 2008). Even though in some experiments LM models were able to outperform other traditional models like term frequency models (Ponte & Croft, 1998), it is still considered as a variation of the bag of words models and thus it fails to capture a number of important IR features like semantic matching and proximity.

2.3 Deep Learning Concepts

Conventional machine learning approaches struggled for decades with the problem of processing raw data where they needed a very careful preprocessing stage in that

engineers and domain experts transform the data and extract the required features for the learning procedures. In the other hand, representation learning emerged as a new class of ML methods that can take raw data as input and learn to extract the required features for different ML tasks. Deep leaning can be considered as a representation method which consists of multiple representation level or layers. The first layer extracts some basic features from raw data and passes it to the next layers. Each layer is a basic neural networks layer that typically consists of a linear transformation function and a nonlinear activation function. Consequently, each layer can be regarded as a nonlinear module which learns a more abstract level of representation (LeCun, Bengio, & Hinton, 2015). The popularity of deep learning nowadays can be returned to three factors: the increase in the GPU units processing capabilities, the low cost of computing hardware and the dramatic advancement in the learning algorithms and techniques which allows for reasonably efficient training of neural networks with many hidden layers (Deng, 2014).

2.3.1 Basic Neural Network

A simple neural network consists of multiple processors (classifiers) called neurons. Each neuron can have multiple inputs and one output. Typically, a shallow neural network consists of three layers. The first one is called the input layer where neurons receive their

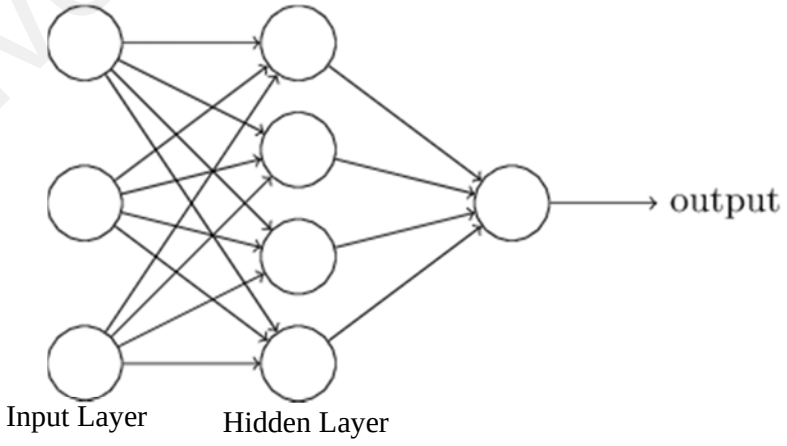

Figure 2.1: Basic Neural Network

inputs from the external environment as raw data. The second layer is called the hidden layer and neurons in this layer receive their inputs through weighted connections from the previous layer's neurons (Schmidhuber, 2014). The final layer is called the output layer and it plays as a global classifier which chooses the most important features from the previous layer to give the final output Figure 2.1.

Traditionally, training a neural network consists of two stages: the first one is called the forward pass and the second one is called the back-propagation pass. In the forward pass, the input is fed into the first hidden layer and as such the output of that layer will be the input for the next layer and so on. The output of each neuron could be calculated from the input array *x* as follows:

$$h = \sigma(w.x + b)$$

Where *w* is the weights of the connections for that neuron.

$w.x$ is the dot product of the input array and the weights array.

σ is the non-linearity activation function and *b* is the bias value.

After computing the final output, the result is compared with the intended output and the difference is considered as the network error. Then the back-propagation pass starts by updating each weight according to the partial derivative of the error with respect to that weight.

In the domain of information retrieval, this basic neural networks models (also known as fully connected layer) were used in multiple models as a ranking layer where the input is the features which have been extracted using different techniques and the output is one scalar representing document rank (Mitra, Diaz, & Craswell, 2017; Pang, Lan, Guo, Xu,

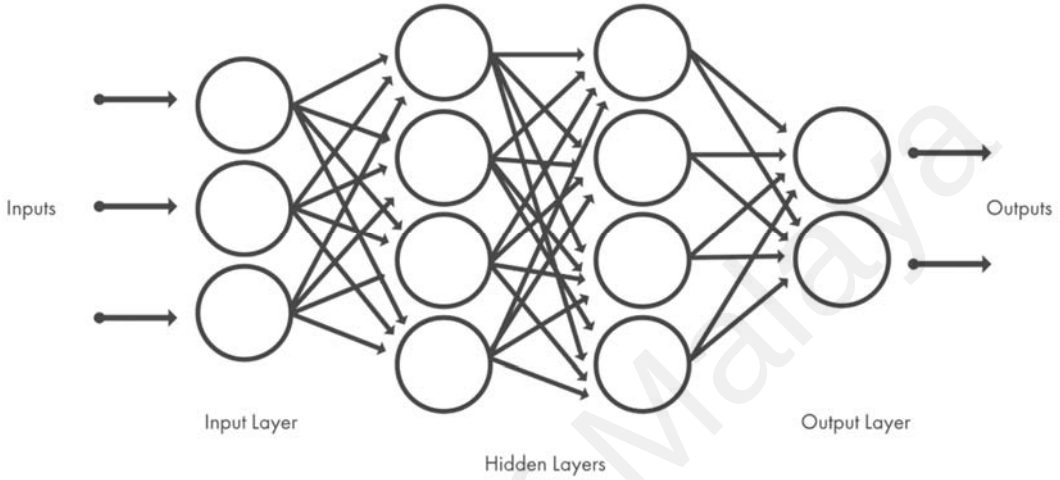

Figure 2.2: Deep Neural Network of two hidden layers

& Cheng, 2016; J. Wang et al., 2017). By stacking more hidden layer the network becomes deeper and then the model is called Deep Neural Network (DNN) Figure 2.2[1]. Like the basic network, DNN has been used by different neural IR models for different purposes. For instance, a deep network of two hidden layers was used by (Guo et al., 2016) as a matching network whereas (Huang et al., 2013a) used a deep network of three hidden layers for transforming high dimension text features into low dimension semantic space.

2.3.2 Word Representation

The first problem that encounters all deep learning applications for NLP, in general, is that neural network input should be numerical, not textual. In response, several models have been suggested such as the very basic one-hot vector, the trigram word hashing

[1] Image taken from www.mathworks.com/discovery/deep-learning.html

(Huang et al., 2013a) and the word embedding vector model (Tomas Mikolov∗ , Wen-tau Yih, 2013).

The one-hot vector represents each word as a binary vector where all values are zeros accept the index of the corresponding word is one. The problem in this representation is the high dimensionality where the number of dimensions is the number of target language words.

In 2013, Huang et al. proposed word hashing as a solution to the high dimensionality problem where words are broken down into letters n-grams. Then each word is represented as a vector of letters n-grams. As such, the queries or documents words can be represented with lower dimensionality compared to the one-hot vector because unlike the number of the words in a language, the number of the possible n-grams is limited and much smaller.

Even though one-hot vector and word hashing transform a word from the literal space to numerical space, they fail to preserve and express most of the desired features like syntactic and semantic relations which left the door open for more advanced representation approach.

It is widely agreed that the invention of word embedding played an important role in the success of deep learning applications in NLP for two reasons: first its ability to preserve and reflect words semantic and syntactic relations. Second is the ability to build and update it in an unsupervised way from any corpus.

Simply put, word embedding is a method that transforms words from literal space to the latent (semantic) space where words with similar features can have close representations. In order to learn such words representations, there are two popular approaches: the bag of word matrix decomposition (Deerwester et al., 1990) and the neural approach (Goldberg & Levy, 2014). However, the neural approach which is built using general

corpus like Wikipedia performs better than the matrix decomposition approach on different NLP tasks (Mitra & Craswell, 2017a).

In order to learn word embedding based on word's syntactic context, (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) introduces two similar unsupervised neural architects of one hidden layer. The first one is called the skip gram model and its objective is to find the representations of the surrounding words given the central word of the input sentence; whereas, the second one is called the continuous bag of words model (CBOW) and its objective is to predict the central word from its context.

Interestingly, the learned word vectors are able to preserve very useful features like semantic and syntactic relations. For example, the distance between a word vector (e.g. "shirt") and its hypernym (e.g. "clothing") will be always close to a fixed constant. Similarly, the distance between a word vector (e.g. "Apple") and it's plural form (e.g. "Apples") will be rather fixed.

In the domain of neural information retrieval, a wide range of words representation techniques were exploited. Works like (Huang et al., 2013b; Liu et al., 2015; Nalisnick, Mitra, Craswell, & Caruana, 2016; Palangi et al., 2016; Shen, He, Gao, Deng, & Mesnil, 2014) used different forms of word hashing for word representation. However, since word hashing is not able to express words syntactic or semantic relations those models rather failed to solve the semantic matching problems.

Similarly, different forms of word embedding were used in more recent works such as (Ai, Yang, Guo, & Croft, 2016; Guo et al., 2016; Hui, Yates, Berberich, & de Melo, 2017b; Jaech, Kamisetty, Ringger, & Clarke, 2017; Pang et al., 2017a; Pang, Lan, Guo, Xu, Wan, et al., 2016). In 2016 Mitra et al. argued that the use of the current form of word embedding is not suitable for the ad-hoc retrieval because the current neural architect for learning word embedding tends to give a closer representation for words that have the same type or function than words that are about the same topic or domain. In order to

overcome this drawback, Mitra et. al. suggested to use two embedding one for the query and one for the documents where query's terms and document's words that are from the same topic will have a close representation (Mitra, Nalisnick, Craswell, & Caruana, 2016).

Furthermore, (Xiong, Dai, Callan, Liu, & Power, 2017) trained the whole model end-to-end and hence the word embedding was tuned gradually to become more task-specific representations.

2.3.3 Convolution Neural Network (CNN)

Convolution neural network is a widely used model in computer vision. It originates from neuroscience where Hubel and Wiesel in 1962 showed that some neurons in visual cortex activated only for edges or some orientations (LeCun, Kavukcuoglu, & Farabet, 2010). The idea is to have specific components which learn to detect a very basic feature in the input image. These detected features are then fed for other layers which can detect more advanced features by summing up the former basic features.

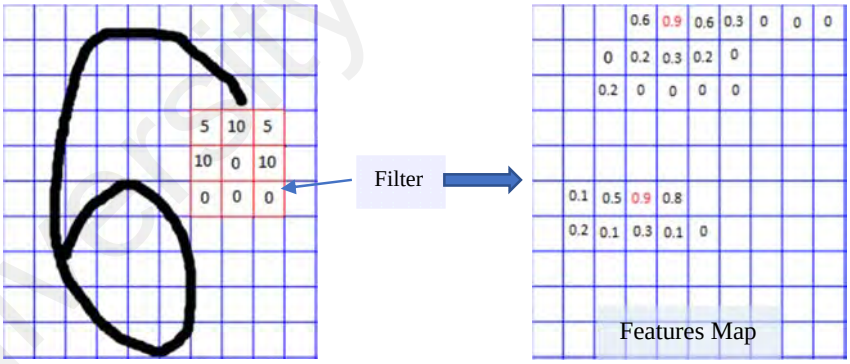

Figure 2.3 Convolution Filter

The basic unit in CNN is called filter or feature detector. This filter is a small block which scans the whole image and results in a map. The map represents that feature distribution in the input image. Figure 2.3 shows that after applying the filter for the image, it detects a horizontal curve in two positions.

Convolution means that the filter convolves around the entire images searching for its feature. More generally, it implies that each feature in the top-level layer was extracted by scanning the whole input.

The main difference between CNN and other classical computer vision approaches is that in CNN the filters weights are learned gradually during the training phase. Typically, multiple filters are used for detecting different features, and thus, the CNN output a different features map for each filter. These maps are then reduced using max function and this reduction is called max pooling or pooling layer. Consequently, the features vector preserves and represents the positional distribution of the detected features over the whole input matrix. CNN for detecting complex objects like digits or faces may contain several convolution layers. Although the final layer depends on the application, in general, it is a classification network which consists of a stack of fully connected layers. Typically, the output of the final layer is an N-dimensions output vector where N is the number of required classes. In this way, the classification layer is called positional classification network because the final features vector represents the positional distribution of all features over the whole input matrix.

Even though CNN is widely used in the field of computer vision, recently there is an increasing number of research papers which incorporate CNN in NLP applications. Nevertheless, one of the most challenging problem in using CNN for NLP tasks is the input size. That is, CNN originally designed to take a matrix of pixels as input whereas the input for NLP tasks is a sequence of words. In this concern, multiple approaches were suggested for constructing an equivalent representation matrix for the input sentence or

document using some numerical word representation like word hashing or word embedding (Kim, 2014) Figure 2.4.

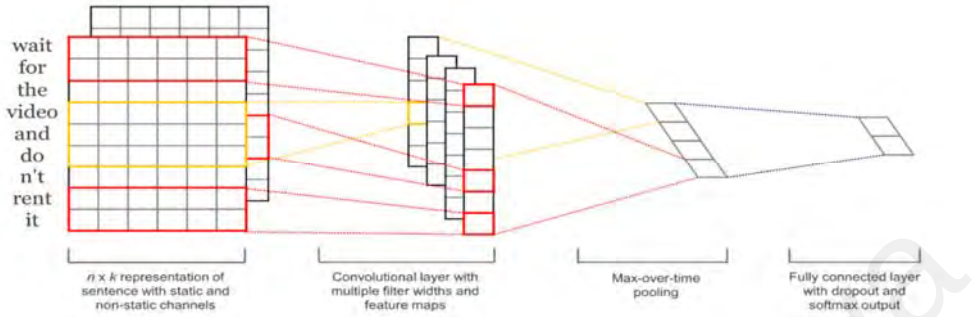

Figure 2.4 CNN for sentence classification (Kim, 2014)

With the growth of the neural IR domain, the use of CNN increased gradually, and it is obvious that in the last couple of years CNN dominates other deep learning models in neural IR. In 2014 Shen et. al. used a one-dimension CNN to learn query and document representation. The model was able to outperform most of the traditional models in the short text retrieval task (Shen et al., 2014). Afterward, and starting from 2016, when Pang et al. introduced the first interaction model, CNN has been used in several models like (Hui, Yates, Berberich, & de Melo, 2017c; Jaech et al., 2017; Mitra et al., 2017; Pang et al., 2017a). Finally, driven by the expectation that IR features follow rather simple patterns, a simple CNN network of one layer was used by almost all works.

2.3.4 Long Short-Term Memory Neural Networks (RNN, LSTM)

LSTM is a special variation of a more general model of neural networks known as recurrent neural networks RNN. Unlike other types of NN, recurrent networks predict the next input term in the input sequence. This type of networks is more suitable for temporal processing and learning sequences.

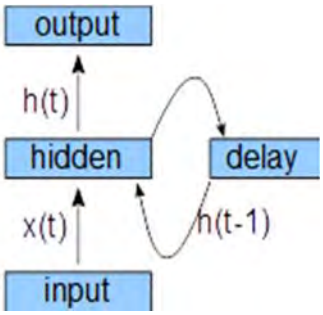

Figure 2.5 : Recurrent Neural Network

As illustrated in Figure 2.5, in time t, the input for the hidden units is the input value $x(t)$ in addition to the activation value of the hidden unit in the previous step $h(t-1)$. To memorize activation values for a long time, LSTM extends RNN by an analog memory circuit which has three types of gates: write for storing information, read for reading its value and forget for resetting.

Although RNN gives a great performance in NLP applications in general, it has not shown the same performance in ad-hoc retrieval tasks. Specifically, RNN and LSTM have been used by (Palangi et al., 2016) to build document representation based on what they called the sentence embedding. Sentence embedding is constructed by feeding each word representation to RNN or LSTM network. The output of the network at the last word is considered as the semantic representation of the input sentence.

In more recent work, (Pang et al., 2017a) used an RNN for aggregating each query term representation from different contextual positions.

Lastly, (Hui et al., 2017a) used LSTM as a final layer which takes as input a sequence of features vectors where each vector represents the output of the CNN layer for one query term combined with the IDF of the corresponding term.

2.4 Neural Information Retrieval

Neural IR is a new domain that has emerged in the last couple of years to contain and organize all the researches and activities which employ deep learning techniques for

solving IR problems. Even though the domain is concerned with a wide range of known IR tasks such as question answering and recommendation systems, the ad-hoc retrieval task is still the main issue of this domain and most of the published works are concerned with it. The neural IR works for ad-hoc retrieval can be classified into three main approaches; namely, learn to rank approach, representation approach, and the interaction approach. The review focused more on the interaction approach for it is the most recent approach and, as will be shown, it was able to outperform other neural IR approaches and gain state-of-the-art performance.

2.4.1 Learn to Rank (LTR)

Learn to rank is rather an old approach which, besides the neural networks, have employed many other machine learning models, such as support vector machine and decision tree for the ranking task. However, the rise of deep learning opens the door for a new level of machine learning models. In response, a number of new works have been done to explore these new capabilities for learn to rank task.

The main idea in the LTR approach is to represent a query/document pair as a vector of hand-crafted features. The features can be divided into query features like query length, document features like document popularity and dynamic features which depends on both query and document like term frequency (Mitra & Craswell, 2017b). After constructing the features vector, a neural model is trained to rank the corresponding query/document using the features vector. As such, most of the available neural IR models can be considered as LTR models which differ from each other in the feature extraction approach. Some of these models are using supervised approach for feature extraction like (J. Wang et al., 2017; Xiong et al., 2017) while others are using an unsupervised approach like (Mitra et al., 2017; Pang, Lan, Guo, Xu, & Cheng, 2016).

Interesting work has been done by (J. Wang et al., 2017) in which they use the generative adversarial network (Goodfellow et al., 2014) to learn to rank query/documents pairs after representing each query/document as a list of handcrafted features. Two neural models were used: the first one is called the discriminative model and it is responsible for ranking query/document pairs. The second one is called the generative and it is responsible for generating (finding previously unseen) documents that can fool the discriminative model and get high rank for the input query. Their experiment showed that after several training iterations, the generative model learns to find new relevant documents that have not seen before. Even though the model was only compared with other LTR models, it is not expected to be able to outperform the state-of-the-art performance. Nonetheless, the models could be very promising in training other models from semi-supervised or unlabeled data.

2.4.2 Representation Based

In this approach, the model is trained to transform the document and the query from the literal space to latent space separately. Then, a similarity function (e.g. Cosine) is used to measure the relevance between the document and the query representations Figure 2.6. The main problem of the representation based is that they postpone the interaction between the document and the query until the last step which, in most cases, makes the model discard relevant information.

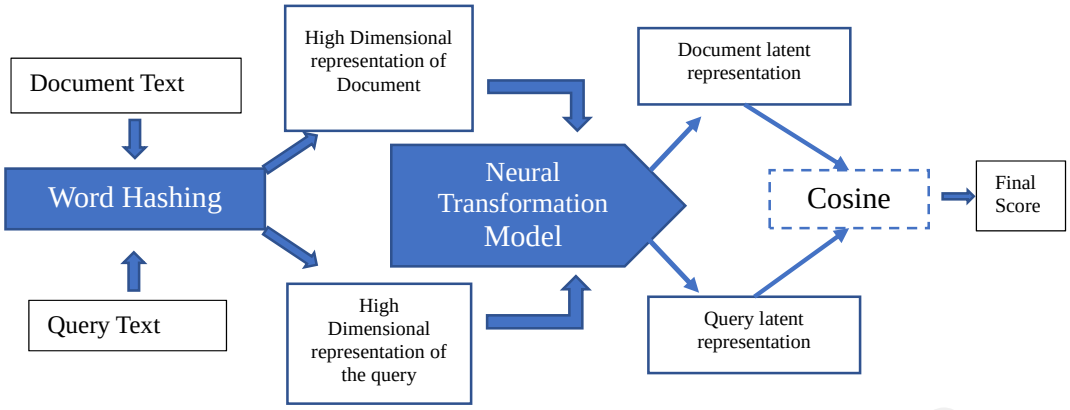

Figure 2.6: Representation based paradigm

In 2013 Huang (Huang et al., 2013b) proposed the first representation based model for ad-hoc retrieval task (coined as DSSM). In their model, they used word hashing to represent each word as a low dimension vector. Using words representation, a DNN of several hidden layers was trained to project a document into a low-dimension dens features vector in latent space. The experiment proved that DSSM was able to outperform most of the traditional IR models (e.g. TF-IDF, BM25, and LSA). Later on (Shen et al., 2014) suggested a more advanced representation architect (known as CLSM or CDSSM) which uses CNN as a transformation layer instead of DNN. Specifically, the document or the query is divided into sentences. A sliding window of length *n* is used to scan each sentence and for each window, a matrix is constructed by concatenating all included words hashing vectors. Next, the matrix is fed into a convolution layer to project the corresponding matrix into a low-dimension features vector. Max pooling over all windows is used to select the best representation of the input sentence, and finally, a fully connected layer is used to extract the global representation of the whole document or query. CDSSM outperformed DSSM in addition to the most traditional IR models.

Another variation of DSSM proposed by (Liu et al., 2015) (referred to as MT-DNN) where the DNN layer shared between different tasks (i.e. ad-hoc retrieval and query

classification) in order to get more general representation. The model was able to outperform both DSSM and CDSSM.

Further, LSTM and word hashing were used by (Palangi et al., 2016) to extract what they call sentence embedding. Simply, the LSTM takes as input a sequence of words hashing vectors and return a vector of the same dimension as sentence embedding. Afterward, the sentence embedding was used to construct query and document representation. The model (referred to as LSTM-RNN) was able to outperform DSSM, CLSM, and traditional models. In the same time, other embedding architects were utilized to build the query and the document representations like paragraph embedding or PV-DBOW (Ai et al., 2016) and IN/OUT embedding (DESM) (Mitra et al., 2016).

(Mitra et al., 2017) proposed a hybrid model (coined as DUET) which incorporate both the representation based and the interaction-based approaches. While the interaction-based model was used for the exact matching, the representation based was used for detecting the inexact (i.e. semantic) matching. Similar to CDSSM, DUET uses word hashing, CNN and max-pooling for projecting query and document into latent space. However, they use two different models with different configurations for the query and for the document. Furthermore, the representation part of DUET used a different approach for matching query and representations. First, they get the elementwise product of the two representation and then they use a fully connected layer to get the final score. Even though DUET was able to outperform most of the proposed neural models (e.g. DSSM, CDSSM, DRMM) and most of the traditional models, it was outperformed later by other interaction-based models (Hui et al., 2017a, 2017c).

2.4.3 Interaction Based

Originally, the interaction approach was proposed in the NLP domain as a method for measuring semantic matching between two sentences. First, an interaction structure of the two input sentences is constructed using some word representation like word embedding. Next, a set of features that reflect the relevance matches are extracted from the interaction structure using a matching model. Finally, the extracted features are passed into a scoring model in order to get the semantic similarity score (Hu, Lu, Li, & Chen, 2014).

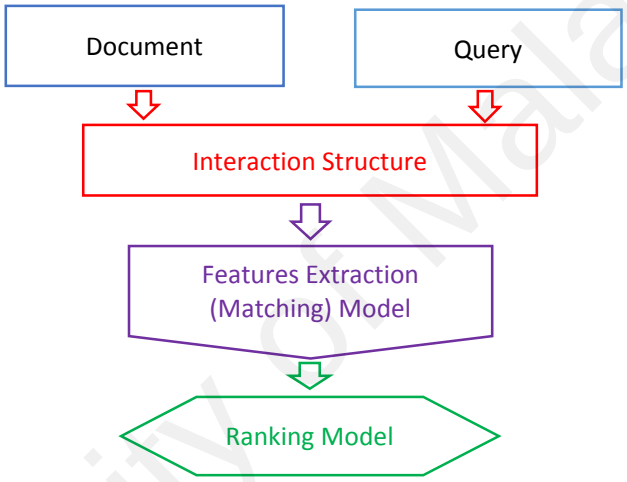

Figure 2.7 Interaction Approach Basic Components

Using a query and a document instead of two sentences, the interaction approach for ad-hoc retrieval utilizes the same paradigm which consists of three main components Figure 2.7:

- The Interaction Structure
- Feature Extraction Model
- Ranking (Scoring) Model

The following sections describe and review each of the interaction components.

2.4.3.1 Interaction Structure

The purpose of the interaction structure is to reflect the matches between the input query and the document; therefore, the richness of the matching features that the proposed interaction structure is able to reflect significantly affects the whole model performance.

In 2016, Pang et al. proposed the first interaction based model for information retrieval in which they utilized the same interaction structure that was originally proposed by (Hu et al., 2014) for the semantic matching task. The interaction structure is called the *Interaction Matrix* and it is still considered as the most plausible structure in the interaction-based retrieval domain.

In the interaction matrix, each cell represents the semantic similarity between a document word and a query term Figure 2.8. In order to compute the semantic similarity, they used the word embedding as word representation and some similarity function like dot product of the two embedding vectors or the cosine function of the angle between them. As a result, we get a matrix of length equals to the document size and height equals to the query size in which two types of cells can be distinguished: the exact match and the background cells. The exact match cell is any cell of the maximum similarity value while the background cells represent the rest of the cells in the matrix (Pang, Lan, Guo, Xu, & Cheng, 2016).

What makes the interaction matrix a very promising interaction structure for IR applications is that in addition to the semantic similarity between the query and the document, it completely represents the exact match signal. Moreover, the interaction

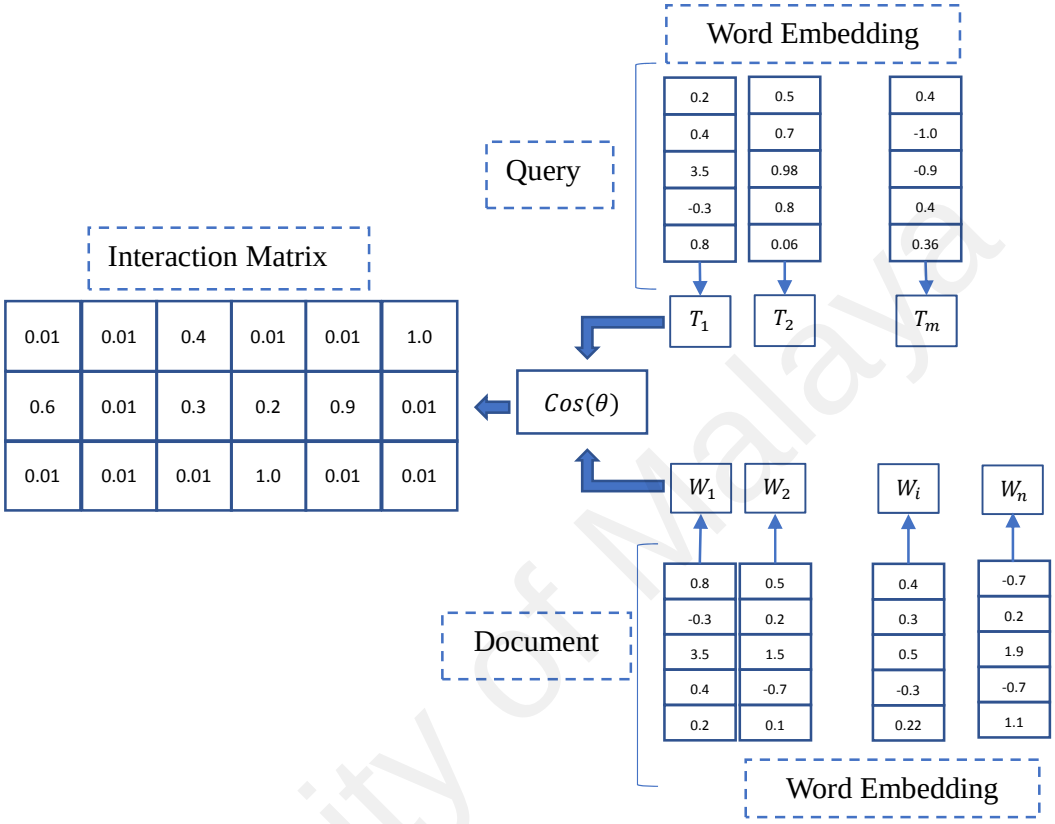

Figure 2.8: Interaction Matrix

matrix preserves all positional matching information which makes it suitable for detecting more advanced IR factor like term proximity.

In the same year, (Guo et al., 2016) argued that the positional information in the interaction matrix is noise in the context of the information retrieval, and therefore, they proposed a more simple structure called match histogram. Match histogram represents the distribution of the similarity values in each row of the previous interaction matrix structure. As a result, instead of the interaction matrix, there will be a matching histogram for each query term representing the distribution of similarities values between that term and all document words.

Later on, in 2017 Mitra et al. used the binary form of the interaction matrix in order to represent the exact match signal in a hybrid model which combine the representation-based and the interaction-based approaches (Mitra et al., 2017).

Conversely, in 2017 two works suggested that the interaction matrix is not enough and they proposed to extend that structure. (Jaech et al., 2017) hypothesized that interaction matrix is not able to capture both local and global relevance, hence; they suggested to extends the interaction matrix by other similarity channels. Using representation neural model, the word embeddings of both query and document are reduced into low dimensional representation and the element-wise products of each query term and document word new representations vectors were embedded into the interaction matrix. Similarly, (Pang et al., 2017b) suggested embedding the corresponding word embedding vectors of both the query term and the document word into each interaction matrix cell.

2.4.3.2 Feature Extraction Model

After constructing the interaction structure, it is passed into the feature extraction (or the matching) model in order to capture any evidence of relevance matches between the corresponding query and document. If the features extraction fails to capture all the required features in the interaction structure, the model performance will be poor no matter how expressive the interaction structure is. Among the reviewed works two approaches for feature extraction can be distinguished: unsupervised and supervised.

1. Unsupervised Feature Extraction:

   In the unsupervised approach, the features are extracted implicitly without any human intervention using deep learning techniques. That is, the feature extraction is a neural network which scans the interaction structure and learns to extract some hidden features in order to increase the overall accuracy of the retrieval model.

   The first deep learning technique for features extraction proposed by Pang et al. in 2016. In their model (called MatchPyramid) they looked at the interaction

matrix as a 2D image and hypothesized that using some deep learning techniques for visual pattern recognition, most of the important features will be extracted. Their model consists of one CNN layer followed by max pooling layer for feature reduction. However, the model was not able to outperform some traditional IR model like BM25 and the model was not able to capture some important IR factors like term proximity (Pang, Lan, Guo, Xu, & Cheng, 2016).

Since then, deep learning techniques for visual pattern recognition have been the dominant features extraction approach for interaction-based IR. Some works proposed to stack more CNN layers in order to detect more advanced features (Jaech et al., 2017). While others argued that it is more important to used multiple CNN networks with different filter sizes. For example, a filter of size 2x2 should capture the match of two successive query terms whereas 3x3 should capture three terms and so on (Hui et al., 2017a, 2017b; Pang et al., 2017a). Furthermore, Hui et al. in the latest version of their model (coined as PACRR) proposed to use an extra CNN layer with filter size equal to the query size in order to capture term proximity (Hui et al., 2017b).

Other deep learning techniques for visual pattern recognition were tested like spatial RNN which is an extended version of RNN designed to work over a 2D matrix. However, experiments show that CNN is more effective for interaction feature extraction (Pang et al., 2017a).

2. Supervised Feature Extraction

Even though unsupervised techniques like deep learning dominate the feature extraction task in the interaction-based IR, some studies cast some doubts that these techniques may not be the most effective way for features extraction. In particular, Guo et al. in 2016 argue that models (e.g. MatchPyramid) that use deep learning techniques for visual patterns are designed to only capture positional

regularity which is not the most important factor in the context of information retrieval. Instead, they proposed to extract the features by applying histogram function over each row in the interaction matrix. The purpose of the histogram function is to represent the strength of the matching signal (exact and semantic) of each query term regardless of its position (Guo et al., 2016).

In more recent work, Xiong et al. utilized handcrafted kernel functions to calculate the distribution of similarities values over each query term. Afterward, the similarities distributions are used as model features that are then passed to a ranking model (Xiong et al., 2017).

2.4.3.3 Ranking Model

After features extraction form the interaction structure, they are passed into a neural model to get the final score. Out of the reviewed works, many different deep learning architects were designed for the ranking task. However, these models can be classified into two categories: positional classification-based ranking, term-based ranking.

1. Positional classification Based

   Positional classification is a technique that has been used by most of the deep learning models for visual pattern recognition tasks. In this technique, the final features vector is reflecting the positional distribution of the features across the whole input matrix (image). Consequently, each neuron in the classification model eventually learns to look for some specific feature in specific positions.

   Even though IR ranking is not a positional classification problem, this technique was used by most of the proposed works and interestingly it was able to give effective performance in almost all experiments (Jaech et al., 2017; Mitra et al., 2017; Pang, Lan, Guo, Xu, & Cheng, 2016).

2. Term-Based

   This technique proposed by Guo et al. in 2016 where a features vector is separately extracted for each query term. The features vector of each term is then fed into a DNN of several layers to get a score for each term. Finally, a term gating network takes these scores and get the final score by incorporating terms importance weights.

   Likewise, (Xiong et al., 2017) extract a separated features vector for each term. However, instead of computing a separated score for each term, the features vectors are concatenated and fed into learn-to-rank similar architect in order to get one final score. Instead of concatenating features vectors and feeding them into a learn-to-rank network, (Hui et al., 2017a) proposed to sequentially feed them into an LSTM in order to get the final score. Nevertheless, in their updated versions (Hui et al., 2017b, 2018) found that both LSTM and learn to rank structure give a close performance; and therefore, they abandoned the LSTM because it is much more computationally expensive.

   A more advanced version of term based ranking was introduced by (Pang et al., 2017a). In this ranking model, the features are extracted for each query term from several contexts. Afterward, all feature vectors for each term are ordered according to their context position in the original document and fed into an LSTM to get a score for each term. At the end, similar to (Guo et al., 2016), a term gating network was used to aggregate the scores of all terms based on their importance weights.

Most of the interaction-based models were able to outperform both traditional IR models and other neural IR models like the representation-based models. More specifically, the first interaction based IR model proposed by (Pang, Lan, Guo, Xu, & Cheng, 2016) which called MatchPyramid was able to outperform representation-based models like DSSM

and CDSSM. However, the same study showed that MatchPyramid falls behind traditional IR models like BM25 and QL. In the same year (Guo et al., 2016) proposed their model which called DRMM which was able to outperform traditional IR models (i.e. BM25, QL) representation-based models (i.e. DSSM and CDSSM) and more importantly it was able to outperform MatchPyramid as well.

In 2017, several interaction-based models were proposed and most of them proved that the interaction-based models were gaining the state-of-the-art performance for ad-hoc retrieval. For example, DUET (Mitra et al., 2017), K-NRM (Xiong et al., 2017) and Deep Rank (Pang et al., 2017b) were able to outperform traditional IR, representation-based IR and other interaction-based models like MatchPyramid and DRMM.

Even though there is no study that compared all interaction based models, (Hui et al., 2017a) show that their model which called PACRR was able to outperform most of the other interaction-based models (e.g. MatchPyramid, DRMM, DUET, K-NRM). This makes PACRR and its updated versions (i.e. Co-PACRR (Hui et al., 2018) and PACRR-DRMM (McDonald et al., 2018)) the state of the art model for neural IR.

2.4.4 Interaction Models Simplicity & Efficiency

Model simplicity, in this context, reflects which degree a model is amenable for interpretation and analysis that are necessary for the future researcher to gain some insights into the relationship between IR and deep learning (Mitra & Craswell, 2017a, p. 91). In order to measure the simplicity of an interaction model we chose to consider the following four factors:

1. Interaction structure size: represents the size of the main input of the interaction model and it is expressed as a function of document and query sizes.
2. Networks Types: shows what types of neural networks used in the model.
3. Depth: represents the number of the overall layers in all networks.

4. Parameters size: the overall trainable parameters that should be optimized in the training process.

In the other side, even though efficiency evaluation requires the measurement of the training and execution times of each model on the same dataset, the above four factors are enough to give an estimation of models' efficiency since model training and execution time is proportional to the number of trainable parameters, networks types, and the model depth.

From Table 2.1, we can notice the increase in the models' complexity with time. This tendency toward increasing model complexity basically roots from the need for optimizing feature extraction process in order to optimize model performance. The majority of the recent interaction-based models responds to this need by one of the following two approaches: expanding the interaction structure or adopting more advanced deep learning models for feature extraction. However, both approaches led to more complex and less efficient models.

Specifically, (Pang et al., 2017b) in their model (DeepRank) proposed to embed query term and document word embedding vectors along with each similarity value in the interaction structure which resulted in an interaction structure of size $(n \times q_{size} \times c \times (1 + 2m))$ where c is the width of *n* short textual snippets from the document called contextual windows and *m* is the size of the embedding vector. Similarly, (Jaech et al., 2017) in Match-Tensor, suggested using two networks that consist of an LSTM and two DNN's to reduce both query and document words embedding into vectors of 50 dimensions. Afterward, the element-wise products of query term vectors and document words vectors are used to construct an interaction structure of size $(q_{size} \times d_{size} \times (k + 1))$ where k is the new embedding vectors size.

In both cases, the interaction structure is turned into a 3 dimensions tensor that needs for a kernel of the same rank to apply deep learning techniques like CNN for features extraction. For example, in the case of Match-Tensor, the size of a CNN filter of a kernel (3×3) will be $(3 \times 3 \times (k + 1))$. If we set k to 50 and used 18 of such filters (as proposed in their work) we need to train 8262 parameters for one layer in the feature extraction model.

As a result, in one hand we got heavy feature extraction techniques that need to train thousands of parameters and in the other hand, it is not clear what these features may reflect or represent which hinder our ability to analyze or interpret such models.

Differently, (Mitra et al., 2016) propose the DUET model that combine the representation and interaction approaches. In the interaction approach part, they used a CNN of kernel size (1×1000) to extract the set of features for each query term. By using 300 different filters followed by 3 DNN layers, they needed to train a model of 1.38×10^6 parameters for the interaction part alone. Later on, in 2018 (Hui et al., 2018) proposed to use *k* different CNN networks of different kernel sizes in their new model (CoPACRR) in order to extract richer features. Further, in order to extract the term proximity feature, they added another CNN network of kernel size (16×16) and 32 filters which alone requires the training of 8192 parameters.

Table 2.1: Interaction models simplicity comparison. TGN stands for term gating network. K CNN means k CNN networks with different kernel size. * indicates that the number of the parameters was provided in the study. ~ indicates that parameters number is an estimation.

	Year	Interaction Structure Size	Networks Types	Model Depth	Parameters Number
MatchPyramid	2016	$q_{size} \times d_{size}$	CNN, DNN	5	4980
DRMM	2016	$q_{size} \times d_{size}$	DNN, TGN	4	155
DUET	2016	$q_{size} \times d_{size}$	CNN, DNN	6	1.38×10^6
PACRR	2017	$q_{size} \times d_{size}$	K CNN, LSTM	6	1000
DeepRank	2017	$n \times q_{size} \times c \times (1 + 2m)$	CNN, LSTM, TGN	7	More than 2000~
K-NRM	2017	$q_{size} \times d_{size}$	Embedding Net, Kernel Pooling, LTR	4	$(49 \times 10^6)^*$
Match-Tensor	2017	$q_{size} \times d_{size} \times (k + 1)$	DNN, LSTM, 3 CNN	10	104000*
CoPACRR	2018	$d_{size}(q_{size} + 1)$	K CNN, DNN	6	18368
PACRR-DRMM	2018	$q_{size} \times d_{size}$	K CNN, DNN	6	943~

2.5 Loss Function

As mentioned in Section 2.3.1, in order to train any neural network, we should be able to measure the error (loss) in the network output. This error plays as a guide for the back propagation phase in which networks weights are updated to minimize that error. The main idea that has been used in nearly all works is as follow: depending on the available relevance judgments, for a query q, one positive (relevant) document d^+ and one or more negative (irrelevant) documents $D = \{d_1^-, d_2^-, \dots, d_j^-\}$ are used to create a training sample (q, d^+, D). Using the proposed neural IR model, the relevance scores $rel(q, d)$ of all documents in the training sample are computed given the input query q. Afterward, the relevance judgments and the computed scores are compared and using some predefined functions the model loss is computed for the corresponding training sample.

Specifically, two loss functions have been used in the reviewed works: the cross-entropy and the max margin.

2.5.1 Cross-Entropy Loss

The cross-entropy is the well-known loss function in the deep learning domain in general and it has been used for many different tasks. The first step in utilizing the cross-entropy for IR loss is to turn the training sample scores into a probability distribution using the softmax function Equation (1):

$$P(q, d^+, D) = \frac{e^{-rel(q,d^+)}}{\sum_{d \in \{d^+ \cup D\}} e^{-rel(q,d)}} \quad (1)$$

Then the loss value for the training sample (q, d^+, D) is computed using the equation

$$Loss(q, d^+, D) = -\log P(q, d^+, D) \quad (2)$$

When the proposed model is giving a higher score for the positive document than the negative documents, $rel(q, d^+)$ value will be higher than all $rel(q, d^-) : d^- \in D$. In this case, the probability value $P(q, d^+, D)$ will be close to one which makes the loss value close to zero.

On the other hand, when some or all the negative documents get higher scores than the positive document, the probability $P(q, d^+, D)$ value will be close to zero. In this case, the loss will be some large positive value.

Consequently, the objective of the neural IR model is to minimize the loss value given by Equation (2).

2.5.2 Max-Margin Loss

The max margin loss is designed for pairwise loss wherein the training sample there are only one positive document and one negative document (q, d^+, d^-). The objective of this

loss function is to make the difference between the two scores larger than some given threshold (typically 1).

$$Loss(q, d^+, d^-) = \max\left(0, 1 - \left(rel(q, d^+) - rel(q, d^-)\right)\right) \quad (3)$$

That is when the positive document score is higher than the negative document score by at least one the loss is zero, otherwise, the loss will be some positive value that is proportional to the difference of the two scores.

One thing that can be noted form the previous two loss functions is that both are only designed for binary relevance judgment. In other words, both functions divide the documents into two classes: relevant and irrelevant which makes them not suitable for rated relevance judgment where documents are classified into more than two categories.

2.6 IR Factors

IR factors refer to different aspects of relevance match that information retrieval models measure in order to estimate the relevance between a given query and a document. In fact, these factors are independent of the retrieval approach and, in the literature, there is no agreement on one unified set of factors.

Here, we only focus on the IR factors that have been considered by the available representation-based or interaction-based neural IR models.

Table 2.2: Common IR factors in the representation-based models

	DSSM	*CLSM*	*PV-DBOW*	*LSTM-RNN*	*MT-DNN*	*DESM*
Semantic Match	✓	✓	✓	✓	✓	✓
Query Coverage	✓	✓	✓	✓	✓	✓
Term Proximity	×	✓	×	×	×	×
Term Importance	×	×	✓	×	×	×

Table 2.2 shows that all representation models focus on the semantic match which corresponds to the fact that representation-based models are originally driven from the Latent Semantic paradigm. Query coverage is covered by all models since query representation always considers all query terms. Some models like CLSM paid special attention to terms co-occurrence by using terms n-grams embedding instead of the word embedding (Shen et al., 2014). This term co-occurrence awareness suggests that such a model could cover term proximity to some degree.

Term importance was considered by the PV-DBOW model where document frequency are included in the negative sampling strategy instead of the corpus frequency (Ai et al., 2016).

Table 2.3: Common IR factors in the Interaction-based models

	MatchPyramid	*DRMM*	*DUET*	*PACRR*	*CoPACRR*	*DeepRank*	*K-NRM*	*Match-Tensor*
Exact Match	✓	✓	✓	✓	✓	✓	✓	✓
Semantic Match	✓	✓	✓	✓	✓	✓	✓	✓
Query Coverage	×	✓	×	✓	✓	✓	×	×
Term Proximity	✓	×	✓	×	✓	✓	×	✓
Term Importance	×	✓	×	✓	✓	✓	×	✓
Term Disambiguation	×	×	×	×	✓	×	×	×
Cascade Reading/ Diversity	×	×	×	×	✓	✓	×	×

As Table 2.3 illustrates, both exact match and semantic matches were considered in all reviewed models while query coverage, term proximity, and term importance were considered in fewer models. Term disambiguation, which refers to the ability to exclude any other term meaning that does not fit to the current context, was suggested and covered in CoPACRR (Hui et al., 2018). Cascade reading or term diversity, on the other hand, refers to the importance of the positional information of local relevance matches

(Craswell, Zoeter, Taylor, & Ramsey, 2008). It was taken into account in two models CoPACRR and DeepRank (Hui et al., 2018; Pang et al., 2017b).

In this connection, it is important to note that not all of the above IR factors can be covered by the sole usage of the interaction matrix. For instance, term importance requires including other resources for estimating term importance. Besides, some of the above factors are only used by one or two works such as term disambiguation and cascade reading.

2.7 Discussion & Research Gaps

For several years, deep learning approaches struggled to outperform other information retrieval approaches. First, learn to rank approach proposed as a way to optimize other IR ranking models. Then, with the risen of deep learning techniques, the representation-based approach has emerged as the first deep learning approach for information retrieval. In this approach, a deep neural network is trained to project both query and document into latent space and the similarity of these projections is then used for ranking. From the reviewed works, it is clear that the representation-based approach capabilities sit somewhere in between of the latent semantic models (i.e. LSI) and the term frequency models (i.e. BM25). Considering its paradigm, the representation-based approach can be regarded as one of the latent semantic models. However, since most of the models are using the word hashing technique (which does not reflect or preserve words semantic relations), they moved more toward the lexical features of the text. The majority of the proposed models give close or better performance than most of the traditional models like BM25, LSA, and LM. Apparently, they failed to outperform the interaction-based approaches. This failure can be attributed to the fact that most representation-based models are not able to preserve or use positional information and therefore they are not able to meet important IR requirements such as term proximity. Indeed, representation-based models postpone the interaction between the query and the document until the final

stages which in most cases leads to discarding important information that may dramatically affect the final rank.

Later on, the interaction-based approach has been proposed as a solution for the delayed interaction problem in the representation-based approach. Inevitably, as shown in the review, interaction-based models were able to outperform both traditional IR and representation-based models and gain state-of-the-art performance. Still, in its turn, the interaction-based approach suffers from serious drawbacks that affect the possibility of incorporating it in real-world production applications.

First and most important, the interaction structure is a dynamic structure that must be constructed for each newly coming query with all documents. In order to build it, the whole document content is used each time. This limitation alone is enough to prevent building any interaction-based search engine. That is, it is vital for any search engine to be able to independently (of queries) reduce each document into a very simple structure at the offline phase and use that simple structure at the searching time. To alleviate this limitation, several works proposed to use interaction-based models in the telescoping mode in which a more efficient traditional IR model is used to get the first look at the whole documents. Then, the interaction model is used to re-rank the result of the traditional IR model (Hui et al., 2017a, 2018). However, this setting limits the interaction model performance by the used traditional model.

Not less important, as shown in Section 2.4.3, almost all proposed interaction-based models are based on the hypothesis that relevance matches are following some visual pattern in the interaction structure. Therefore, the majority of these models are using different deep learning techniques for visual pattern recognition. To our knowledge, no study has yet identified what features represent relevance matches and how they may be reflected in any of the proposed interaction structures. Guo et. al. cast some doubts that deep learning techniques are not the most effective way to extract the interaction feature.

Instead, they proposed to extract them using simple histogram function which considers positional information noise and discard them. Though simple, their model outperforms more complicated interaction based models and its performance is still close to the state of art (Guo et al., 2016). However, the fact that they are discarding positional information prevents the model from capturing some important IR factors like term proximity. The dynamic construction of the interaction structure added to the heavy and complicated deep learning features extraction models, made most of the interaction-based models: sophisticated models that are very hard to comprehend or analyze, and heavy models that need lots of computational resources for training and operating.

Additionally, unlike other neural IR approaches, the interaction-based approach has a problem dealing with varied query/document sizes. Specifically, the majority of the proposed features extraction and ranking neural models require a fixed input size. This restriction becomes a serious problem when the document or the query has varied sizes because, unlike other structures (e.g. images or sounds), text cannot be shortened or lengthened without affecting the meaning or losing some important information. To this end, most of the interaction-based models are either cutting the extra text when the query/document are longer than the intended size or padding some dummy characters when they are shorter.

Finally, nearly all the existing neural IR models are utilizing loss functions that are designed for binary relevance judgment. While this works well with the relevant/irrelevant case, it discards important information in the case of rated relevance judgment.

2.8 Conclusion

This chapter provided a broad overview of neural information retrieval domain by introducing concepts and techniques from information retrieval and neural networks that constitute the basic elements in this new domain. Afterward, the representation-based and interaction-based neural IR approaches have been reviewed in details and special attention was paid for the interaction-based models' simplicity and efficiency. Additionally, the review shed the lights on the used loss functions in both approaches and explored the set of IR factors that have been considered by different models. Finally, a discussion and possible research gaps where provided.

Chapter 3: Research Methodology

Recently, the interaction based deep learning approach has gained the state-of-the-art performance for the ad-hoc retrieval task. The approach can be divided into three main components: the interaction structure, the features extraction model and the ranking model. For feature extraction, the majority of the proposed interaction-based models are utilizing deep learning techniques that were originally designed for visual pattern recognition tasks. However, until now no study has provided proof that this extraction approach is able to extract all the required features. Besides, these extraction techniques have led in most cases into heavy and complicated models which makes them unsuitable neither for production applications nor for academic research.

The purpose of this works is to identify the features set that reflect the selected set of IR factors and provide a simpler and more effective approach for extracting those features. Based on the proposed features extraction approach, a simple neural IR ranking model will be developed and used in several experiments to evaluate the effectiveness and the efficiency of the proposed approach.

This chapter provides a detailed description of the research design, research methods, tools and data that have been used to achieve the research objectives.

3.1 Research Design

As mentioned in the introduction Chapter, the study concerns with the ad-hoc retrieval task which means that the main inputs for the proposed model are a previously unseen query and a collection of documents. In this respect, the main objective for our model is to rank all the documents in the collection according to the new query. To achieve that, the study is following the interaction deep learning approach which represents each query/document pair by an interaction matrix reflecting the relevance between the

corresponding query and document. In order to identify and measure the relevance signals in the interaction matrix, four IR factors were considered in the study:

1. Exact match
2. Semantic Match
3. Query Coverage
4. Term proximity

The above factors were selected based on the following two criteria: factors that are commonly used in the neural IR domain and factors that do not require any resources other than the interaction matrix (For more details on each factor see Section 2.1. For details on the relationship between neural IR models and the above factors see Section 2.6).

Given the interaction matrix as input and the above IR factors, the following research questions are answered in this work:

1. How the specified IR factors appear in the interaction matrix.
2. What features that best reflect the specified IR factors in the interaction matrix?
3. What is the most effective approach for extracting the identified features form the interaction matrix?
4. How to build an IR ranking model using the extracted features?
5. How to evaluate and prove the effectiveness and efficiency of the proposed model?

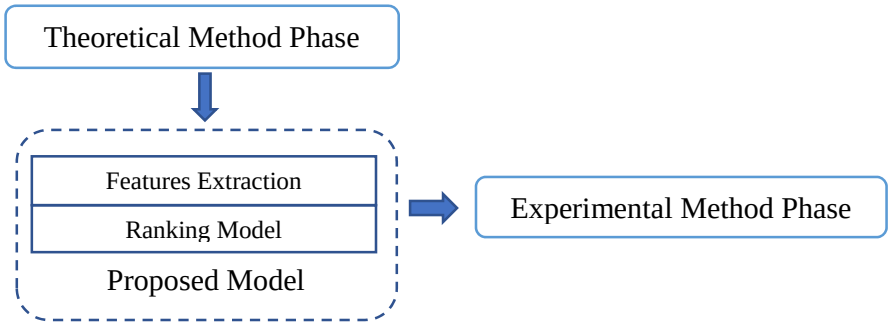

Figure 3.1 Research Design. The research starts with a theoretical study leading to the development of a new neural IR model. Afterward, the proposed model is evaluated in the experiment phase.

In order to answer the above research questions, research design of two components was followed. The first component is using a theoretical method to answer the first four questions while the second component is utilizing an experimental method to prove the effectiveness of the proposed mode Figure 3.1.

3.2 Theoretical Method

Following a theoretical method, this research component is responsible for analyzing the interaction matrix given the considered IR factors. The result of this analysis is the identification of a set of features called the interaction features (see Section 4.2.6) which reflect how the considered IR factors manifest in the interaction matrix. Afterward, a new features extraction technique is proposed to extract the interaction features form the interaction matrix. In the end, a new deep learning ranking technique is proposed which takes the extracted features and return the rank of the corresponding query/document.

Additionally, new solutions for two known problems in the literature are provided. Namely, the dynamic query/document size problem and the rated relevance judgment loss function.

A detailed description of each method and techniques that have been used in this research component is provided in the next chapter.

3.3 Experimental Method

In this research component, several experiments are conducted in order to prove the proposed model effectiveness, efficiency, and simplicity and to analyze the effect of each component on the model performance.

This section details all the conducted experiments and explain all required data, tools, and measures that are necessary for replicating those experiments.

3.3.1 Dataset

The dataset that has been used in this study is a subset of ClueWeb-09 and ClueWeb-12. It includes all documents from both datasets that have been used in the Text REtrieval Conference[2] (TREC) web tracks from 2009 to 2014.

Table 3.1: Dataset Statistics

TREC Web Track Year	Queries IDs	Documents Number	Junk	Non	Rel	HRel	Key	Nav
2009	1-50	23601	0	16743	4832	1625	0	0
2010	51-100	25329	1431	18363	3731	1077	138	0
2011	101-150	19381	1019	15205	2038	711	408	0
2012	151-200	16055	858	11674	2208	405	52	858
2013	201-250	14474	234	10090	2281	700	179	7
2014	251-300	14432	556	8211	3788	1614	230	33
Total	**300**	**113272**	**4098**	**80286**	**18878**	**6132**	**1007**	**898**

For each year, TREC track used a different set of 50 queries (see Appendix A). The whole data from ClueWeb-09 and ClueWeb-12 were ranked for each query by different IR models and the top K results from all models were merged in one pool and manually

[2] TREC is an ongoing series of large-scale evaluation workshops for text retrieval methodologies (https://trec.nist.gov/)

judged. Thus, our dataset consists of all documents that have been manually judged for each year Table 3.1.

The manual judgment of each query/document pair is one of the following grades/categories:

1. **Nav**: the document represents a home page for the query
2. **key**: the document is a dedicated page to the query topic.
3. **Hrel**: the content of the document represents substantial information on the topic.
4. **Rel**: the content of the document provides some information about the topic
5. **Non**: the content of the page does not provide useful information on the query topic
6. **Junk**: the document is not useful for any topic.

The total queries number in our dataset is 300. The smallest query size is 1 and the maximum size is 5. The total documents number is 113272 documents. 75% of the documents are judged as not related (i.e. Non or Junk) to the corresponding query.

3.3.2 IR Evaluation Measures

In order to measure the effectiveness of the proposed model, four IR evaluation measures were used in the different experiments two for the binary judgment and two for the graded judgment:

1. **Binary judgment measures**:

   Binary judgment means that the relevance judgment that is provided for each query/document pair in the data set is either relevant or not relevant. For this type of judgment two known IR measures were used:

   a. Precision at K (P@K): for a list of ranked results, it represents the proportion of relevant documents at rank k.

b. Mean Average Precision (MAP): for a set of queries *Q*, it takes the average of all possible precisions at *k* for all possible *k* for each query and finally, it takes the average overall queries as follows:

$$MAP(Q) = \frac{1}{|Q|}\sum_{q \in Q}\frac{1}{m_q}\sum_{k=1}^{m_q} P@k(R_k(q))$$

Where $R_k(q)$ is the top k ranked results list for a query q and m_q is the number of all relevant documents for q (Manning et al., 2008, p. 129).

2. **Graded judgment measures**:

Graded judgment means that the relevance is classified into more than two categories (e.g. the six relevance judgments categories in TREC web track). For this type of relevance judgment there are two known measures:

a. Normalized discount cumulative gain (NDCG): in this measure, the gain of each retrieved document decreases logarithmically with its rank which reflects the fact that the likelihood that a user will view a retrieved document decrease with respect to its rank (Y. Wang et al., 2013).

DCG for a ranked results list at position *k* can be computed using the following formula:

$$DCG@k = \sum_{i=1}^{k}\frac{rel_i}{\log(1+i)}$$

Where rel_i is the graded relevance judgment for the result at position *i*.

In order to normalize the DCG, it is divided by the ideal DCG which is computed for the sorted list of relevance judgments at the same position.

b. Expected Reciprocal Rank (ERR): which measures the expected time that the user may spend to get a relevant document from the retrieved list (Chapelle, Metlzer, Zhang, & Grinspan, 2009).

ERR can be computed for a ranked results list at position k as follows:

$$ERR@k = \sum_{i=1}^{k} \frac{1}{i} P(rel_i) \prod_{j=1}^{i-1} (1 - P(rel_j))$$

Where $P(rel_i)$ is the graded relevance judgment for the result at position *i* transformed into a probability value.

3.3.3 Baseline Models

Three known baseline models were used in this work in order to evaluate the proposed model effectiveness and efficiency.

1. BM25: as the most known traditional IR model which uses a hand-crafted formula incorporating term frequency and term importance.
2. MatchPyramid: as the original interaction-based IR model using a one-layer CNN for features extraction and a DNN for ranking.
3. CoPACRR: as the state-of-the-art model for interaction-based IR incorporating multiple CNN's with different kernel sizes for features extractions, contextual information and term importance as additional input, and DNN as a ranking network.

Even though there are many other interaction-based models, it is not necessary to include all of them in experiments to prove the proposed model effectiveness for two reasons: First, as mentioned, CoPACRR is the state-of-the-art model which its performance has been compared with most of the proposed models. Second, in our evaluation, we are following a similar experimental design to what has been used by CoPACRR using the same dataset.

3.4 Effectiveness Experimental Design

The purpose of this experiment is to prove the effectiveness of the proposed model by comparing its performance to the above baseline models. Since the dataset is collected from TREC web tracks, for each year there is a different set of 50 topics and a judgment file which contains all the relevance judgments for each topic. For simplicity, we group each topic with all the documents that appeared in the relevance judgments for that topic along with the relevance judgments in one component and call it a *topic evaluation set*.

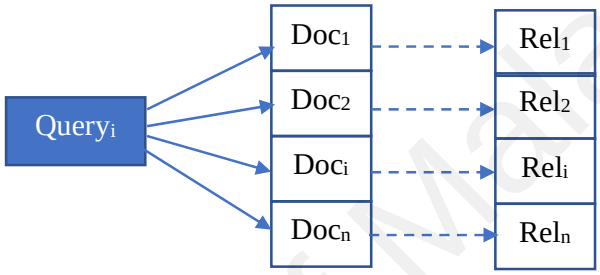

Figure 3.2: Topic Evaluation Set

Each evaluation set consists of a query, more than 300 documents and the corresponding manual relevance judgments for each query/document pairs in that set Figure 3.2. Accordingly, the main idea of this experiment is to use a model (i.e. our proposed model or one of the baseline models) to rank the documents list in each topic set and evaluate the effectiveness based on the corresponding relevance judgments. It is worth mentioning that topics sets are kept intact in all our experiments.

That said, at first, the data set is divided into five years (250 topic set) training/development dataset and one-year (50 topic set) evaluation data. The training data set is then randomly divided into 85% training dataset and 15% development dataset. Afterward, the training data is used to train the input model for N iterations where each iteration represents a complete pass over the whole training dataset. After each training iteration, the development dataset is used to evaluate the model and the evaluation results are stored for reference. The number of iterations for each model is different and it is based on the recommended settings for each model. For example, for CoPACRR 100

iterations have been used and for MatchPyramid 30 iterations while for our model, only 20 iterations have been used. After the training/development phase is finished, the development results are used to pick up the best model overall training iterations and that

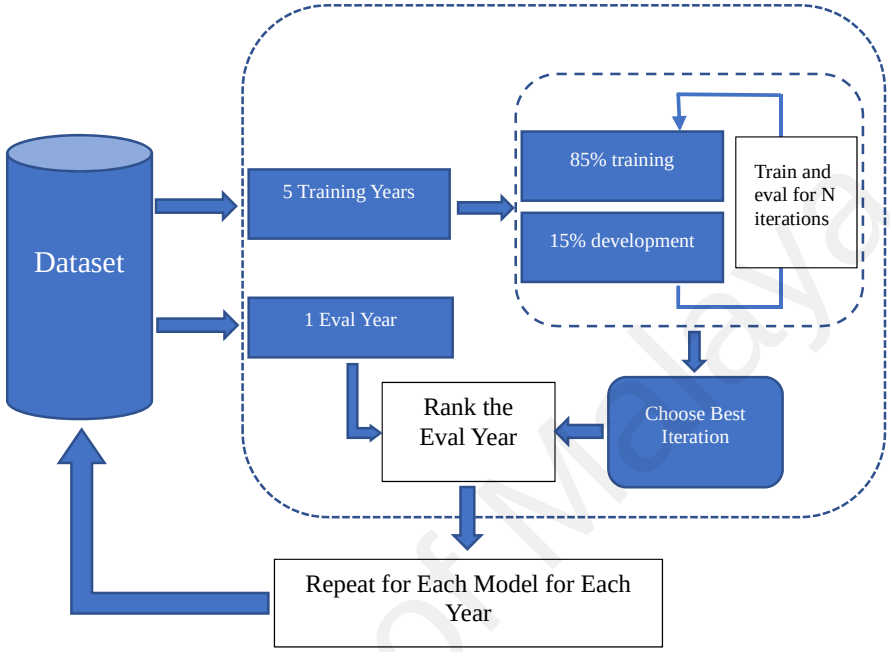

Figure 3.3: Effectiveness Experimental Design

model is then used to rank the evaluation dataset Figure 3.3.

The evaluation procedure for the development and the evaluation sets is the same. The input model is used to rank the documents list for each topic in the set. After that, the ranked list and the corresponding relevance judgments are used as input for one of the considered IR evaluation measures. Consequently, for each query in the input set, there will be four scores (i.e. NDCG, ERR, MAP, P@K) and the average over the whole queries in the set is used as the final evaluation results.

The previous training/evaluation process is repeated for each year which means that there will be six different results sets for each model.

It must be noted that unlike (Hui et al., 2017a, 2018), the topic query is used rather than the topic description for all models during training and evaluation.

Finally, since BM25 is based on a handcrafter formula, there is no training/development phase for this model. That is, the model is directly used to evaluate the evaluation set for each year.

3.5 Features Effectiveness Experimental Design

This experiment aims to analyze and understand the effectiveness of each extracted feature in the proposed model (for more information on model features see Section 4.3). To this end, a process similar to the above training, development and evaluation process is used to evaluate the proposed model with two differences:

1. One feature will be off each time.
2. The experiment is executed using: data of four years for training, one-year (TREC web 2013) for development and one-year for evaluation (TREC web 2014).

Unlike the above experiment, the experiment will not be repeated for all years and it will be done only on our proposed model.

We used the last two years of TREC web tracks in this experiment because the topics collection is more diverse (i.e. contains a mixture of broad and specific topics) in comparison to the previous years.

3.6 Efficiency and Simplicity

The purpose of this experiment is to prove the proposed model efficiency and simplicity in comparison to the baseline models.

To this end, model efficiency is measured by two factors:

1. The average time for one iteration training
2. The average execution time for the entire evaluation set.

For both factors, a server of 4 Cores and 16 GB RAM is used for all models.

In the other hand, model simplicity is measured by the used networks types, network depth and the number of the overall parameters in each model.

3.7 Conclusion

This chapter shows that two components research design is followed in this work. The first one is using a theoretical method for features identifying, extracting and ranking while the second one is using an experimental method to prove the effectiveness and the efficiency of the proposed model.

While the details of the theoretical component are left into the next chapter, this chapter focused more on the experimental method design and provide enough description of all the methods, measures and data that are necessary for replicating those experiments.

Chapter 4: The Proposed Neural IR Model Design

This chapter provides a detailed description of the theoretical methods that have been used to answer our first four research questions Section 3.1. First, it starts by giving an example of the interaction matrix. Afterward, it explains the tools and the techniques that have been proposed and used to analyze the interaction matrix in order to identify a set of features. After features identification, the chapter goes on to present our proposed approach for features extraction and give a detailed description of the proposed ranking model. Finally, a detailed description is provided for our extension of the max-margin loss function.

4.1 Interaction Matrix

As mentioned in Section 2.4.3, the interaction matrix is a structure that reflects the interactions or the matches between the given query and document. In order to introduce the interaction matrix and clarify its structure and features, the following query and documents will be used as an example in this chapter.

- **Query**:

  "*U.S. election meddling*"

- **Document**:

  "*The American president claimed his Russian counterpart was "extremely strong and powerful in his denial" of any election meddling … Trump said that he actually accepts the intelligence findings that Russia meddled in the U.S. political race*"

The first step in constructing the interaction matrix is to clean the text by removing stop words, special characters, and numbers. Thus, the previous query/document becomes as follows:

- **Query**:

  "*US election meddling*"

- **Document**:

  "*American president claimed Russian counterpart extremely strong powerful denial any election meddling Trump said actually accepts intelligence findings Russia meddled US political race*"

Unlike traditional IR indexing methods, most of the neural IR approaches that use the word embedding for word representation do not use any analyzer or stemmer to remove words postfixes or suffixes because different word forms are represented in the word embedding space as a separated vector.

Word2Vec was used in this work as word embedding resource which represents each word by 300 dimensions vector and the Cosine function between the words vectors is used as a similarity measure (Goldberg & Levy, 2014; Tomas Mikolov, n.d.).

As explained in Section 2.4.3.1, for constructing the interaction matrix, all query terms and document words are replaced by their word embedding vectors. Then the query terms vectors are taken one by one and the similarity between this term and all document words are computed and stored in a new row Table 4.1.

As Table 4.1, depicts each cell in this matrix represents the semantic similarity of one query term and one document word. As such, the length of the matrix equals to the document length after the cleaning stage and the matrix width (i.e. rows number) is the number of query terms after the cleaning stage as well.

Table 4.1: Interaction Matrix

	American	president	claimed	Russian	counterpart	extremely	strong	powerful	denial	election	meddling	Trump	….	Russia	meddled	US	political	race
US	0.82	0.63	0.43	0.56	0.35	0.22	0.31	0.25	0.31	0.48	0.27	0.58	…	0.56	0.42	1	0.51	0.32
election	0.52	0.61	0.32	0.51	0.34	0.13	0.21	0.17	0.16	1	0.57	0.54	…	0.51	0.57	0.53	0.54	0.63
meddling	0.35	0.31	0.21	0.54	0.16	0.21	0.11	0.13	0.22	0.51	1	0.51	…	0.46	0.95	0.36	0.27	0.21

Specifically, three types of cells can be recognized in the interaction matrix:

1. **Exact match Cell**: Each cell of value equals or is very close to the maximum similarity value (e.g. yellow cells). Since the Cosine function was used as a similarity measure, the maximum similarity value in our case is 1.

   Values that are very close to the maximum similarity value represent the similarity of two different forms of the same word like <meddled, meddling>.
2. **Semantically Close Cell:** Each cell of relatively high similarity value such as <US, American> and <election, race>.
3. **Non-Related Cell:** Includes the rest of the cells that do not belong to one of the previous types.

Besides, the above matrix shows that the document words and the query terms appear in the matrix in their original order. In other words, the interaction matrix preserves the positional information which is a very important feature for advanced retrieval models.

4.2 Features Identification

As stated in Section 3.1, four IR factors are taken into consideration in this work; namely: the exact match, the semantic match, the query coverage, and the term proximity. This section explores all the methods and tools that have been used in the study to identify the set of features that reflect the aforementioned factors in the interaction matrix.

4.2.1 Interaction Image

Since most of the available works for interaction based neural IR are looking at the interaction matrix as 2D image and utilizing visual patterns recognition for feature extraction, we found it more suitable to follow the same direction in analyzing the relationship between the interaction matrix and the specified IR factors. Consequently, the interaction matrix is transformed into a 2D image where each cell is represented as a pixel of color degree that reflects its similarity value

Figure 4.1.

Figure 4.1: Interaction Image, Y-axis matches with query length, X-axis matches with document length.

The similarity values range [-1, 1] is mapped into the color range [Dark Blue, Yellow] using the *Parula* color map where yellow represents exact match or high similarity value and dark blue represents low similarity value.

Further, to make the analysis process more efficient an analyzing tool (Figure 4.2) of the following functionalities was developed:

1. Transform and display a set of interaction matrices into 2D images grid.
2. Select interaction matrices using several criteria like the relevance judgment score, and size.

3. Extract several statistical information from the selected interaction matrices set such as the maximum/minimum similarity values, their frequencies, and the interaction matrix size.

Using the analysis tool for selecting and comparing the interaction images, the following sections provide a description of the obtained results of the analysis for each IR factors.

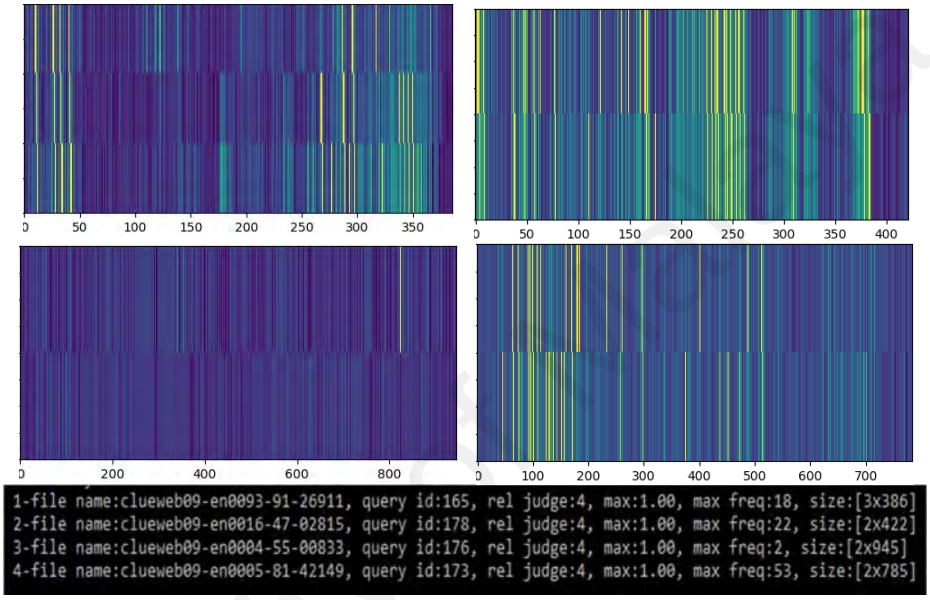

Figure 4.2: Analysis Tool Output

4.2.2 Exact (Lexical) Match

The exact match signal reflects how many times the exact lexical match of query terms appeared in the interaction image.

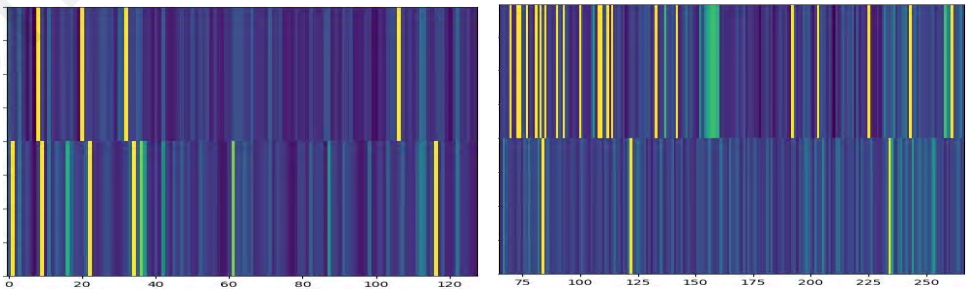

Figure 4.3: Left low exact match, Right high exact match

As Figure 4.3 depicts, the low density of yellow cells is associated with low exact matches whereas the high density is associated with high exact matches signal.

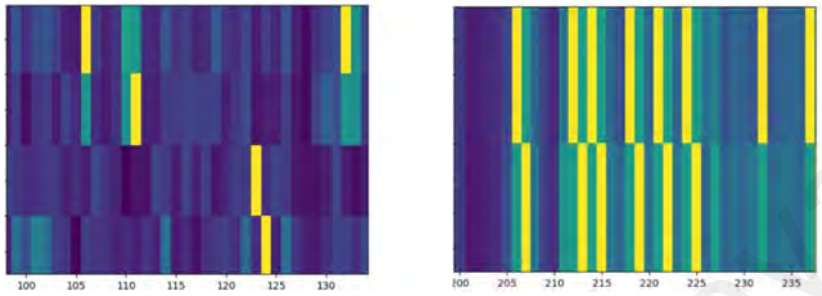

Figure 4.4: Exact Match Patterns

More importantly, we observe that the exact match does not follow any specific visual pattern. In other words, the exact match cells may be distributed in the interaction image in any form and the only thing that matter is the density of that cells Figure 4.4.

4.2.3 Semantic Match

Semantic match reflects how much the document is about the query topic. In the interaction image, cells can be divided into two types: the exact match and the background cells. The background cells correspond to the rest of the document words which do not

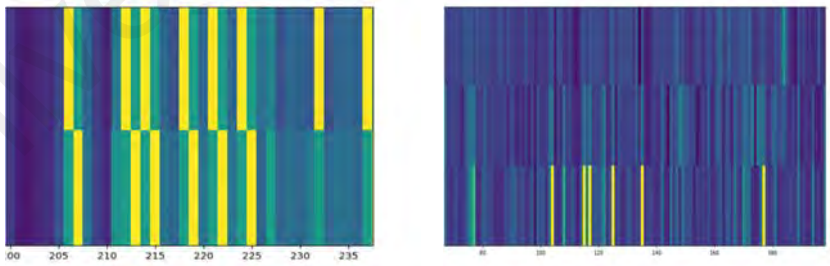

Figure 4.5: High semantic match left, low semantic match right

match any of the query terms.

By looking at the background words, it is possible to judge if the document is semantically close to the query topic or not. Furthermore, if the background cells around an exact match cell are light (Figure 4.5 left), this means that the context of the exact match word is

related to the query and if they are rather dark, the context is not related which indicates that the exact match may have a different meaning (Figure 4.5 right).

Like the exact match signal, there is no specific visual pattern for the semantic match signal and the only way to measure it is by looking at the whole interaction image color. If the image looks light (i.e. high percentage of the cells are yellow and green) then the semantic similarity is high. Otherwise, the semantic similarity is low.

4.2.4 Query Coverage

Query coverage measures how many query terms have been covered in the document. In the interaction matrix, if a query term is covered (i.e. exactly matched) it will appear as a yellow cell in the corresponding term row.

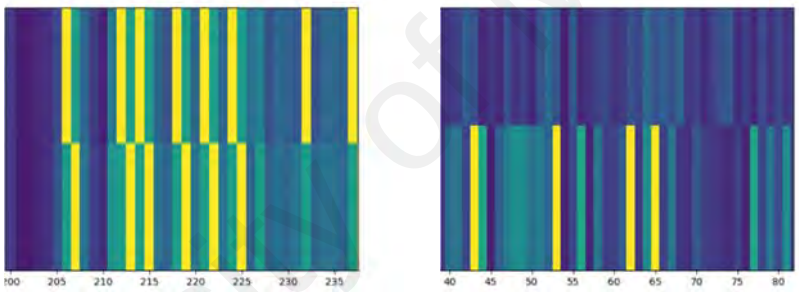

Figure 4.6: Fully Covered Left, Partially Covered Right

In Figure 4.6, there are two interaction images of two terms queries. In the left image, both terms were covered multiple times while in the right image only the second term was covered.

In order to measure query coverage, it is enough to look at the distribution of the exact matches cell over the vertical dimension.

Like the exact match and the semantic match, we have not recognized any specific visual pattern for the query coverage.

4.2.5 Term Proximity

Proximity expresses to which degree query terms co-occur in a close distance. In other words, if the document has high frequency of all query terms the exact match and query coverage may be high but this is not enough to give the document a high rank. It is necessary for the advanced retrieval models to make sure that these terms appear close together and that they are not scattered in different places.

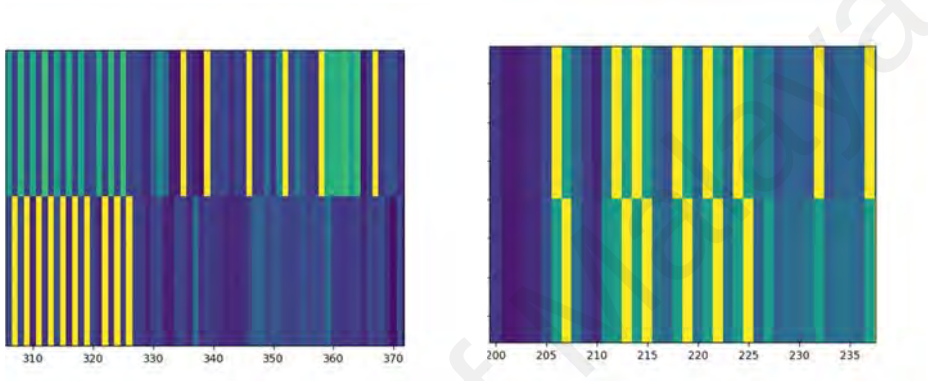

Figure 4.7: Low proximity high coverage left, high proximity high coverage right

As illustrated by Figure 4.7, in the left figure, all query terms appear in the document but the first term always appears away from the second term whereas in the right figure both terms appear next to each other.

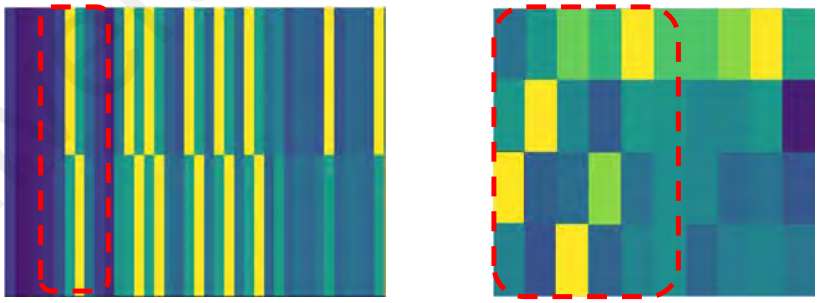

Figure 4.8: Term Proximity Patterns

If we restrict our look at a small window, and if the term following the same query order in that window, term proximity appears as a stair of successive yellow cells (Figure 4.8, left). However, this pattern is not fixed since query terms can appear in the document in any order (Figure 4.8, right).

4.2.6 Interaction Features and the Extraction Approach

Interaction features is a new term that has been coined in this work to refer to any feature in the interaction matrix that may reflect one or more of the selected IR factors in this work.

The most important result of the previous analysis is that none of the specified IR factors follows any specific visual pattern inside the interaction matrix. Furthermore, through our analysis of many different interaction matrices from different ranking categories, we could not specify any fixed visual pattern that may reflect more than one IR factors.

Based on the above analysis, the following interaction features have been specified for the specified IR factors:

1. The density of cells with high similarity values corresponds to the exact match
2. The density of cells with medium or high similarity values corresponds to the semantic match
3. Distribution of the exact match cells over the vertical dimension corresponds to the query coverage
4. The density of exact matches within close distance corresponds to the term proximity.

The previous findings are very important because differently from the majority of the previous works they suggest that the deep learning techniques for visual patterns are not the most effective methods for interaction features extraction. That is, all deep learning techniques are designed to extract fixed patterns (e.g. circles, lines, cross lines) using a fixed set of filters; whereas, the findings suggest that all the interaction features are based on the density and the distribution of some values which cannot be measured or extracted using fixed filters no matter how complicated they are.

Additionally, in particular, the term proximity feature can only be captured within a small window size and since the other features depend on cumulative features like density and distribution, we argue that it is more effective to extract all the interaction features within a small window size. We even argue that this features extraction approach is more informative and it goes in accord with the local and global relevance point of view.

Based on the previous argument, our interaction features extraction approach can be summarized by the following points:

1. Deep learning techniques for visual patterns are not suitable for interaction features extraction task.
2. Instead, a set of manually designed functions will be proposed to extract all the required features.
3. The interaction matrix will be divided into a set of successive small size windows and all the features will be extracted for each window independently as a measure for the local relevance in the corresponding position.
4. The ranking model will take the list of the local relevance and estimate global relevance.

4.3 Features Extraction

As stated above, instead of using deep learning techniques for interaction features extraction a set of manually designed functions is proposed to extract the features from a small size window. This section provides a detailed description of the extraction procedure and each feature extraction function.

4.3.1 Extraction Procedure

Based on the proposed extraction approach, the interaction matrix is partitioned into small successive windows called the contextual windows. Since the purpose of the contextual window is to look for the local relevance it is more meaningful to set the window size to

match the average sentence size in the target language. However, we chose to leave the window size and the partitioning steps as a model parameter and several options were tested.

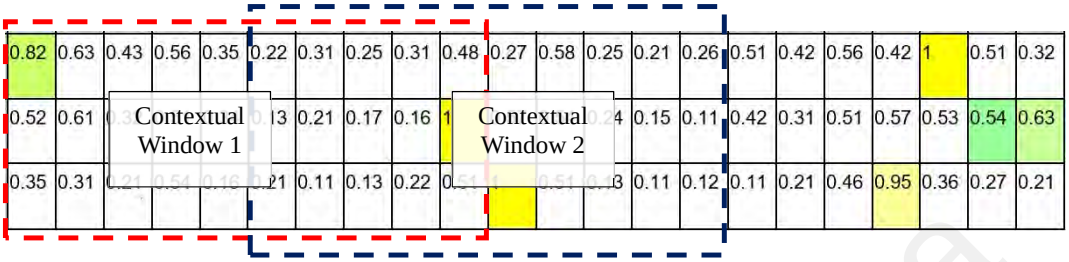

Figure 4.9: Contextual windows of size 10 and step 5

Figure 4.9 shows an interaction matrix and two contextual windows of size 10 and of step size 5.

The number of the contextual window for the given interaction matrix depends on the size of the corresponding document. After constructing the contextual windows list, each window will be passed independently into the following extraction functions to extract the interaction features.

4.3.2 Proximity & Exact Match Function

As stated in the identified interaction features, for measuring the exact match signal we should look at the density of the exact matches cells. Similarly, for measuring term proximity we should look at the density of the exact matches within a predefined distance. Given that the interaction matrix is divided into contextual windows and the features are being extracted for each window, counting the number of the exact cells within the window is enough to measure both the exact match and the term proximity.

$$P(W_i) = \frac{Number\ of\ exact\ matches}{Number\ of\ cells} \quad (4)$$

Equation (4) is used to compute the proximity and exact match feature from the input window where the numerator is the number of exact matches and the denominator is for normalization.

For counting the exact matches, we use the following criterion:

$$cell\ value \geq \alpha$$

By choosing a proper value for α , the function not only considers exact matches but also it considers synonyms and other semantically close words. This extends the meaning of this function beyond the exact matches and term proximity to hold some semantic matching value.

By looking at the results of this function for all the contextual windows, it is possible to get an accurate estimation of the exact match signal and its distribution over the document. Besides, it gives a good estimation of term proximity over the whole document.

However, depending only on the exact matches is not enough for detecting terms co-occurrence because the exact match does not distinguish between term repetition and terms co-occurrence. That is, whether the same term repeated N times or N terms co-occurred next to each other the value will be N. Consequently, this function alone does not reflect the proximity feature completely.

4.3.3 Query Coverage Function

According to the identified features, query coverage can be estimated by looking at the distribution of the exact match cells over the vertical dimension in the contextual window. This is equivalent to the number of the rows in the window which contains at least one exact match cell.

$$Cov(W_i) = \frac{Rows\ number\ containing\ a\ cell\ value > \alpha}{rows\ number} \qquad (5)$$

For normalizing the result, the rows count is divided by the total rows number which equals to the query size Equation (5).

Similar to the proximity function, the exact match criterion extended to includes other semantically close words.

Even though this function is intended to measure the query coverage feature, along with the previous function it gives an accurate estimation of term proximity. For example, if the previous function gives a high value while query coverage is low, this indicates that the term proximity is low in the input window. This case happens when most of the exact matches cells are for one term.

4.3.4 Semantic Match Function

The semantic match is proportional to the density of cells with medium or high similarity values. We argue that it is enough to estimate this feature by taking the average of all cells in the window. That is, each cell in the interaction matrix is representing the semantic similarity of one query term and one document word and therefore it is expected that the average of all semantic similarities should give a good estimation of the semantic match between the query and the document Equation (6).

$$Sim(W_i) = \frac{Sum\ of\ all\ cells\ values}{cells\ number} \tag{6}$$

Although it is possible to come up with more advanced extraction methods to make the extraction more accurate or to extract other features, we think that these three simple extraction functions are enough to build an advanced ranking model that is able to extract and integrate the selected set of IR factors. As mentioned before, the mapping between the proposed functions and the four IR factors (i.e. exact match, semantic match, query

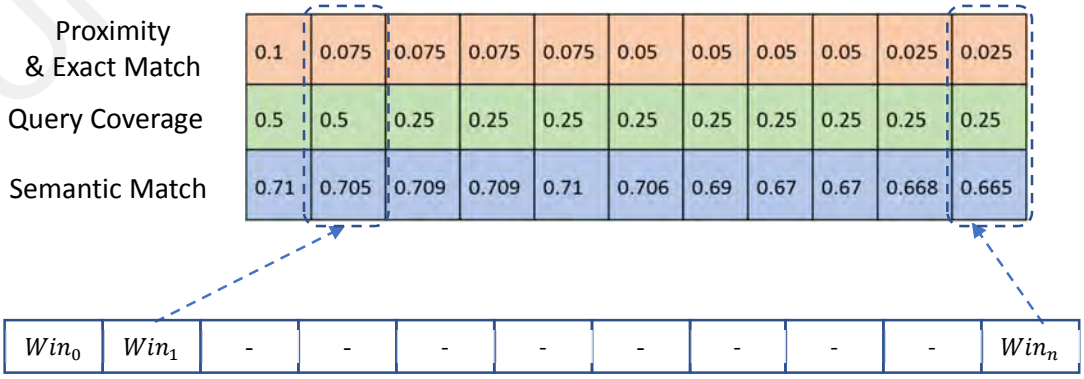

Figure 4.10: Features Extraction Results

coverage, and term proximity) is not one. Instead, it expected that the three proposed function together are able to reflect the intended IR factors.

At the end of the extraction phase, for each interaction matrix, there will be three features vectors one for each function. The length of each features vector equals the number of the contextual windows in the corresponding interaction matrix (Figure 4.10).

4.4 Ranking Model

After extracting the interaction features, these features are passed into the ranking model to get the final score for the corresponding query and document. The proposed ranking model consists of four components.

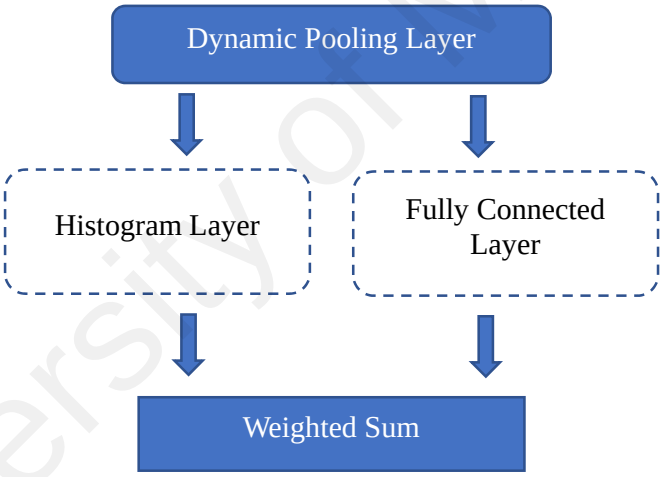

Figure 4.11: Ranking Model

The pooling layer is responsible for unifying the size of the feature. After that, the unified features are passed into two ranking layers: a fully connected and the histogram layer where each layer gives a different score for the same features. Finally, the two scores are combined using a simple weighted sum layer to get the final score. The following sections, describe each of the above component Figure 4.11.

4.4.1 Dynamic Spatial Max Pooling

Before passing the extracted interaction vectors into the neural ranking model, the size of these vectors must match the input size of the neural model. As mentioned in the literature review this restriction is still an open problem and most of the studies are using the zero padding and cutting techniques which may result in either loosing relevant or padding irrelevant information.

Specifically, two cases can be distinguished: the first one when the feature vector is larger than the neural model input size and the second one when it is smaller.

Our solution used a different technique for each case. In one hand, when the features vector is larger than *k* where *k* is the intended input size, the vector is divided into *k* spatial partitions and each portion is then reduced to the max feature value Figure 4.12.

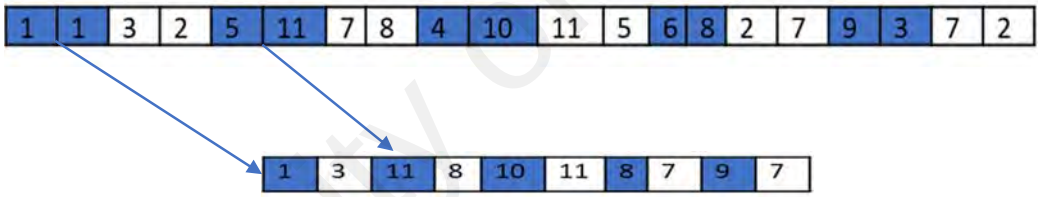

Figure 4.12: Spatial Max Pooling

This technique is called spatial pooling and it was originally used for computer vision deep learning models to reduce the variable size features matrix into fixed size representation.

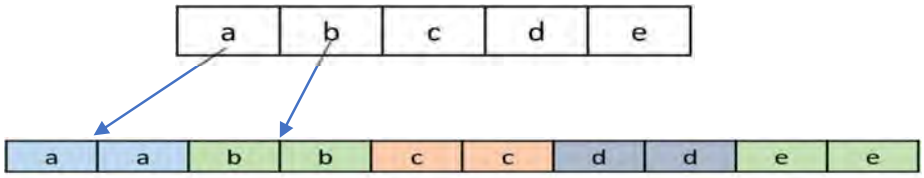

Figure 4.13: Features Up Sampling

In the other hand, when the feature vector is shorter, an empty vector of the intended size is created and divided into *l* spatial partitions where *l* is the original vector size and

each partition is filled with the same value from the corresponding position in the original vector Figure 4.13.

4.4.2 Fully Connected Layer

The model consists of two fully connected layers (see Section 2.3.1). The first layer takes the concatenation of the three features vectors as input and returns a new vector of predefined size M. An activation function is applied to the first layer output and the result is then fed into the second layer which returns a single value as the final score.

$$\begin{aligned} X &= concat(V_1, V_2, V_3) \\ L_1(X, \theta) &= W_1 X + b_1 \\ h &= \sigma\,(L_1) \\ S_1 &= L_2(h, \theta) = W_2 h + b_2 \end{aligned} \tag{7}$$

Where L_1 is the first layer, W_1 is first layer weights matrix and b_1 is the bias.

σ is the activation function.

L_2 is the second layer, W_2 and b_2 are the second layer weights and bias.

As an activation function, several functions were tested like *Segmiod*, *Tanh* and *Relu*. And after several tests, *Relu* was able to give the best results.

Additionally, we run several experiments with different numbers of fully connected layers and with different numbers of layers sizes. It turns out that increasing the network depth by stacking additional layers only improves the performance on the training data set whereas it leads to poor performance on both the development and the evaluation data sets. This can be attributed to the fact that increasing the network depth will allow the model to overfit the training data and thus give a bad performance when used on a different data set.

4.4.3 Histogram Model

The main problem of the above fully connected layer ranking model is that each unit in the hidden layer will eventually learn to look only at specific positions in the features vector to calculate its output. While this paradigm is suitable for positional classification tasks like image classification, we argue that it is not effective for ranking tasks where features position is not important. That is, while positional information is important for local relevance (i.e. at the level of one contextual window), it does not hold any additional information for the global relevance.

In response, we proposed the histogram model which represents the distribution of all windows scores as a histogram and trains a simple neural network to rank that histogram.

The intuition behind the histogram model is that by looking at how many contextual windows got high relevance score and how many got low scores, it is possible to estimate the score of the whole input interaction matrix. To this end, the proposed histogram model consists of three components: window scorer, histogram constructor, and histogram ranking network.

1. Window Scorer:

   It is a simple neural network (Section 2.3.1) which for each window take the corresponding three features (i.e. proximity, query coverage, and semantic match) and return a normalized score using the following equation:

   $$Score(W_i) = Sigmoid(Y\ W_i) + b \quad (8)$$

   Where Y is a vector of three weights, W_i is the three features of the window i and b is the bias value. The sigmoid function is used for returning all scores into value form [0 1].

2. Histogram Constructor:

After computing all windows scores the score range (i.e. [0 1]) is divided into k equal interval and each window score is accumulated into the corresponding interval.

3. Histogram Ranking Network:

   The constructed histogram is then fed into a fully connected network of two layers (Section 2.3.1) to estimate the overall score.

Even though our analysis suggests that the fully connected layer is not suitable for the ranking task, the experiments do not support that analysis. In particular, we found that the fully connected layer ranking model gives better performance than the histogram ranking model in almost all experiments. However, we found that combing both ranking models give even better performance.

4.5 Loss Function

In order to train the proposed neural model, it is necessary to use an effective loss function which uses the difference between the model output and the intended output to fix the network by updating its weights. As shown in Section 2.6, two loss functions have been used in the literature; namely: the cross-entropy and the max margin. Besides, it was shown that these two functions are used for binary relevance judgment and that they are not suitable for the graded relevance judgment.

The following section explains our extension for the max-margin loss function to be more suitable for the graded relevance judgment.

4.5.1 Graded Max-Margin

The max margin equation (shown in Section 2.5.2) states that the distance between positive document score and negative document score for the same query should be at least 1. While this is enough for binary relevance judgment, lots of useful information will be discarded if the same method used for graded relevance judgment. In other words,

when there are N different relevance grades, there will be $\frac{Nx(N-1)}{2}$ possible cases of unmatched judgments and it will be more accurate to have different distance threshold for each case. To this end, we suggested a new version of the binary max margin (Equation (3)), called the graded max margin, which simply states that the minimum distance between the scores of two documents for a given query should increase or decrease in accordance with the distance of their relevance judgments.

Using the same terminology that has been used in Section 2.5, for a given training sample (q, d^+, d^-) first, we define the normalized distance between the corresponding relevance judgment of $d^+$and d^- using Equation (9).

$$GD(q, d^+, d^-) = \frac{judge(q, d^+) - judge(q, d^-)}{Max_j - Min_j} \quad (9)$$

Where $judge(q, d^+), judge(q, d^-)$ are the positive and the negative relevance judgments grades mapped into numerical values (in our case it will be between 0 and 5).

Max_j , Min_j are the maximum and the minimum possible grades values.

Then, the above distance is used to extend the max-margin loss as shown in Equation (10).

$$MaxMargin(q, d^+, d^-) = \max\left(0, GM(q, d^+, d^-) - SD(q, d^+, d^-)\right) \quad (10)$$

$$GM(q, d^+, d^-) = \beta + GD(q, d^+, d^-) \quad (11)$$

$$SD(q, d^+, d^-) = Score(q, d^+) - Score(q, d^-) \quad (12)$$

Where β is predefined offset.

$Score(q, d^+)$ and $Score(q, d^-)$ are the output of our model for both query/document pairs.

The previous loss formula means that the distance between the given positive and negative scores should be greater than a given offset plus the normalized distance of the

two documents relevance judgments. In this way, the neural model learns to distinguish between all possible judgment grades and thus gives more accurate scores.

4.6 Conclusion

The relation between the considered IR factors in this study and the interaction matrix have been identified in this chapter as a set of features called the interaction features. Additionally, the current chapter provides a detailed description of the proposed features extraction approach and of the proposed ranking model.

Chapter 5: Results and Findings

This chapter presents and discusses the results of the experiments described in Chapter 3: using the proposed model that has been explained in Chapter 4: As stated in Sections 3.4, 3.5 and 3.6 there are three different experiments that aim at evaluating model effectiveness, analyzing model components and evaluating model efficiency and simplicity. Following the same structure that has been presented in Chapter 3:, each experiment results are presented and discussed respectively.

5.1 Effectiveness Experiment

This experiment intends to prove the proposed model effectiveness by comparing its performance to several baseline models. Following the experimental design that have been shown in Section 3.4, this section presents the results of our proposed model along with the results of the considered baseline models. For simplicity, we chose to present the results of each evaluation measure separately.

Results structure is explained only for the first measure as the same structure is followed for the rest of the measures.

5.1.1 ERR[3] results

As shown in Table 5.1, there are six results for each model one row for each TREC web track year which matches with the evaluation dataset that has been used for each experiment. Besides, the overall average is provided in the last row. The last column represents the results of our proposed model (i.e. IF_NIR). The result in each cell represents the ERR score for the corresponding model on the corresponding TREC track

[3] For ERR, NDCG and P@K measures, which work for specific ranking threshold we chose to use ranking level 20 as ranking threshold for the three measures.

dataset. The significant results are distinguished by * which indicates that the p-value of the paired t-test over means difference is less than 0.05.

Table 5.1: ERR@20 results. * indicate a significant difference. The last column shows our proposed model results.

	BM25	MatchPyramid	CoPACRR	IF-NIR
TREC web 2009	0.035*	0.125*	0.107*	0.151
TREC Web 2010	0.071*	0.118*	**0.152**	**0.142**
TREC Web 2011	0.12	0.151	**0.157**	**0.14**
TREC Web 2012	0.11*	0.276	0.285	0.303
TREC Web 2013	0.121*	0.165	**0.18**	**0.178**
TREC Web 2014	0.165*	0.177*	0.222	0.231
Average	0.104*	0.169*	0.184	0.191

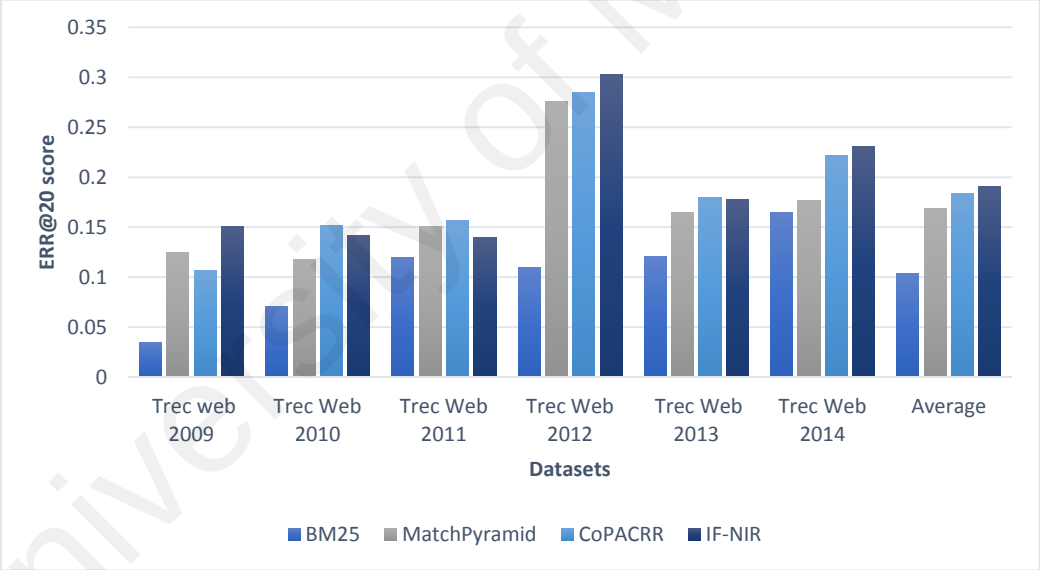

Figure 5.1: ERR@20 results histogram

Figure 5.1 represents the same results as a histogram where for each TREC web track dataset there are four columns each of which represents the result of one model.

The results show that our model was able to outperform almost all baseline models in almost all years. Specifically, the proposed model significantly outperformed BM25 and MatchPyramid in all years according to ERR evaluation measure. However, although

IF_NIR was able to significantly outperform CoPACRR in some years, the difference was not significant in almost all years.

5.1.2 NDCG Results

Similar to the above results, Table 5.2 and Figure 5.2 shows that our proposed model was able to outperform almost all baseline models in almost all years and in the overall average for NDCG evaluation measure. However, like the ERR results, the difference between our results and the CoPACRR was not significant in most of the years.

Table 5.2: NDCG@20 Results. * indicate a significant difference. The last column shows our proposed model results.

	BM25	MatchPyramid	CoPACRR	IF-NIR
TREC web 2009	0.067*	0.24*	0.202*	**0.279**
TREC web 2010	0.137*	0.2*	**0.259**	0.245
TREC web 2011	0.285	0.27	**0.32**	0.291
TREC web 2012	0.098*	0.196	0.212	**0.223**
TREC web 2013	0.207*	0.281	**0.328**	0.305
TREC web 2014	0.273*	0.284*	0.324	**0.335**
Average	0.178*	0.246*	0.274	**0.279**

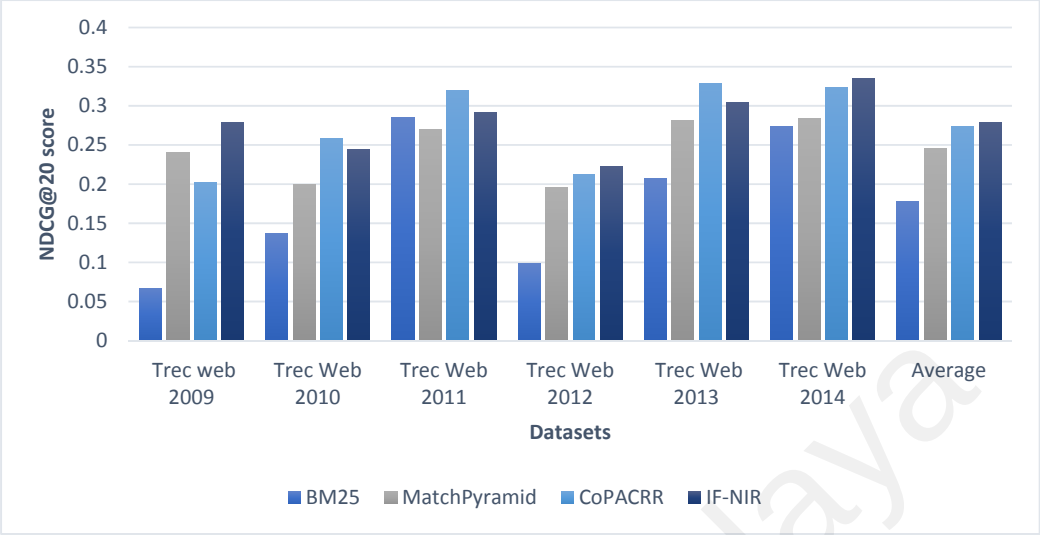

Figure 5.2: NDCG@20 Results Histogram

5.1.3 P@k Results

Table 5.3: P@20 Results. * indicate a significant difference.

The last column shows our proposed model results.

	BM25	MatchPyramid	CoPACRR	IF-NIR
TREC web 2009	0.138*	0.399*	0.353*	**0.475**
TREC web 2010	0.29*	0.4*	**0.484***	0.448
TREC web 2011	**0.381**	0.344	0.372	0.371
TREC web 2012	0.255*	0.345*	0.356	**0.405**
TREC web 2013	0.332*	0.472	0.48	**0.489**
TREC web 2014	0.5250	0.49*	**0.566**	0.556
Average	0.32*	0.408*	0.435*	**0.458**

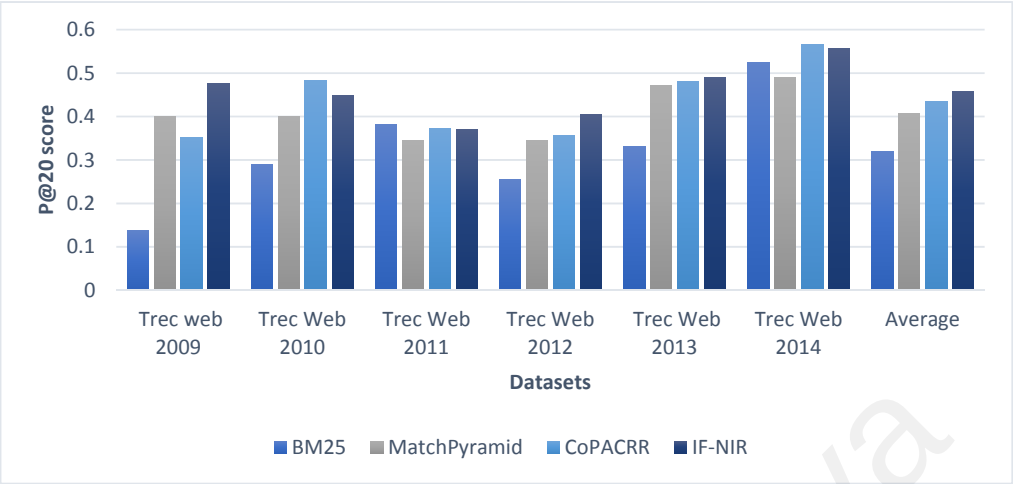

Figure 5.3: P@20 Results Histogram

Table 5.3 and Figure 5.3 show that, like ERR and NDCG, P@K results indicate that our model outperformed all models in almost all years and in the overall average. Also, the above results are compatible with the previous results in that the difference between our model and CoPACRR is not significant in most of the results.

5.1.4 MAP Results

Differently, from the results of the previous measures, MAP results illustrated in Table 5.4 and Figure 5.4, indicate that our model outperforms other models only in two years. However, the results also showed that the difference between our model and all baseline model is not significant in almost all years.

Table 5.4: MAP Results. * indicate a significant difference.
The last column shows our proposed model results.

	BM25	MatchPyramid	CoPACRR	IF-NIR
TREC web 2009	0.327*	0.389	0.365*	**0.408**
TREC Web 2010	0.35	0.367	**0.419**	0.388
TREC Web 2011	0.366	0.307	**0.384***	0.323
TREC Web 2012	0.315*	0.337*	0.344	**0.373**
TREC Web 2013	0.374*	0.416	**0.437**	0.434
TREC Web 2014	**0.5334**	0.507	0.5328	0.514
Average	0.378	0.387	**0.414**	0.407

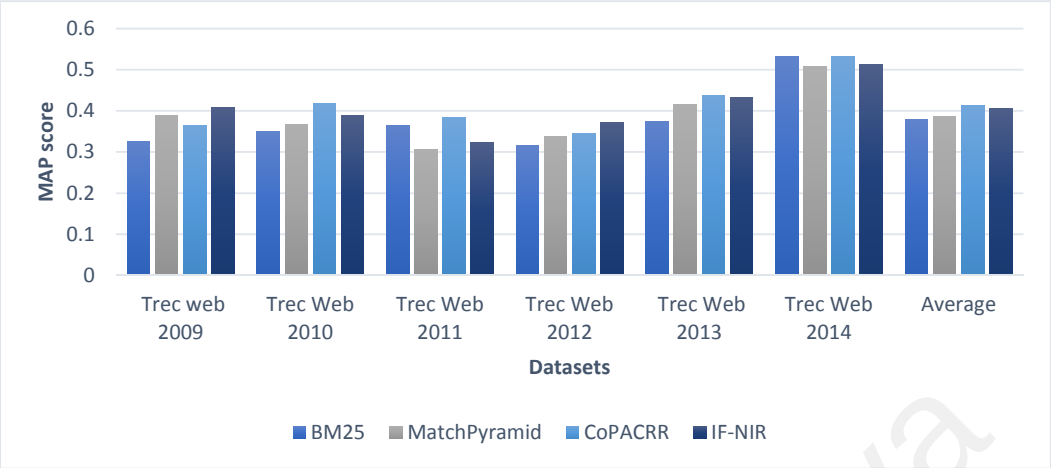

Figure 5.4: MAP Results Histogram

5.1.5 Performance Experiment Discussion

It is clear from the above results that the proposed model significantly outperformed BM25 in all years and in almost all evaluation measures which alone is a great indication of the effectiveness of our approach. This indication was reinforced by comparing our model performance to MatchPyramid which also showed that our model was able to outperform MatchPyramid in almost all years and all measures. Apparently, even though our model outperformed the state-of-the-art model CoPACRR in several years, the difference was not significant in most of the cases. To a certain degree, this close performance challenges our findings in the interaction features analysis which states that deep learning techniques for visual pattern recognition are not suitable for interaction features extraction. There are three possible reasons explaining the above issue:

1. The extraction approach is not effective for all or some of the identified interaction features.
2. The study focused only on a subset of all IR factors that have been considered in CoPACRR. Therefore, the current interaction features are not enough to outperform the state of the art.

3. The proposed features extraction approach in CoPACRR is able to extract some hidden features that are out of reach of our feature extraction approach.

While the first possibility will be investigated in the next experiment results, the second and the third are out of the scope of this study. However, supporting our model by more IR factors such as term importance and semantic disambiguation is an expected extension for our model and it will not contradict with our approach for features extraction and ranking.

Back to the results, it can be noted from Table 5.4 and Figure 5.4 that according to MAP measure all models give a relatively close performance. This can be attributed to the fact that unlike the other three measures, MAP evaluates the whole ranking list. That is, while the other measure only focuses on the first k retrieved documents (i.e. 20 in our case), MAP considers all available documents for a given query. Therefore, the above results can be understood as all models are giving a close performance to the BM25 when looking at the whole retrieved documents list for each query whereas our model and CoPACRR are giving a significantly better performance by looking at the first 20 results. However, looking at the first k retrieved documents is more compatible with web search engine requirements where users usually do not go further than the first page of the results.

5.2 Features Effectiveness Experiment Results

This experiment was designed to analyze the effect of each proposed interaction feature on the effectiveness of the entire model. According to Section 3.5, the experiment was conducted on the TREC web 2014 using the rest of the data for training and development.

As Table 5.5 depicts, the first row is the performance of our model using all features; after that, each row represents the performance after excluding one of the interaction features. The performance was evaluated using the four IR measures that have been used in the previous experiment using the same t-test significance level (0.05).

Table 5.5: Features Effectiveness Results. * indicates a significant difference

	ndcg@20	err@20	MAP	P@20
All Features	0.31184	0.19835	0.496	0.55
Without Proximity	0.31897	0.21785	0.5005	0.535
Without Query Coverage	0.26757*	0.18031	0.4719*	0.496*
Without Semantic Match	0.28471	0.19681	0.4927	0.518

5.2.1 Features Effectiveness Experiment Discussion

The first thing that can be noticed from the above results is that the query coverage feature significantly affected the model performance in comparison to the other features. More specifically, this significant effect appeared only in three measures; namely: NDCG, MAP, and P@K. This result confirms the importance of the query coverage feature and proves that the proposed approach for that feature extracting is successful.

For the semantic matching feature, even though the effect was not significant, by looking at the corresponding p-values of the significance test shown in Table 5.6, it is possible to observe that for NDCG and P@K the value is relatively small which indicates that the effect of the feature is marginally significant. This result along with the argument that has been provided in Section 5.1.5 indicates that the proposed method for the semantic matching feature extraction may need to consider other factors like semantic disambiguation. In a broader context, this marginal significance can be attributed to the defects of the used word embedding or the used word similarity function.

Table 5.6: Significance Test P-Values

	ndcg@20	err@20	MAP	P@20
Without Proximity	0.672407	0.405249	0.531912	0.413751
Without Query Coverage	0.006721	0.369957	0.023586	0.013164
Without Semantic Match	0.09501	0.935345	0.631253	0.101155

On the other hand, unlike query coverage and semantic matching, this experiment showed that the term proximity feature has no effect on model performance. This result gives a great impression that the proposed method for term proximity extraction is not successful. This implication may explain why our proposed model was not able to significantly outperform the state of the art as shown in the previous experiment. However, further experiments are needed to confirm this conclusion.

5.3 Efficiency Experiment Results

This experiment aims at proving our model efficiency and simplicity in comparison to the other interaction-based neural model. As specified in Section 3.6, the comparison is done on two factors the first one estimates the time complexity while the other is measuring the model size and structural complexity.

For the first factor, Table 5.7 shows the average training and the execution time for our model, MatchPyramid and CoPACRR. BM25 is not included in this experiment since it is not a neural model.

Table 5.7: Training/Execution Time Results

	One Training Iteration Time	Execution Time
IF-NIR	15 min	4 min
CoPACRR	56 min	40 min
MatchPyramid	21 min	7 min

For the second factor, networks types, layers number, and the number of all model parameters are provided in Table 5.8.

Table 5.8: Model Simplicity Results

	Networks Types	Layers Number	Total Model Parameters
IF-NIR	DNN	4	2288
CoPACRR	4 CNN, DNN	6	18368
MatchPyramid	CNN, DNN	5	4980

5.3.1 Efficiency Results Discussion

The above results demonstrate that our model is more efficient and much simpler than both CoPACRR and MatchPyramid. This result is very interesting because MatchPyramid is one of the simplest interaction-based models and CoPACRR is the state-of-the-art performance. Adding this result to the performance experiment results, show that our model with a very simple and shallow neural model was able to give a close performance to the state-of-the-art model.

It is clear that our approach for interaction features extraction was the main reason behind our model simplicity and efficiency. That is, both CoPACRR and MatchPyramid are exploiting a wide range of convolution filters to extract all possible forms of the interaction features. These filters are the heaviest and most complicated components in both models. Instead, our model explicitly utilizes manually designed functions to extract a set of predefined features.

In particular, CoPACRR, as shown in the above results, is far more complex and heavier than our model. The reason behind this is that, for feature extraction, they are using four different CNN networks each of which consists of 32 filters of different kernel size. This need for a high number of convolutional filters correlates with our main findings in Chapter 4 which states that interaction features do not follow any specific visual pattern. Therefore, increasing the number and diversity of the filters is necessary to match more forms of the same feature.

5.4 Conclusion

The results of three experiments where shown and discussed in this chapter. The results of the effectiveness experiments prove that the proposed model was able to significantly outperform some baseline models and to give a close performance to the state-of-the-art model. Next, the results of the effectiveness of the features showed that while some features like query coverage significantly affect our model performance other features like term proximity surprisingly has no effect. Finally, the last experiment proves the efficiency and simplicity of the proposed model.

Chapter 6: Conclusion

This dissertation investigated the relationship between a set of common IR factors and the interaction matrix structure in the context of interaction based neural models for ad-hoc retrieval. The main outcome of this work is the identification of a set of interaction features and a new approach for extracting these features. The features and the extraction approach have been validated by developing a new neural ranking model based on the extracted features. In order to prove the effectiveness and efficiency of the proposed model, several experiments have been conducted.

This concluding chapter summarizes the contributions of the work and assesses the possible implications on the available knowledge. Possible limitations and important future recommendations are provided at the end of this chapter.

6.1 Problem

Interaction based neural models is a new approach for ad-hoc retrieval which introduces the interaction between the query and the document as a matrix of words semantic similarities. The interaction structure is fed into a deep learning model that gradually learns to extract some hidden features and use them to rank the corresponding query/document. The majority of the available works are using deep learning techniques for visual pattern recognition to extract interaction features. However, until now there is no proof that the interactive features are following a specific visual pattern in the interaction matrix and therefore it is not clear if that features extraction approach is the most effective approach. Furthermore, the utilization of deep learning techniques for visual pattern recognition enforce some limitations and affect the proposed models' effectiveness and efficiency.

6.2 Solution

In order to identify what is the interaction features and what is the most effective approach for extracting them, a set of IR factors were specified at the beginning of this work. Using the specified factors, an analysis study is conducted to discover how these factors are reflected in the interaction matrix. The outcome of the analysis study was the identification of a set of interaction features and a new approach for extracting them from the interaction matrix. To prove the effectiveness of the proposed approach a new neural ranking model is developed based on the extracted features and its performance was compared to three baseline models.

6.3 Contributions

The key contribution of this study is that it is the first work to shed light and excessively investigate the relationship between the interaction structure and a set of important IR factors. This investigation led to the following implications and achievements:

a. Our analysis study for the interaction images showed that the selected set of IR factors do not follow any specific pattern in the interaction matrix. This led to the identification of an explicit set of interaction features that reflect the considered IR factors. What makes this approach different from the previous works is that the features are explicitly identified and analyzed before training any model. That is, nearly all works in the literature depend on some deep learning techniques to identify and extract some hidden features. This unsupervised approach hinders our understanding of the relation between IR and deep learning in general and therefore it was an obstacle in the way of inventing a more IR compatible deep learning model.
b. The study found that the identified interaction features are density based rather than following some specific visual patterns. Consequently, the deep learning

techniques for visual patterns recognition, which have been used by the majority of the available interaction-based models, are not effective for extracting these features.

c. A simpler procedure for feature extraction was proposed which divides the interaction matrix into smaller elements called contextual windows and instead of using heavy deep learning techniques, three functions with simple formula are used to calculate three values for each window. The first function is used to measure exact match and proximity. The second is used to measure query coverage while the third is used to measure the semantic matches.

d. Based on the above findings, a new deep ranking model that combines the histogram ranking network and DNN based ranking network.

e. To evaluate the proposed model in term of effectiveness, its performance was compared to three baseline models. The experiments show that the proposed models give comparable performance to the state of the art.

f. In the other hand, to evaluate the model in term of efficiency and simplicity, we compared the training and execution time, and the network size of our model to two deep learning baselines models.

g. The impact of each component in the proposed models was analyzed in a separated experiment.

h. The experiments showed that in addition to its effectiveness, the proposed model is relatively efficient in compression to other models that depend on deep learning techniques for features extraction which makes it more efficient for production use. Further, since the model does not depend on hidden features, it is relatively simple and interpretable which is necessary for any further academic research.

i. Finally, an extended version of the max-margin loss function was introduced in this study which allows for training neural ranking models on graded relevance judgments.

6.4 Limitations

Here we list several limitations that may have affected the study finding and implications:

a. Like almost all interaction-based modes, the study used word2vec as a word embedding to compute word to word semantic similarity. However, word2vec like other word embedding resources suffer from several problems such as out of vocabulary words and homonyms. Therefore, it is not clear how these problems may affect our features analysis and model performance.

b. The study depended on the whole collection of TREC web tracks data and relevance judgments from 2009 to 2014 for both features identification and evaluation. Even though TREC web track is one of the most robust IR evaluation resources, we noticed that the available queries are relatively short; besides, the number of the queries for each year is only 50 which is relatively small in comparison to other datasets.

c. Only three baseline models were used for the evaluation experiment. We argue that this experiment is enough to prove the effectiveness of the proposed approach because we are comparing our model performance to the state-of-the-art model introduced by (Hui et al., 2018) on the same dataset using almost identical evaluation procedure. Nevertheless, not all interaction-based models have been compared with Hui et. al. model and it is possible that small changes for the evaluation procedure significantly affect the results of some model that have been used in their experiments.

6.5 Future Works

As stated above, this study is the first work to explicitly identify the relationship between the interaction structure and a selected set of IR factors. Consequently, the findings of this work open the door for many further research and recommendations. In the following section, we summarise the most important and relevant recommendations.

6.5.1 Interaction structure reduction

Even though interaction-based models were able to significantly outperform other traditional or neural IR models, there are many doubts that they are suitable to be used in any real-world applications. In fact, all interaction-based models require the construction of the interaction structure for each new query. This means that for real-life applications we need to construct all the interaction matrices for each newly coming query with all available documents in the documents repository. No matter how effective the construction process is it needs a too long time and lots of computation resources. To alleviate this problem some works suggested using the interaction models in the telescoping mode. In that mode, the interaction-based models work behind some more efficient model (e.g. BM25) in that they are only used to re-rank its results. The problem with that approach is that if the first model fails to retrieve some relevant document the whole model fails.

Before this work, it was impossible to think of any way to reduce the interaction structure or transform it into a more efficient structure because we did not know what are the most important features that we must preserve in any reduction or transformation.

In view of our findings, we found that in particular, only two interaction features had the credit for our model performance (i.e. query coverage and semantic similarity). Therefore, it may be possible to reduce or transform the interaction matrix into a more efficient structure (e.g. indexes or vector representation) that only preserve these features.

6.5.2 Ranking Model

As clarified in (2.4.3.3 and 4.4.3) several interaction-based models depend on positional classification layers as a ranking model. Additionally, in our analysis, we suggested that global relevance does not follow any specific structure; thus, positional classification-based models are not suitable to rank global relevance. As a result, we proposed a new ranking model (i.e. the histogram ranking model) which focuses on relevance quality and quantity rather than positions. Surprisingly, the experiments show that positional classification-based models outperformed the histogram ranking model. We think that this issue deserves more investigations and that discovering the reason behind this inconsistent result may allow for developing a more effective ranking model.

6.5.3 Features extension

Based on the results and the discussions that have been provided in the effectiveness experiment in Section 5.1, we found that extending our model with other IR factors like semantic disambiguation and term importance can lead to better performance. However, integrating such factors need further analysis and experiments to find the most effective approach for each one.

6.5.4 A new method for term proximity extraction

Term proximity is one of the most challenging factors in ad-hoc retrieval because it is a combination of several features like the number of terms co-occurrence and the distance between these terms. In the proposed features extraction approach, the interaction matrix is divided into contextual windows and all exact matches in each window are counted as an estimation of term proximity. Even though this estimation alone does not completely reflect term proximity, we expected that by integrating this feature with query coverage feature the ranking model should learn to accurately measure term proximity. However, our experiments show that our approach was not effective. More experiments are needed to confirm these results and our approach for proximity extraction should be reviewed.

6.5.5 Word Embedding

As stated in the limitation section, word embedding suffer from several drawbacks like out of vocabulary words and homonyms. Recently, more advanced versions of word embedding have been published to alleviate these drawbacks (Peters et al., 2018). Consequently, it is useful to test how these new words embedding may affect our performance and perhaps our implications.

6.5.6 Further experiments

In our experiments, only three baseline models were used and all experiments have been conducted on TREC web tracks using documents form ClueWeb09 and ClueWeb12. As noted in the limitations section, each TREC web track contains only 50 queries and in general, the queries are relatively short. Hence, further experiments are necessary to confirm our finding by comparing our model to other baselines and by using different datasets.

References

Ai, Q., Yang, L., Guo, J., & Croft, W. B. (2016). Analysis of the Paragraph Vector Model for Information Retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval - ICTIR '16* (pp. 133–142). New York, New York, USA: ACM Press. https://doi.org/10.1145/2970398.2970409

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022. Retrieved from http://www.jmlr.org/papers/v3/blei03a.html

Chapelle, O., Metlzer, D., Zhang, Y., & Grinspan, P. (2009). Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09* (p. 621). New York, New York, USA: ACM Press. https://doi.org/10.1145/1645953.1646033

Collins-Thompson, K., Macdonald, C., Bennett, P., Diaz, F., & Voorhees, E. M. (2014). TREC 2014 Web Track Overview. *NIST Special Publication 500-308: The Twenty-Third Text REtrieval Conference Proceedings (TREC 2014)*, 1–15. Retrieved from http://trec.nist.gov/pubs/trec22/papers/WEB.OVERVIEW.pdf

Craswell, N., Zoeter, O., Taylor, M., & Ramsey, B. (2008). An experimental comparison of click position-bias models. In *Proceedings of the international conference on Web search and web data mining - WSDM '08* (p. 87). New York, New York, USA: ACM Press. https://doi.org/10.1145/1341531.1341545

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391.

Deng, L. (2014). a-tutorial-survey-of-architectures-algorithms-and-applications-for-deep-learning, *3,e2*.

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR, abs/1402.3*. Retrieved from http://arxiv.org/abs/1402.3722

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Nets. Retrieved from http://papers.nips.cc/paper/5423-generative-adversarial-nets

Guo, J., Fan, Y., Ai, Q., & Croft, W. B. (2016). A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16* (pp. 55–64). New York, New York, USA: ACM Press. https://doi.org/10.1145/2983323.2983769

Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences. Retrieved from http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L., … Heck, L. (2013a). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13* (pp. 2333–2338). New York, New York, USA: ACM Press. https://doi.org/10.1145/2505515.2505665

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L., … Heck, L. (2013b). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13* (pp. 2333–2338). New York, New York, USA: ACM Press. https://doi.org/10.1145/2505515.2505665

Hui, K., Yates, A., Berberich, K., & de Melo, G. (2017a). PACRR: A Position-Aware

Neural IR Model for Relevance Matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 1060–1069). Copenhagen, Denmark. Retrieved from http://arxiv.org/abs/1704.03940

Hui, K., Yates, A., Berberich, K., & de Melo, G. (2017b). RE-PACRR: A Context and Density-Aware Neural Information Retrieval Model. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Hui, K., Yates, A., Berberich, K., & de Melo, G. (2017c). RE-PACRR: A Context and Density-Aware Neural Information Retrieval Model. In *SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17)*. Shinjuku, Tokyo, Japan. Retrieved from http://arxiv.org/abs/1706.10192

Hui, K., Yates, A., Berberich, K., & de Melo, G. (2018). Co-PACRR. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18* (pp. 279–287). New York, New York, USA: ACM Press. https://doi.org/10.1145/3159652.3159689

Jaech, A., Kamisetty, H., Ringger, E., & Clarke, C. (2017). Match-Tensor: a Deep Relevance Model for Search. Retrieved from http://arxiv.org/abs/1701.07795

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Retrieved from http://arxiv.org/abs/1408.5882

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (pp. 253–256). IEEE. https://doi.org/10.1109/ISCAS.2010.5537907

Liu, X., Gao, J., He, X., Deng, L., Duh, K., & Wang, Y.-Y. (2015). Representation

Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *The 2015 Annual Conference of the North American Chapter of the ACL* (pp. 912–921). Denver, Colorado: NAACL. Retrieved from https://www.microsoft.com/en-us/research/publication/representation-learning-using-multi-task-deep-neural-networks-for-semantic-classification-and-information-retrieval/

Manning, C. D., Raghavan, P., Schütze, H., & others. (2008). *Introduction to information retrieval* (Vol. 1). Cambridge university press Cambridge.

McDonald, R., Brokos, G.-I., & Androutsopoulos, I. (2018). Deep Relevance Ranking Using Enhanced Document-Query Interactions. Retrieved from http://arxiv.org/abs/1809.01682

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Retrieved from http://papers.nips.cc/paper/5021-distributed-representations-of-words-andphrases

Mitra, B., & Craswell, N. (2017a). An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval*. Retrieved from https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/

Mitra, B., & Craswell, N. (2017b). An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval*, 1–117. https://doi.org/10.1561/XXXXXXXXXX

Mitra, B., Diaz, F., & Craswell, N. (2017). Learning to Match using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web - WWW '17* (pp. 1291–1299). New York, New York, USA: ACM Press. https://doi.org/10.1145/3038912.3052579

Mitra, B., Nalisnick, E., Craswell, N., & Caruana, R. (2016). A Dual Embedding Space Model for Document Ranking. Retrieved from http://arxiv.org/abs/1602.01137

Nalisnick, E., Mitra, B., Craswell, N., & Caruana, R. (2016). Improving Document Ranking with Dual Word Embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion* (pp. 83–84). New York, New York, USA: ACM Press. https://doi.org/10.1145/2872518.2889361

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., … Ward, R. (2016). Deep Sentence Embedding Using Long Short-term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, *24*(4), 694–707. Retrieved from http://dl.acm.org/citation.cfm?id=2992449.2992457

Pang, L., Lan, Y., Guo, J., Xu, J., & Cheng, X. (2016). A Study of MatchPyramid Models on Ad-hoc Retrieval. In *Neu-IR '16 SIGIR Workshop on Neural Information Retrieval*. Pisa, Italy. https://doi.org/10.1145/1235

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., & Cheng, X. (2016). Text matching as image recognition. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press. Retrieved from https://dl.acm.org/citation.cfm?id=3016292

Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017a). DeepRank. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17* (pp. 257–266). New York, New York, USA: ACM Press. https://doi.org/10.1145/3132847.3132914

Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017b). DeepRank. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17* (pp. 257–266). New York, New York, USA: ACM Press.

https://doi.org/10.1145/3132847.3132914

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. Retrieved from http://arxiv.org/abs/1802.05365

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '98* (pp. 275–281). New York, New York, USA: ACM Press. https://doi.org/10.1145/290941.291008

Rasolofo, Y., & Savoy, J. (2003). Term Proximity Scoring for Keyword-Based Retrieval Systems (pp. 207–218). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36618-0_15

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, *60*(5), 503–520. https://doi.org/10.1108/00220410410560582

Robertson, S., & Zaragoza, H. (2010). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, *3*(4), 333–389. https://doi.org/10.1561/1500000019

Schmidhuber, J. (2014). Deep Learning in Neural Networks: An Overview. Retrieved from http://www.idsia.ch

Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management - CIKM '14* (pp. 101–110). New York, New York, USA: ACM Press. https://doi.org/10.1145/2661829.2661935

Tomas Mikolov. (n.d.). Google Code Archive - Long-term storage for Google Code

Project Hosting. Retrieved November 9, 2018, from https://code.google.com/archive/p/word2vec/

Tomas Mikolov∗ , Wen-tau Yih, G. Z. (2013). Linguistic Regularities in Continuous Space Word Representations. *Hlt-Naacl*, (June), 746–751. Retrieved from http://anthology.aclweb.org/N/N13/N13-1.pdf#page=655

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., … Zhang, D. (2017). IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. https://doi.org/10.1145/3077136.3080786

Wang, Y., Wang, L., Li, Y., He, D., Liu, T.-Y., & Chen, W. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. Retrieved from http://arxiv.org/abs/1304.6480

Xiong, C., Dai, Z., Callan, J., Liu, Z., & Power, R. (2017). End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17* (pp. 55–64). New York, New York, USA: ACM Press. https://doi.org/10.1145/3077136.3080809

Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., … Lease, M. (2016). Neural Information Retrieval: A Literature Review. Retrieved from http://arxiv.org/abs/1611.06792

Appendix A – Sample of Topic Set

251: identifying spider bites
252: history of orcas island
253: tooth abscess
254: barrett's esophagus
255: teddy bears
256: patron saint of mental illness
257: holes by louis sachar
258: hip roof
259: carpenter bee
260: the american revolutionary
261: folk remedies sore throat
262: balding cure
263: evidence for evolution
264: tribe formerly living in alabama
265: F5 tornado
266: symptoms of heart attack
267: feliz navidad lyrics
268: benefits of running
269: marshall county schools
270: sun tzu
271: halloween activities for middle school
272: dreams interpretation
273: wilson's disease
274: golf instruction
275: uss cole
276: how has african american music influence history
277: bewitched cast
278: mister rogers
279: game theory
280: view my internet history
281: ketogenic diet
282: nasa interplanetary missions
283: hayrides in pa
284: where to find morel mushrooms
285: magnesium rich foods
286: common schizophrenia drugs
287: carotid cavernous fistula treatment
288: fidel castro
289: benefits of yoga
290: norway spruce
291: sangre de cristo mountains
292: history of the electronic medical record
293: educational advantages of social networking sites
294: flowering plants
295: how to tie a windsor knot
296: recycling lead acid batteries
297: altitude sickness
298: medical care and jehovah's witnesses
299: pink slime in ground beef
300: how to find the mean