TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

Cross-Lingual Semantic Search

Authors:	Liyan Jiang, Phuong Mai, Dmytro Rybalko
Mentor(s):	Dr.-Ing. Andreas Schoknecht, TWT GmbH
Co-Mentor:	Michael Rauchensteiner (Department of Mathematics)
Project lead:	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor:	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Abstract

In Information Retrieval, we are given a query and want to retrieve the according relevant information. Cross-Lingual Semantic Search deals with retrieving information from more than one language. In this work, we focus on retrieving relevant documents for a query that can be either written in English or German. The retrieved documents are also not restricted to one language. Therefore we compare the state-of-the-art methods, that are based on exact matching, with a neural approach. We evaluate our methods on the publicly available Cranfield dataset, which is a collection of abstracts from areodynamic academic papers. Our goal is to improve the results from Balabel (1), using popular information retrieval baselines such as boolean search, TF, IDF, TF-IDF, etc. Additionally, we deal with Natural Language Processing in order to figure out how preprocessing contributes to the information retrieval results. Moreover, we try to solve the task via Topic Modelling, which provides a semantic point of view for tackling the problem.

Contents

List of Figures **III**

List of Tables **IV**

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Dataset Description . . . 1
- 1.3 Current Implementation . . . 2
  - 1.3.1 Models . . . 2
  - 1.3.2 Retrofitting . . . 3
  - 1.3.3 Evaluation . . . 4

2 Preprocessing **5**

3 Natural Language Processing **5**
- 3.1 Information Extraction . . . 6
  - 3.1.1 Word pairs and phrases . . . 6
  - 3.1.2 Word2vec's vocabulary . . . 7
  - 3.1.3 Negative Sampling . . . 7
- 3.2 Interaction-based networks . . . 7

4 Information Retrieval Techniques **8**
- 4.1 LDA Topic Model . . . 9
  - 4.1.1 Introduction to LDA Topic model . . . 9
  - 4.1.2 Intrinsic Evaluation . . . 10
  - 4.1.3 Query-document Relevance Score . . . 10
- 4.2 Hierarchical Dirichlet Process . . . 11
- 4.3 Exact Matching methods . . . 12
  - 4.3.1 Terrier . . . 12
  - 4.3.2 BM25 . . . 13
  - 4.3.3 Markov random field model . . . 14
  - 4.3.4 Other Terrier models . . . 15
- 4.4 Adversarial Discriminative Domain Adaption . . . 15
  - 4.4.1 Domain Adaptation Approach . . . 15
  - 4.4.2 Neural Network Architecture . . . 16
  - 4.4.3 Datasets . . . 16

5 Evaluation Metrics **18**

6 Implementation and Experiments **19**
- 6.1 Trec Eval . . . 20
- 6.2 Natural Language Processing . . . 20
- 6.3 Topic Modelling . . . 22
  - 6.3.1 LDA Topic Modelling . . . 22
  - 6.3.2 HDP Topic Modelling . . . 25

6.3.3 Conclusion 26
6.4 Terrier 27
6.4.1 Data Preparation 27
6.4.2 Results 27
6.4.3 Conclusion 29
6.5 Word2vec Phrase detection 30
6.5.1 Model description 30
6.5.2 Preprocessing 30
6.5.3 Parameters 30
6.5.4 Results 31
6.5.5 Conclusion 32
6.6 Transfer Learning 33
6.6.1 Preprocessing 33
6.6.2 Results 34
6.6.3 Comparison to exact matching methods 35
6.6.4 Conclusion 35
6.7 Comparison of all information retrieval techniques 36

7 Future work **36**

References **37**

List of Figures

1 Skip gram architecture (2) 4
2 Pipeline of preprocessing 5
3 Interaction-based network (3) 8
4 Exact and semantic Matching baselines on different collections (4) 12
5 Overview of Terrier's retrieval process (5) 13
6 Relevance matching model with adversarial discriminator (6) . . 15
7 Duet Model Architecture (7) 17
8 Tuning the Coherence and Perplexity of LDA Topic Model . . . 22
9 Visualization of LDA Model 23
10 Visualization of the HDP Model 26

List of Tables

1 Statistics about Cranfield dataset. 1
2 Evaluation metrics on the Cranfield dataset from (1) 5
3 Statistics of Cranfield dataset with different preprocessing 21
4 Trec eval results for different preprocessing on Cranfield dataset . 21
5 LDA Document clustering results 24
6 LDA Topic model results on Cranfield dataset 25
7 Trec eval results for the HDP Model and LDA Model 25
8 Trec eval results for Terrier on the English dataset 28
9 Trec eval results for Terrier on the German dataset 29
10 Trec eval results with word2vec's phrase detection on the English dataset (with lemmatization, stop words removal and tokenization) 31
11 Trec eval results with word2vec's phrase detection on the German dataset (without lemmatization and stop word removal) 32
12 Trec eval results with word2vec's phrase detection on the German dataset (with lemmatization) 32
13 Transfer learning results on Duet Model 34
14 Adversarial learning results on the stacked dataset 34
15 Results of the best exact matching methods and the duet adversarial network 35
16 Results of the all information retrieval techniques 36
17 Comparison of top-5 predictions for exact matching and neural network model 36

1 Introduction

1.1 Background

The main goal of our project is to build an advanced information retrieval system for the Cranfield collection. This collection contains documents solely in English. Additionally, we translate the corpus to german. A user can formulate question to the system in both language, so one of the requirements to the system is that it is able to handle cross-lingual retrieval. Further challenges include retrieving semantically relevant documents even if the words in query and answer do not match exactly and prioritizing documents about queried topic over those that only mention the topic. The first attempt to build such a system was done in the master thesis of Balabel (1). Cross-lingual Engine for Informed Semantic Search engine in the Technical Domain (CLEISST) was developed, which implements standard retrieval techniques such as the boolean model, different variants of TF-IDF as well as a dual embedding space model (DESM). Based on the results achieved in the thesis we will further develop the system, propose new retrieval approaches and compare results achieved by our models with the results of the thesis.

1.2 Dataset Description

Our research is based on the Cranfield Collection. As we do not have documents written in German, we generate a parallel German version by performing machine translation. For the corpus, we take 80 percent of the data as our training set and ten percent of the data as our validation set. The rest ten percent becomes our test set, which we use to evaluate our model in the end.

The Cranfield Collection is a corpus used for information retrieval experiments. It contains 1400 abstracts of aerodynamics journal articles from the collection of academic papers of the college. The whole corpus consists of three components: the queries, the documents and the relevance score of every document-query match.

At first we explore the corpus by performing simple statistical analyzation of the raw data. There are 1400 queries and documents in total. We try to analyze the co-occurrence of "theme words" in both sets. So as a necessary preprocessing step, we removed the punctuations, the stop words and other signs that don't have any semantical significance. We also do the tokenization and lemmatization on the original raw data. The table below shows the number of terms and the size of vocabulary in both query set and document set respectively.

	Query set	Document set
Vocabulary size	811	8263
Number of terms	2193	120791

Table 1: Statistics about Cranfield dataset.

Furthermore, it is straightforward to think that the words in the query are very likely to occur in its relevant documents, especially when most of the documents are of the same topic. To prove this, we calculate the most frequent 50 words in query and document set and come to the fact that 27 out of 50 words are overlapping.

'flow', 'number', 'pressure', 'result', 'boundary', 'effect', 'method', 'theory', 'layer', 'solution', 'equation', 'body', 'wing', 'surface', 'distribution', 'problem', 'condition', 'heat', 'plate', 'made', 'experimental', 'speed', 'laminar', 'analysis', 'cylinder', 'hypersonic', 'transfer'

For each document, the proportion of the words that also occur in the query is averagely 0.0718, weighted by its relevance scale. Among all documents with relevance score, 596 out of 816 documents have overlapping vocabulary with its corresponding query. These statistics gives us hints that focusing on the semantic relation of the common words in the query and the documents might contribute to the performance of information retrieval techniques.

1.3 Current Implementation

In this section, we present the status of the Master Thesis CLEISST of Balabel (1). The thesis deals with the question "whether retrofitting word vectors with knowledge from lexical resources can improve the performance of a search engine" (1). Therefore, Balabel implemented a few models and applied retrofitted words to some of them. In the following, we describe the models in detail and introduce the technique retrofitting in the later part of the section. Afterwards, we briefly mention the evaluation metrics used for the models.

1.3.1 Models

Boolean Model This model categorizes a document either as relevant, indicated with 1, or irrelevant, indicated as 0. This is chosen regarding the existence of words from the query in the document. Besides its simplicity, it is impossible to determine how relevant the document is based on the static categories given. In addition to that, cross-lingual relevance cannot be represented by this model as it only deals with word occurrence. (1)

TF-IDF Another popular ranking method is the Term Frequency-Inverse document frequency, also known as TF-IDF. For each word in the query, we count the occurences in a single document and the occurrence document wide. These results are multiplied and represent the overall relevance of a word throughout the documents. This model approaches the difficulties of the Boolean Model and can be extended with weights on the documents (1).

Continuous models: DESM More advanced techniques can be found in the Neural information retrieval (NIR) techniques that can be also called continuous models. Contrary to the presented techniques, continuous models focus on semantic similarity rather than simple count methods. Therefore, both input query and documents are represented by word embeddings that we will face in the later part of this section. Balabel (1) implemented a simplified Dual Embedding Space Model (DESM) proposed by (8). In this model, we represent both query and document as normalized unit vectors whose dot product indicates the document relevance for a query. In the original model, we have two matrices: one for context words and one for center words, which we also call OUT and IN. In CLEISST, the model considers solely the IN matrix for simplification (1).

Combined Models In practise, continuous models are often combined with traditional models. An example that was proposed by (8) is the combination of DESM with TF-IDF. Moreover, (1) presented the DESM to work the best in this combination. The combination score is the following

$$s(q, d) = s1(q, d) + (1-)s2(q, d)$$

with $0 < \alpha < 1$ and s_1 and s_2 being the scores of the TF-IDF model and the continuous model.

1.3.2 Retrofitting

In this section, we want to gain more semantic information in a raw query with the help of lexical relation. In retrofitting, we have a word vector and a lexicon as input and get a refined vector as output (9). Balabel (1) stated that the technique is especially beneficial in the cross-lingual retrieval after testing it on the TF-IDF and the continuous model. Therefore, monolingual and cross-lingual word embeddings were trained. In the following, we introduce the first one in more detail. Furthermore, the ranking is improved when using the combined model. On the other hand, we take loss in string-based retrieval. As retrofitting focuses on semantic similarity only, we can conclude that the technique overall improve the output of the model.

Skipgram Technique A well-known technique for generating monolingual word embeddings is the skip-gram technique of word2vec (2). Thus, we consider a word w and try to predict probabilities for words surround w. Therefore, we distinguish between context words, that should return high probability, and the contrary negative words, whose probability we want to minimize. In the following, we represent w as a one hot vector and use it as an input for the skip-gram neural network. This network predicts the probability of every word in the vocabulary to occur near to w. The input layer on the left represents the target word as a one hot vector, i.e. one entry in the vector is 1, the others are set to 0, thus it is V-dimensional. It is passed through an N-dimensional hidden layer and produces scores for specific words. These scores are transformed to probabilities in the output layer, respectively called the softmax layer (2). In case we have several contexts C, this layer has dimension CV as we want to predict more than one word near our input word (10). Figure 1 depicts the architecture of the skip-gram technique.

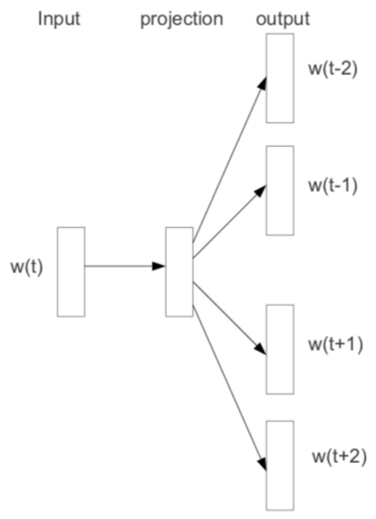


Figure 1: Skip gram architecture (2)

Lexical Semantic Resources In addition to the query, another input for the retrofitting technique are lexical semantic resources. Balabel (1) included Beolingus, FrameNet and Paraphrase Database.

Retrofitting technique For retrofitting, the embedding first needs to be initialized considering different languages. In addition to that, we have and English and German vocabulary, represented in word vector set, as well as an ontology. For the initialization, we differentiate between three cases described in (1):

Case 1. Given a word that is in the ontology solely: Vector initialization is random.
Case 2. Given a word that is found in both English and German vocabulary: Vector initialization based on the average of English and German vectors.
Case 3. Given a word that is found in one vocabulary solely: Vector initialization based on language mapping and average of the languages. Optionally, it follows the optimization after the initialization step. The thesis introduced an update method that relies on the number of adjacent nodes and the weights of the edges in the ontology.

1.3.3 Evaluation

Balabel's evaluation (1) considers the metrics precision, recall, F1 for traditional methods. The Mean Average Precision (MAP) and R-precision served as an indicator for the ranking of the documents. Additionally, a modified Mean

Reciprocal Rank (MRR) evaluated placement of documents in a list. In section 5, we present the metrics in more detail. For now, it is sufficient to know that the higher the metric value, the better the ranking was. The metrics are ranging in the interval $[0, 1]$. On Table 2, Balabel's results for the evaluation on the Cranfield dataset are listed. MAP has a quite low value of 0.131 where we see high potential in improvement. MRR has a value of 0.396 that we also increase in this project.

MAP	MRR
0.131	0.396

Table 2: Evaluation metrics on the Cranfield dataset from (1)

2 Preprocessing

We perform text cleaning in every document pieces. After that we do the translation and remove stop words in English and German document sets respectively. The whole process pipeline is demonstrated as a flowchart in Figure 2.

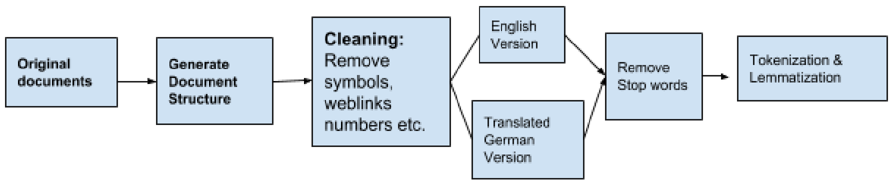

Original documents
Generate Document Structure
Cleaning: Remove symbols, weblinks numbers etc.
English Version
Translated German Version
Remove Stop words
Tokenization & Lemmatization

Figure 2: Pipeline of preprocessing

We want every token in our input text to be terms that of semantic meanings. Observing the original text in Technical Documents corpora, there are many symbols and strings which do not contain any semantic meaning, such as weblinks, number of orders, some typesetting symbols and serial numbers. Our solution is to use regular expression to detect and remove these strings.

After that, we can build the vocabulary of the corpus. However, some words, such as "the", "a", "and" and so on, have high word frequency throughout the whole corpus. These stop words appears in almost every document. As we want to analyses the topic of the document based on word frequency in the following steps, these stop words might hinder the accuracy of the topic detection. So we remove them based on a online stop words list provided by gensim(11).

3 Natural Language Processing

Currently, we use raw queries for the input of the retrofitting technique. Often, some words do not contain necessary information for information retrieval, e.g. stop words such as *a* or *the*. These words decrease the performance of retrieval techniques. In order to find the right chapters relating to a query, the whole documents are scanned. This is computationally expensive. This section suggests methods to extract information from queries by disregarding irrelevant words. Furthermore, it introduces interaction-based networks that focus only on relevant parts of a document instead of the whole document.

3.1 Information Extraction

This section suggests improvements regarding word embeddings and information extraction on queries. In CLEISST (1), the skip-gram technique is used to generate word embeddings. In the skip-gram model, word embeddings are built by predicting the nearby words of a center word (2). Therefore, scores are passed through the neural net and finally use a softmax function. The drawback of this method relies on the tremendous number of weights in the neural net that have to be updated with every new training sample. As the softmax layer contains all words in the vocabulary, training the net causes high computation that is not always necessary. Some probabilities, such as for stop words, do not urgently have to be updated as they play no significant role in the query. Mikolov et al. suggested improvements of the word2vec's skip-gram model in their second paper (2). In the following, some of the approaches are introduced. You find the implementation of the approaches as a fork in (12) that is written in C.

3.1.1 Word pairs and phrases

In natural language, we often come across words that belong together, e.g. "Scrum Master". This bigram has a different context and meaning than the unigrams "Scrum" and "Master". Therefore, treating "Scrum Master" as a single word makes more sense for our use case than tokenizing them. Mikolov et al. addressed this issue and implemented a tool that follows a simple data-driven approach (2): A phrase with words w_1 and w_2 is assigned a score based on the frequency of the unigrams and the bigram

$$score(w_1, w_2) = \frac{count(w_1, w_2) - d}{count(w_1) \cdot count(w_2)}$$

The coefficient d reduces the score for phrases containing too infrequent words. Additionally, Mikolov et al. defined a threshold that a score has to reach in order to add the phrase $w_1 w_2$ to the vocabulary (2). If the occurrence of the phrase $w_1 w_2$ is approximately as high as the occurrences of the single words w_1 and w_2, then we have a high score. Frequent stop words words might occur more often as single words rather than in phrases and therefore would increase the denominator. This results in a low score. After the first iteration, the phrase consisting of two words is treated just as a single word. In order to extend phrases, the authors of the paper iterated the dataset in their tool typically up to two until four times. This can extend our bigram to a trigram like "Scrum Master Seminar".

Especially in the aerospace domain, we find lots of phrases, such as ”heat transfer” or ”stagnation point”. These have a completely different meaning when they appear in the single context. Keeping them as a phrase, i.e. a single token, we believe in better information retrieval results.

3.1.2 Word2vec's vocabulary

In addition to the word phrase detection, Mikolov et al. introduced subsampling of frequent words, a technique to disregard frequent words that do not infer high information (2). The authors trained the techniques on over 100 billion words of a Google News Dataset resulting on a dataset of up to three million words. Each word vector contains 300 features. You can find the vocabulary at (13). It is split into 30 text files each listing 100,000 words in single lines. These words are sorted in the original order. In the description, we can assume that stop words such as “a” are disregarded, while some like “the” can be found in the vocabulary. You can find phrases consisting of two or more single words as well.

3.1.3 Negative Sampling

In section 1.3, the architecture of the skip-gram model was presented. With each new training sample arrived, we have to go through a high number of weights in the output layer. One drawback of this technique is the consideration of the full vocabulary with each input vector crossing through. Skip gram requires an update of every weight in the softmax layer. This can be costly in performance. In order to approach these difficulties, negative sampling was introduced in (2). The word negative refers to words that occur frequently, e.g. stop words. Often, these words are not rich in information and can be disregarded. In comparison to that, positive words can conclude meaning that we can use for information retrieval. We then define k negative samples to update the weights for our neural net. Mikolov et al. suggested k to be between five and 20 for small training datasets, while it can be significantly smaller for big datasets (2).

Negative Sampling plays an important role referring performance enhancement. Instead of updating the weights for all possible words in the vocabulary, we reduce this step by still updating all positive, but only k negative words. As we do not extract information from the whole set of frequent words, it can be valuable improvement for retrofitting. Nevertheless, we do not have a high number of documents and a big vocabulary so that negative sampling could improve our model in a valuable manner. Therefore, negative sampling is low prioritised for the project.

3.2 Interaction-based networks

Until now, we have iterated through the whole documents in order to find similarity to our input query. Balabel (1) suggests to use retrofitting in interaction-based networks that iterates only on parts of both query and documents. The interaction-based network was introduced from Lu and Li (14). The idea is that a convolutional neural network (CNN) is used for sliding over query and document. We can make use of Topic Modelling here, analyzing the most probable topic similarity of query and document parts. Finally, the results are aggregated and the relevance is calculated (1). Figure 3 shows this process.

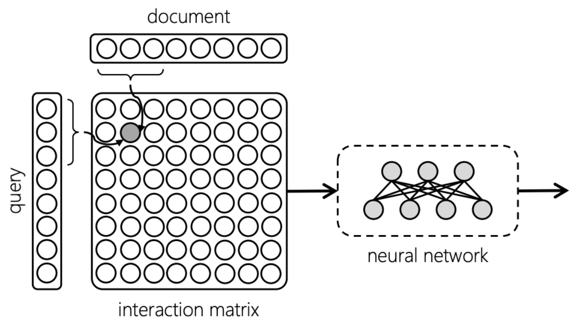


Figure 3: Interaction-based network (3)

Retrofitting in interaction-based network can be beneficial regarding performance. The training needs large datasets for this approach in order to train the CNN. Unfortunately, the project does not have enough queries and documents to train this model. Therefore, we do not see a valuable gain with this approach. The model then iterates through the document from high level, such as the title, to low level, i.e. single paragraphs. We see this as sufficient in order to accelerate the document search.

4 Information Retrieval Techniques

This section introduces the information retrieval techniques theoretically. Therefore, this section is divided into three parts: In section 4.1 and 4.2, we discuss information retrieval with topic modelling. This way, documents can be represented by topics and retrieved accordingly. First, we introduce into the well known LDA Topic Model that extracts the topics of the dataset with a predefined number of topics. Afterwards, we present the HDP model, that is an unsupervised model learning the number of topics from text. In the second part of this section, we discuss Exact Matching Methods. Section 4.3 introduces Terrier, a platform that provides state-of-the-art models. A well known method is BM25 that we discuss more in detail. Finally, we elaborate a neural network approach in section 4.4 in order to compare the exact matching methods to neural approaches.

4.1 LDA Topic Model

Most technical documents are related to specific topics in a technical domain. Therefore we introduce topic modeling technique to facilitate our information retrieval process. A standard approach is to use LDA (i.e. Latent Dirichlet Allocation) (15) to implement the topic model. In this section, we will briefly introduce the mechanism of the LDA topic model and how it can contribute to our information retrieval techniques.

4.1.1 Introduction to LDA Topic model

Topic modelling is based on the assumption that documents are a mixture of latent topics, where topics can be represented as a distribution of words in that document. As the training goes, theme words and topics iteratively defines each other, like the formula shows below.

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j)$$

In this formula, T indicates the number of topics, w_i is the i-th word in a particular document. $P(z_i = j)$ is the probability that j-th topic was sampled for the i-th word token while $P(w_i|z_i = j)$ is the probability of the i-th word under the topic j. These two probability terms can be modeled as multinomial distributions. LDA is the conjugate prior for the multinomial distributions, which means that it provides a prior estimate of the parameters in our likelihood observation among all documents. Our observation on the corpus consist of two parts: topic-word distribution, indicating the probability of a word being assigned to a specific topic; topic-document distribution, indicating the probability of a document being assigned to a specific topic. These two observations are performed in a alternative optimization fashion and update the hyper-parameters of the multinomial distribution and the prior.

The LDA topic model is a generative model. It is based on the bag of words assumption, in which the position of each word is insignificant to the result. First we randomly assign k topics to all M documents. Then for every document, we calculate the probability of each word being assigned to this topic, and decide the overall probability of this document being assigned to every particular topic. For each topic, we also calculate the probability of each word that belongs to this topic in parallel.

As stated before in the preprocessing part2, topic modeling will categorize our documents and make the query more accurate and efficient. It is also meaningful to train topic model for the corpus we use. As the topic model evolves, the words under a specific topic will be more and more particular instead of general topic words. Some trivial words in the query, which may indicate some subtopics that are hard to recognize by similarity matching, will be catched and directed by our topic model and better target the probable document sets which may contain the answer.

4.1.2 Intrinsic Evaluation

Perplexity and coherence are two important metrics used to evaluate LDA topic model as well as other text classification tasks. In the information theory, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. In our case, as we assume that documents are a mixture of different topic distributions, we want the perplexity in each topic be as low as possible.

$$Perplexity = 2^{H(p)} = 2^{-\sum_x p(x)\log_2 p(x)}$$

Obviously in the formula, high entropy entails high perplexity. As the goal of LDA is to categorize documents given a number of topics, we want the documents in each topic to be as similar to each other as possible. So that the entropy in documents assigned to the same topic should be low. And the perplexity is therefore minimized.

Another important metric is coherence. A good model will generate coherent topics, which means that the tokens in this topic are strongly correlated to each other. Here we use intrinsic measure of coherence UMass to evaluate our LDA Model.

$$score_{UMASS}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)}$$

Here w_i and w_j are arbitrary tokens under the same topic. $D(w_j, w_j)$ means the number of documents under this topic containing both w_j and w_j. Likewise $D(w_i)$ is the number of documents under this topic containing token w_i, which is a normalizing term. A Laplace smoothing is used to smooth those rare token pairs and ensuring numerical stability.

Obviously, high coherence score indicates good topic model, as the tokens within the same topic should be coherent.

4.1.3 Query-document Relevance Score

With a trained LDA model at hand, we can get topic frequency matrix for both query and document. To find the relevant document for a specific query, the most straightforward approach is simply returning the documents that are assigned to the same topic as the query by LDA model. However, the drawbacks are obvious. Intuitively, rigid grouping and matching is too general for each specific query-document pair, especially when the number of topics we are modeling the LDA is low. To prove this, we design an experiment and draw some conclusions from the results to show that is approach does not work. The details and analysis of the experiment are in section 6.3.1.

As LDA model generates document-topic frequency matrix as a result, we can use this result to map either documents or queries into a n-dimensional feature space with n corresponding to the number of topics in our model. Then we choose distance measure to calculate the distance between each query-document pair, based on which the ranking of relevant documents for a query is derived.

One possibility with regard to the distance measure is Jensen Shannon Divergence. The Jensen Shannon Divergence is a way of measuring the dissimilarity between two probability distributions.

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M)$$

where D is a Kullback-Leibler Divergence and

$$M = \frac{1}{2}(P + Q)$$

This distance measure utilizes all information provided by LDA model and yields high information entropy, which is beneficial to our classification task. For each query, we iterate through all documents and feed in the topic frequency $q \in R^T$ for query as well as for documents $d \in R^T$ into Jensen Shannon Divergence measure and therefore derive a dissimilarity score, where T is the number of topics in the LDA topic model. In order to make compatible with the evaluation program trec eval (16), which uses the similarity as the measure, we take the inverse of Jensen Shannon Divergence as the similarity score in the final results.

Likewise, other distance measures like Euclidean distance might have comparable performance as Jensen Shannon Divergence, as they take all topic distribution of a document into account.

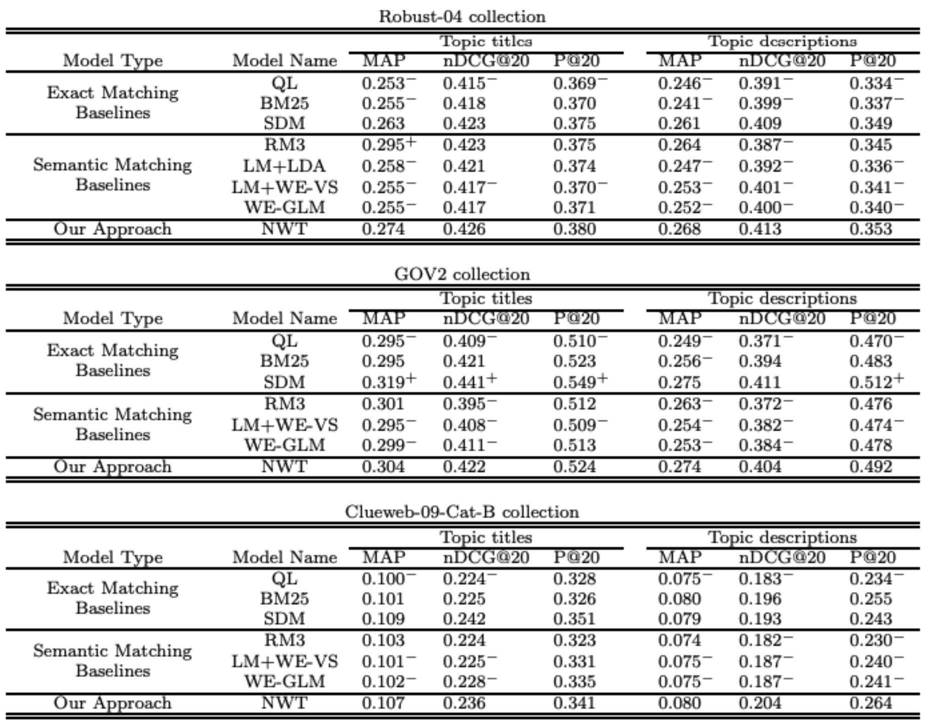
4.2 Hierarchical Dirichlet Process

So far, we introduced the LDA Topic Model in order to specify topics from documents. A main drawback of this method is the pre-definition of the number of topics. In lots of use cases, this information is not available before training the model. Therefore, we consider the **Hierarchical Dirichlet Process** (HDP) topic model, an unsupervised model that learns the number of topics from the text. This implies the possibility of an infinite number of components. In this section, we describe the HDP topic model from Wang et al. (17). The HDP model is mostly used in probabilistic topic modelling, allowing mixed-membership. With the help of posterior inference, the HDP model extracts the number of topics. It defines a Dirichlet process for each group and a global Dirichlet process as a base distribution. The HDP model has the disadvantage that, learning with posterior inference, the dataset has to be passed multiple times which slows down the performance. Therefore, Wang et al. (17) proposed an online version of the HDP model that is applicable for massive data and that we are going to consider in the following. The authors optimized the posterior inference with stochastic approximation, called *online variational inference*. This way, the HDP model is scalable referring to posterior inference.

In order to evaluate the quality of the information retrieval, we can transform the HDP Model to an LDA model with the extracted number of topics. This way, we use, again, the Jensen Shannon Divergence in order to measure the similarity as described in section 4.1.3.

4.3 Exact Matching methods

In order to find a suitable information retrieval technique, recent papers on information retrieval used state-of-the-art baselines. Figure 4 presents the results on GOV2 collection for different information retrieval techniques evaluated in (4). GOV2 is a large collection of documents (25M) used in many Text Retrieval Conferences (TRECs). The evaluation metrics used in the example will be discussed in section 5. For now, it is important to know, that higher values mean better performance and each value must be in a range between 0 and 1. The results show that the Exact Matching Baselines perform comparable to the Semantic Matching Baselines. Referring the GOV2 collection, the state-of-the-art methods even show better results than semantic methods for all metrics. Two of the strongest exact matching baselines are the BM25 and SDM models that are described in detail in section 4.3.2 and 4.3.3.



Robust-04 collection

		Topic titles			Topic descriptions		
Model Type	Model Name	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
Exact Matching Baselines	QL	0.253^-	0.415^-	0.369^-	0.246^-	0.391^-	0.334^-
	BM25	0.255^-	0.418	0.370	0.241^-	0.399^-	0.337^-
	SDM	0.263	0.423	0.375	0.261	0.409	0.349
Semantic Matching Baselines	RM3	0.295^+	0.423	0.375	0.264	0.387^-	0.345
	LM+LDA	0.258^-	0.421	0.374	0.247^-	0.392^-	0.336^-
	LM+WE-VS	0.255^-	0.417^-	0.370^-	0.253^-	0.401^-	0.341^-
	WE-GLM	0.255^-	0.417	0.371	0.252^-	0.400^-	0.340^-
Our Approach	NWT	0.274	0.426	0.380	0.268	0.413	0.353

GOV2 collection

		Topic titles			Topic descriptions		
Model Type	Model Name	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
Exact Matching Baselines	QL	0.295^-	0.409^-	0.510^-	0.249^-	0.371^-	0.470^-
	BM25	0.295	0.421	0.523	0.256^-	0.394	0.483
	SDM	0.319^+	0.441^+	0.549^+	0.275	0.411	0.512^+
Semantic Matching Baselines	RM3	0.301	0.395^-	0.512	0.263^-	0.372^-	0.476
	LM+WE-VS	0.295^-	0.408^-	0.509^-	0.254^-	0.382^-	0.474^-
	WE-GLM	0.299^-	0.411^-	0.513	0.253^-	0.384^-	0.478
Our Approach	NWT	0.304	0.422	0.524	0.274	0.404	0.492

Clueweb-09-Cat-B collection

		Topic titles			Topic descriptions		
Model Type	Model Name	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
Exact Matching Baselines	QL	0.100^-	0.224^-	0.328	0.075^-	0.183^-	0.234^-
	BM25	0.101	0.225	0.326	0.080	0.196	0.255
	SDM	0.109	0.242	0.351	0.079	0.193	0.243
Semantic Matching Baselines	RM3	0.103	0.224	0.323	0.074	0.182^-	0.230^-
	LM+WE-VS	0.101^-	0.225^-	0.331	0.075^-	0.187^-	0.240^-
	WE-GLM	0.102^-	0.228^-	0.335	0.075^-	0.187^-	0.241^-
Our Approach	NWT	0.107	0.236	0.341	0.080	0.204	0.264

Figure 4: Exact and semantic Matching baselines on different collections (4)

4.3.1 Terrier

Terrier is a platform that offers implementations of state-of-the-art IR models. Ounis et al. (5) introduce the retrieval process that is depicted on Figure 5. In the step *Query process*, Terrier takes care of standard Natural Language

Processing (NLP) such as stop-word removal, stemming and tokenization. In this project, we mainly take advantage of the built in NLP steps from Terrier as they show better results than the own approaches. After the query was processed, the *Matching* step selects the retrieval approach. In addition to BM25, we processed 15 other models listed in section 4.3.4. Finally, Terrier outputs the retrieval in a result file in the appropriate format. For evaluation, we made use of Trec Eval (16), an evaluation tool for ad-hoc retrieval. Given the retrieval results and the gold standard, Trec Eval computes measurements such as MAP, MRR, NDCG and precision. These metrics are the main metrics we focus on for the evaluation. In section 5 we introduce the evaluation metrics in detail.

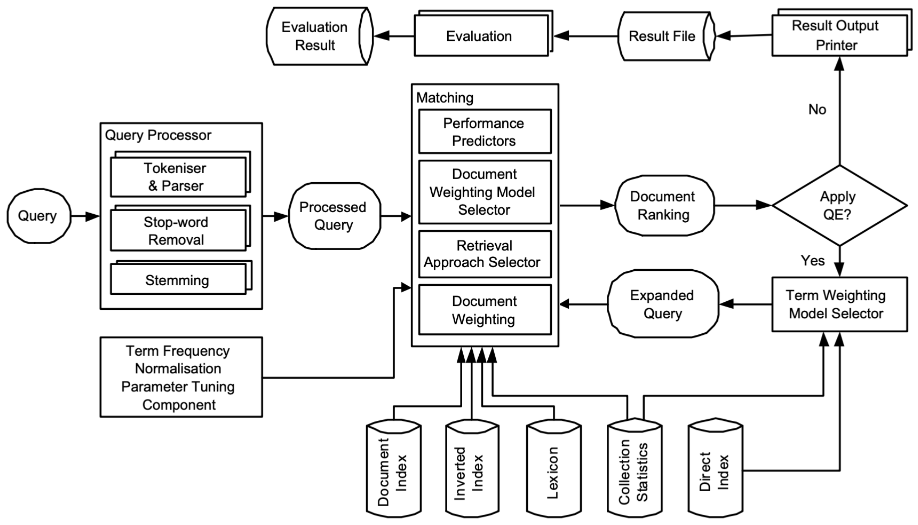


Figure 5: Overview of Terrier's retrieval process (5)

4.3.2 BM25

The BM25 (Best Match) weighting scheme, which is also called Okapi weighting (Okapi BM25) is a probabilistic retrieval model sensitive to term frequency and document length, which can be seen as an instance of TF-IDF retrieval model. BM25 is a not a single function, but rather a family of scoring functions, which have different parameters and components. One of the most popular among these functions is the following:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i,D) \cdot (k_1+1)}{TF(q_i,D) + k_1(1-b+b\frac{|D|}{avgdl})}$$

where D and Q are the sets of document and query words respectively. $TF(q_i, D)$ is the term frequency in document D. $|D|$ is the number of words in document D and $avgdl$ is the average number of words in a document. $IDF(q_i)$ is the inverse document frequency weight of the query term which is computed as follows

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents that contain q_i. BM25 has two free parameters k_1 and b, which have to be chosen in advance. Usual values for k_1 are contained in interval $[1.2, 2.0]$, and b is set to 0.75. The main advantages of the BM25 retrieval model are its efficiency, the ability to perform well even on small datasets and we do not need to train the model.

Okapi BM25 is the most promising approach for technical documents collection, as it is too small for advanced approaches based on neural networks, which can perform better than BM25. There exists also an extension called BM25F, which takes a document structure and anchor text into account, which may further improve the results. This extension requires additional text preprocessing to identify the structure of the document. One possible approach was discussed in previous section on natural language processing.

4.3.3 Markov random field model

Another strong exact matching baseline is SDM model, which is state-of-the-art language model addressing term dependence using Markov random fields (MRF). MRF is used to model a joint distribution $P_\Lambda(Q, D)$ of query words set Q and document D, parametrized by Λ. In general, MRF is constructed from the graph G, in which nodes represent random variables, and the edges define the independence semantics between random variables. In our case nodes can represent documents or query words. There are three variants of the MRF model with different dependence assumptions on words in a query. The full independence model assumes that there is no dependence between query terms given some document D. The sequential dependence model assumes that only neighboring query terms are dependent, while in full dependence variant all query terms in some way are dependent of each other. One of the main problems of this model in our setting is that it requires training to learn the parameters, which is only possible if we have enough relevance-annotated query-document pairs.

Exact matching approaches based on bag-of-words assumption suffer from the query document mismatch drawback. Such mismatch occurs, when the searcher and author use different terms (representations) to describe the same concept.

4.3.4 Other Terrier models

In addition to BM25, Terrier offers a wide range of other exact matching methods. For this project, we included the following models in the evaluation: BB2, DFR BM25, DLH, DLH13, DPH, DFRee, Hiemstra LM, IFB2, In expB2, In expC2, InL2, TF-IDF, Lemur TF-IDF, LGD and PL2. The models are all publicly available on Terrier. You can find their definitions on the Terrier Website (18).

4.4 Adversarial Discriminative Domain Adaption

After evaluating classical approaches to information retrieval such as exact matching and topic modelling using LDA we decided to experiment with advanced deep learning approaches to "learning to rank" problem. It is known, that in order to successfully apply neural networks, one has to collect large amounts of data. But our Cranfield dataset is very small and thus not suitable for training.

In order to deal with this limitation we decided to use transfer learning approach. The model is trained on one dataset and applied to another one. Such approach can work well only in the case when training domain and test domain are the same or at least very similar. Unfortunately, we were not able to find any open-source dataset, that discussed the same topics as Cranfield.

Neural networks for information retrieval expect raw text of query and document as input and output relevance score of such pair, i.e. how well query and document content match. Architecture of such neural networks is designed to extract some hidden features from query and document, that are used for relevance prediction. These features are bound to specific word distribution of the dataset used for training. Authors in (6) proposed to adjust the architecture of NN for relevance matching, such that the features learned by the model should strive to be as dataset invariant as possible. We implemented and tested this approach on Cranfield dataset.

4.4.1 Domain Adaptation Approach

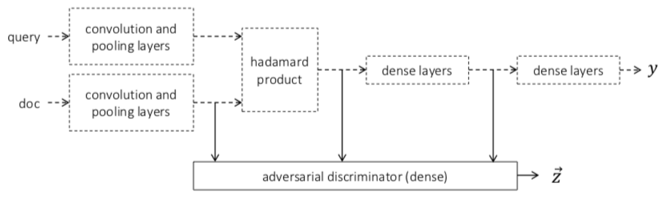

query
doc
convolution and pooling layers
convolution and pooling layers
hadamard product
dense layers
dense layers
y
adversarial discriminator (dense)
$\vec{z}$

Figure 6: Relevance matching model with adversarial discriminator (6)

The main motivation of applying cross domain regularization approach described in (6) is to force a neural network to learn content features for relevance prediction, that are domain invariant, i.e. do not depend on dataset. This can be achieved by introducing adversarial agent: discriminator neural network, constructed of dense layers with softmax output layer, that predicts the domain, from which query-document pair comes from. As input adversarial agent takes outputs of intermediate layers of the ranking model (neural network, which predicts relevance for query-document pair). This approach can be represented as a joint loss function:

$$L = L_{rel}(q, doc_r, doc_{nr}, \theta_D, \theta_{rel}) + \lambda(L_{adv}(q, doc_r, \theta_D) + L_{adv}(q, doc_{nr}, \theta_D)) \quad (1)$$

where L_{rel} is the loss function for relevance output of neural network and L_{adv} is the loss function for output of adversarial discriminator. q is the query, doc_r and doc_{nr} are relevant and irrelevant documents, respectively. θ_{rel} represents the parameters of neural network used for relevance prediction, while θ_D stands for parameters of adversarial agent. λ is used to determine the strength of influence of adversarial discriminator on relevance model. Both L_{rel} and L_{adv} are standard cross-entropy loss functions. The learning of dataset invariant features happens due to the shift of model parameters in the opposite direction to domain specific spaces on manifold. This effect is achieved by utilizing gradient reversal layer, which transforms the standard gradient, $\frac{\delta L_{adv}}{\delta\theta}$ to its additive inverse $-\frac{\delta L_{adv}}{\delta\theta}$. The visualization of cross domain regularization is presented in Figure 6.

4.4.2 Neural Network Architecture

There are a lot of various neural network architectures for relevance prediction of query-document pair. For our approach we selected Duet distributed model (7). It is composed of two separate neural networks: one for matching query and document using local representation, another using distributed representation. These two branches of single neural network are trained jointly. The architecture of Duet model is shown in Figure 7.

4.4.3 Datasets

For this approach, we consider two datasets mentioned in (19) in order to train the neural network: WebAP, insuranceQA and Yahoo L4. In the following we will present the datasets in detail.

WebAP WebAP dataset (20) is derived from the 2004 TREC Terabyte Track Gov2 collection and contains manual annotations of answer paragraphs of candidate documents for each query. It is constructed of 8027 answer passages to 82 TREC queries. The average number of passages per query is 97. Each passage can be annotated with various degrees of relevance: ”perfect”, ”excellent”, ”good”, ”fair”, ”irrelevant”. The average length of the passage is 45 words.

InsuranceQA In 2015, Feng et al. (21) released an insurance question-answering

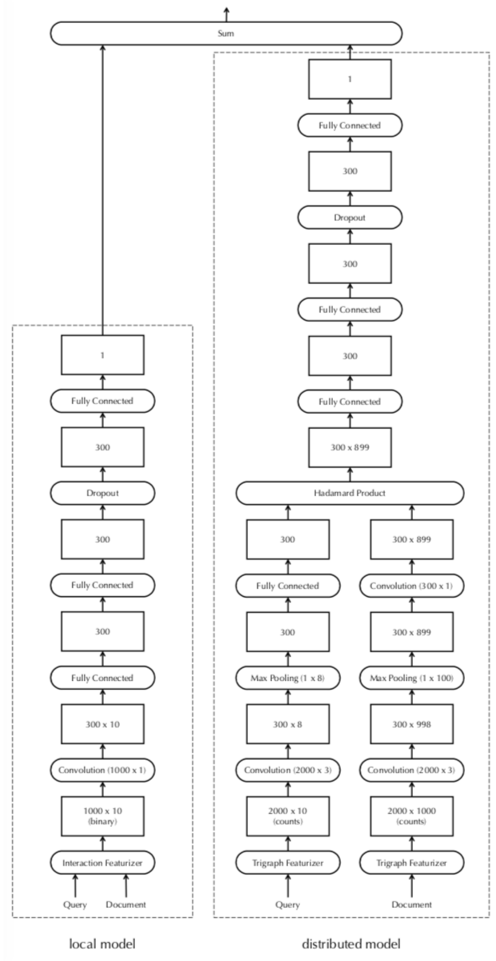


Figure 7: Duet Model Architecture (7)

dataset that is, to their knowledge, the first question-answering dataset published for that domain. The data was collected from an Insurance Library website and contains questions, answers and an answer pool for each question. Note that the questions were stated from real users with domain knowledge. On Github (22), you can find in total, 16,889 questions and 27,413 answers. There are two versions available that differentiate in the poll: in version one, the poll was generated randomly. The poll in version two was generated by Apache SOLR. Apache SOLR is an open source search platform that offers full-text search being well known for scalability (23). Therefore, the answers in version two are semantically related to the question. This does not mean that the ground truth is also included in the poll. As we deal with semantically related answers, we use version two of the insuranceQA dataset.

Yahoo L4 Yahoo published 33 language datasets, out of which the L4 dataset contains Answer Manner Questions (24). The data is a dump from the Yahoo! Answers platform (25) of 10/25/2007. In total, we have 142,627 questions and their answers. Every question is marked with a best answer and several other answers that are ranked on the relevance. In average, a question has 5.74 answers. This dataset is the biggest among the three mentioned and also the richest. It was cleaned so that every questions contains at least four words with at least one noun and one verb. In addition to question and answer, the dataset also holds information about the question category and subcategory as well as a context. The dataset is only available for research purposes and had to be requested from (24).

5 Evaluation Metrics

In this section we are going to introduce the evaluation metrics that we used to evaluate the information retrieval methods in section 6. Therefore, we recap some metrics from Balabel (1) and introduce a new metric.

Precision
This metric considers the relevant documents and all retrieved documents independent from their position. Therefore, it considers the intersection between the relevant and the retrieved documents and divides it to all retrieved documents:

$$precision = \frac{|\{relevant\_documents\} \cap \{retrieved\_documents\}|}{\{retrieved\_documents\}}$$

In the later sections, we consider Precision for the first five and ten retrieved documents, denoted as P@5 and P@10 accordingly.

Mean Average Precision
In order to consider the ranked list, we introduce the parameter k that represents the number of top results of the retrieved document list. Let $precision_i$ be the precision that only takes the first i retrieved documents into account and $rel(d_i)$ is the binary relevance of document d_i.

$$AP = \frac{1}{|relevant\_documents|} \sum_{i=1}^{k} precision_i \cdot rel(d_i)$$

The average over all queries is equivalent to the average of all AP scores, respectively called the **Mean Average Precision (MAP)**:

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q)$$

Mean Reciprocal Rank
The Reciprocal Rank (RR) considers the rank of the most relevant document.

Given a query, it divides 1 by the rank of the most important document. If that document takes the highest rank, it results in an RR of 1 and if it takes the second highest rank, the RR is $\frac{1}{2}$ etc. That means, the higher the RR, the better is the ranking. If we take the average over all queries, we get the **Mean Reciprocal Rank (MRR)**:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Dicsounted Cumulative Gain
The DCG considers the relevance of a document as well as its position. Therefore, the DCG can only be applied for a ranked list. The idea is that highly relevant documents that are low ranked, are penalized:

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

In our case, we have binary relevance for each document and query. Therefore, the fraction results in 0 when the document was not relevant for the query. In the opposite case, the fraction is smaller equal 1, that is the value for the highest relevant document occurring in the first position. The higher the DCG value, the better the ranking is. In the following, we consider the **normalized DCG**, that computes an DCG in the range between 0 and 1. Let IDCG be the ideal DCG at position p, the nDCG is computed as follows:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

The evaluation of the methods in section 6 is going to consider the nCDG for five and ten retrieved documents, denoted as NDCG@5 and NDCG@10 accordingly.

6 Implementation and Experiments

In this section we present the implementation and results of the information retrieval techniques, state-of-the-art baselines as well as a neural network approach. Therefore, we first introduce the evaluation Tool Trec Eval in section 6.1 that compares the gold standard with the retrieval results based on a handful of metrics. Further, we briefly discuss Natural Language Processing methods that include our own approaches and the built in implementation of Terrier. In the following sections, we demonstrate the information retrieval technique, starting with Topic modelling in section 6.3. In section 6.4, the exact matching models are presented, followed by the phrase detection from Word2vec in section 6.5. Finally, we show the results of the neural network approach in section 6.6 and compare them to the state-of-the-art methods.

6.1 Trec Eval

For all the following information retrieval techniques, we use the evaluation tool of the Text Retrieval Conference (TREC), called **Trec Eval**. Given a the RelDocs and DocRank file, Trec Eval computes the metrics Precision, Recall, MRR, MAP, nDCG and many more. RelDocs is a file that contains the following columns: query_id, iter, doc_id, rank. The column, called *iter* is always set to zero and has no particular meaning.
The DocRank file is generated by the retrieval method with the following columns: query_id, iter, doc_id, rank, sim, run_id. The column *iter* is again a constant, this time with the value *Q0* and is disregarded by Trec Eval. *Sim* denotes the similarity of query and document for the predefined model and *Run_id* the model's name. DocRank is compared to the RelDocs file in order to compute the quality of the information retrieval.

Note that we worked with binary ranks solely for the normalized DCG metric. Therefore, we map the relevance value -1 to 0, the values 1 and 2 to 1 and all other relevance values above 2 to 0.

6.2 Natural Language Processing

As shown in Figure 5, in the original terrier pipeline there is a built-in preprocessing module handling all kinds of natural language processing stuff. We want to tweak this process and see how each part of natural language processing contributes to the final results. So we implement our own preprocessing pipeline with nltk(26) and experiment on it.

In Figure 2, the preprocessing pipeline of Cranfield dataset consists of three parts - tokenization, stop words removal and lemmatization. In this section, we will see how different preprocessing pipeline influence the training results. By enabling and disabling each part of preprocessing, we use BM25 from terrier as a fixed information retrieval technique to see how the outcome changes. In our project we use open source library nltk(26) to handle all natural language processing tasks: tokenization, stopwords removal and lemmatization.

Tokenization The most fundamental part of preprocessing is tokenization. Tokenization is the task of splitting text into small pieces of word units, which we call it tokens. Without doing so, any natural language processing tasks cannot be proceed.

Stopwords removal Some words are overwhelmingly common in the text, such as "the", "a", "and" and so on in English. These stopwords are only used for grammatical purpose. In other words, they contributes little semantic significance to the topic of the text. Our research is mostly based on the term frequency. Therefore it is meaningful to remove those "stopwords" in the text, so that their extremely high frequency won't dominate others and hinder the information retrieval.

Lemmatization Lemmatization is the morphological analysis of words. Sometimes we don't care about the exact word forms, but the lemma of the word, in which the word entered in the dictionary. For example, the lemma "go" has a variety of word forms like "went", "goes", "gone". However, in most information retrieval tasks, what we care about is the lemma instead of word forms. So we do lemmatization to map these word forms back to their lemma, so that every different word forms won't be treated as different words.

In Table 3, you can see how the number of tokens and vocabulary size differ when different preprocessing is applied.

	Number of Tokens	Vocabulary
tokenization	145337	6452
tokenization and stopwords removal	133043	5913
tokenization, stopwords removal and lemmatization	126027	5888

Table 3: Statistics of Cranfield dataset with different preprocessing

After tokenization, stopwords removal and lemmatization, the scale of the corpus significantly decrease.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
tokenization only	0.2148	0.2906	0.1307	**0.0884**	0.2336	0.263
tokenization and stopwords removal	**0.216**	**0.2909**	**0.1324**	0.0876	**0.2356**	**0.2632**
tokenization, stopwords removal and lemmatization	0.2113	0.2853	0.1307	**0.0884**	0.2304	0.2604

Table 4: Trec eval results for different preprocessing on Cranfield dataset

We apply these preprocessing on cranfield dataset and train on BM25 model from terrier, then evaluate by trec eval and giving results in Table 4. In most metrics, preprocessing with tokennization and stopwords removal performs the best, while tokenization only and all three preprocessing process come out similar results. Overall, the difference in results are not significant. However, we can still conclude that lemmatization might be over-killed for this information retrieval tasks or for this specific dataset. The variety of word forms might also contribute to the matching of documents, for example, the tense.

6.3 Topic Modelling

In this section, we try some semantic approaches to solve the information retrieval task on Cranfield. Topic modelling is prevailing in many text classification tasks. We use LDA topic model as well as the its hierarchical variant - HDP topic model to develop our information retrieval techiniques.

6.3.1 LDA Topic Modelling

Gensim(11) is an open source library written in python, providing implementations of any topic models. With the implemented LDA model at hand, we still need to tune the hyper-parameters to ensure that the LDA model fits our dataset. In our case, the only hyperparameter is the number of topics we want to model. The number of topics is a latent variable which is unknown. We don't know how many topics can best categorize these documents. Therefore, we need to search for the optimal number of topics so that the LDA model performs the best. Here we use the intrinsic metrics mentioned in section 4.1.2 - perplexity and coherence - to guide us tuning the number of topics.

Hyperparameter tuning We search for the best number of topics in a range of integers from 5 to 30 and plot the perplexity and coherence, as Figure 8 shows.

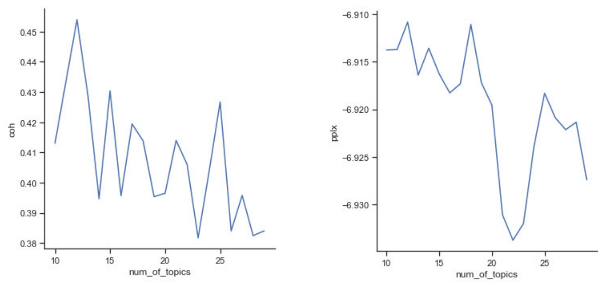


Figure 8: Tuning the Coherence and Perplexity of LDA Topic Model

As you can see in the picture, the coherence score peaks at 12 while the perplexity reaches the minimum at 23. As these two measure don't make consensus on the number of topics, we need to trade off between perplexity and coherence. However, the scale of coherence is substantially larger than the scale of perplexity, which means a difference in coherence is more significant than a difference in perplexity. Therefore, the coherence score dominants the hyper parameter selection. We choose 12 as our optimal number of topics.

Figure 9 shows a visualization by pyLDAvis(27) library of LDA model with 12 latent topics on our cranfield dataset. This interactive visualization allows you to hover on the circle on the left-hand side. The bar chart on the right will shows the most relevant terms in this topic correspondingly.

Each circle in the two-dimensional space on the left represents a topic. The radius of the circle indicates the number of relevant terms in this topic, indicating the amount of contents in the dataset that are covered by this topic. As the representation of the topic is multidimensional, a Principal component analysis is applied - the horizontal axis and vertical axis are two most significant principal components named as $PC1$ and $PC2$. This is how a inter-topic distance map is structured. If two circles are isolated with each other, there are little coherence between these two topics. In the contrary, if two circles are close or even overlapping with each other, they might have high cohesion. That is to say, relevant terms under these two topics usually occur in the same context. In general, the ideal distribution of the topics in the inter-topic distance map is that circles are spread across this space and isolated with each other. Because we want the topics distinct with each other, which is interpreted by the graph as little overlaps between the circles. We achieve this to some extends with the optimal latent number of topics. As you can see in this graph, topic 1 covers the topic 8 which is bad and partially overlaps with topic 7 and topic 3, except which other topics are desirably isolated.

On the right-hand side, the bar chart simply shows the term frequency of the relevant terms under the selected topic. You can apparently compare the overall term frequency and the estimated term frequency within the selected topic. The variable λ is used to balance the saliency and the relevance. In our case, we want the terms within the selected topic to be specific to this topic, so we set λ close to 1 to weight relevance over saliency.

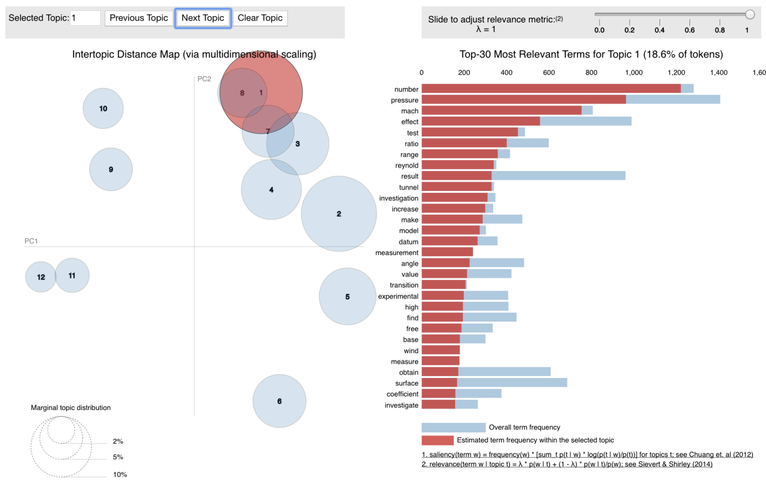


Figure 9: Visualization of LDA Model

Our experiments are all based on the LDA model with 12 as the number of latent topics. We use the document-topic-frequency matrix generated by LDA model to yield relevance score of query document pairs. Two approaches were purposed: one is document clustering; another one is divergence based similar document search.

Document clustering Here we use the concatenated document-topic-frequency of queries and documents as the input, and generate clusters to match them. One approach is intuitive. We take the topic of highest frequency as the predicted topic of the selected document, and group documents of the same predicted topics together. In this case, we must set the number of latent topics in our LDA topic model to be the number of queries in our dataset. By doing so, our model will generate 225 topics. This is based on the assumption that each query is a topic per se. But in the reality it is usually not the case.

Another approach is using the traditional clustering algorithm - KMeans. The topic frequencies are treated as features in KMeans model. In our 12-topic LDA model, it will generate clusters within 12-dimensional feature space. Similarly, based on the assumption stated above, we want the number of cluster be the same as the number of queries in the dataset. So we specify the number of clusters as 225 in the KMeans model and generate the following results.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
most frequent	0.0139	0.0277	0.0053	0.0116	0.0071	0.0206
KMeans	0.0014	0.0059	0.0027	0.0013	0.0031	0.0003

Table 5: LDA Document clustering results

As you can see in this results. The performance of this approach is worse than the baseline.

For the most frequent approach, one probable reason is that when we leave out the topic frequency and squeeze them into either 1 or zero, we loose the information. In a numerically way, the information entropy goes down during this process. With less information in our model, we cannot distinguish documents perfectly, resulting in bad matches to the query. Furthermore, assuming the Cartesian product of queries and documents within the same topic as relevant is just too general. We cannot expect much performance with such large granularity similarity measure.

For the Kmeans approach, as we use the optimal model with 12 topics, the dimension of feature space is restricted to 12. If we increase the dimension, we have to accordingly increase the number of latent topics in our LDA model. This will definitely ruin the model.

Divergence based approach The experiment above is to prove that the divergence based approach is better. The statistical assumption of LDA topic model (28) says that documents are arising from multiple topics, where a topic is dened to be a distribution over a xed vocabulary of terms. Given topics are probability distributions, we can treat the topic frequencies as a joint distribution among all latent topics. Based on which, the dissimilarity between two documents can be measured by divergence metrics, such as Jensen Shannon Divergence.

In our Experiment, we calculate the Jensen Shannon Divergence of each query-document pairs, taking the topic frequency as the input vector. And then sort the inverse of the results to derive a rank of candidates of relevant documents. As a comparison, we also try euclidean distance as the distance measure to see that how distance measure matters. The results are as follows.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
LDA JSD	**0.5097**	**0.5518**	**0.304**	**0.2004**	**0.5333**	**0.5863**
LDA EUD	0.4768	0.4931	0.2942	0.1991	0.494	0.5568

Table 6: LDA Topic model results on Cranfield dataset

The scores of each evaluation metrics are way better than any other models. Moreover, Jensen Shannon divergence is better than euclidean distance as we expected.

6.3.2 HDP Topic Modelling

The implementation of the HDP Model was adapted from (17). With the gensim toolkit of python, we solely need the corpus as a parameter for the model. The preprocessing is similar to the preprocessing for the LDA topic modelling. The output of the HDP model shows the number of topics as well as the most probable words in that topics including their probabilities. Surprisingly, the model returns 150 topics, way more than we have set for the LDA Topic Modelling. On the left of Figure 10 we can see the distribution of the topics as well as their overlaps. We recognize three big topics among our 1400 abstracts and several small topics among which some are included in other topics. The right side of the figure shows the most probable words for topic two with their overall (red) and estimated (blue) term frequency.

After setting up the HDP model, we transferred it to an LDA model in order to use the *Jenson Shannon Divergence* again. Table 7 shows the result according to the metrics MAP, MRR, Precision and NDCG.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
HDP JSD	0.4096	0.4142	0.2963	0.1938	0.4255	0.5013
LDA JSD	**0.5097**	**0.5518**	**0.304**	**0.2004**	**0.5333**	**0.5863**

Table 7: Trec eval results for the HDP Model and LDA Model

In the first row of the table we see the results for the metrics according to the HDP model. The second row of the table recaps the result mentioned in the previous section. Both models are using the Jenson Shannon Divergence to calculate the relevance scores. Unfortunately, for all metrics, the HDP model

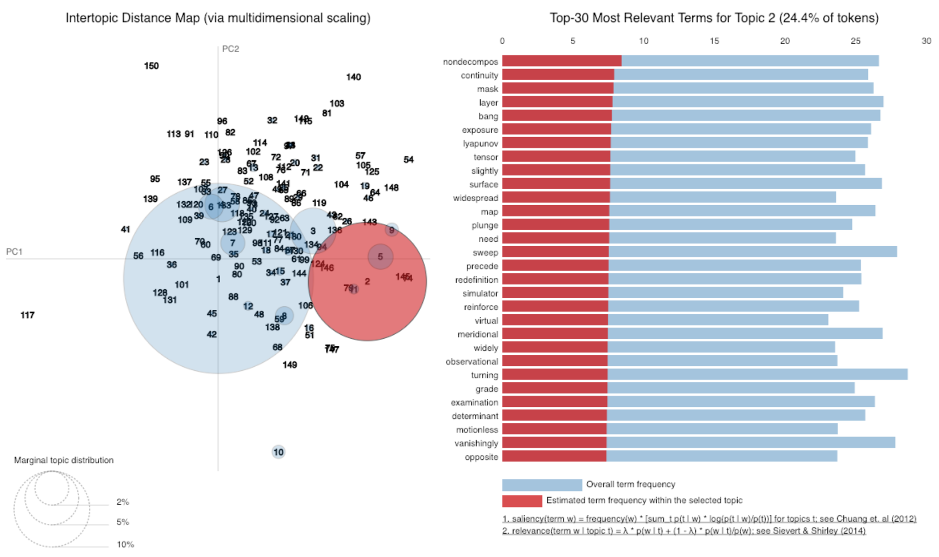


Figure 10: Visualization of the HDP Model

shows worse results than the LDA model. In the previous section, we stated that our goal was to achieve as less overlaps between the topics as possible. Figure 10 depicts a high number of overlaps and inclusions of topic. All in all, it turns out that the HDP Model does not seem appropriate for our use case of information retrieval.

6.3.3 Conclusion

In this section, we see how LDA and HDP topic model perform on our document-query matching task. Generally, LDA model yields satisfying results regarding to all evaluation metrics. From the statistic point of view, the document topic frequency matrix generated by LDA topic model captures well enough semantic features of the corpus. With the help of distance measure like Jensen Shannon Divergence, we can easily exploit these feature and perform accurate matching. The HDP topic model offer an alternative solution to topic modelling, as it look into the hierarchical structure of the clusters. However, there are still some drawbacks from these topic models. One thing is that the text length is not sufficiently long for a perfect modeling. Most texts in Cranfield dataset has only around 200 words. LDA always performs better with longer texts. Another drawback is that the number of topics is hard to tune. As you can see the experiments on both LDA topic model and HDP topic model, we do a lot of trade-offs to find the best parameter. Nevertheless, the number of topics is either too general or prone to overfit the dataset.

6.4 Terrier

6.4.1 Data Preparation

Terrier defines the queries as topics and supposes them in the following XML format (18):

```
<TOP>
<NUM>1</NUM>
<TITLE>query</TITLE>
</TOP>
```

Analogously, the documents are also transformed in XML format as follows:

```
<DOC>
<DOCNO>1</DOCNO>
document text
</DOC>
```

Afterwards, we index the documents and specify the following properties before batch retrieving

- trec.topics: The path to the queries
- trec.model: The retrieval model's name according to (18)

Note that, after batchretrieve, the Terrier models rank way more documents for a single query than in the ground truth file. The ground truth file contains ranking in an interval of $r \in \{1, 2, 3, 4, 5\}$. In contrary, the ranks produced by BM25 are consecutive numbered not limited. Based on this setting, Trec Eval produces bad results as in the retrieved document, there exists many ranked documents for a given query that are non existent in the gold standard file. Therefore, we need to merge the ground truth results with the results of the according Terrier method on query_id and doc_ID. This way, for every query, we have the same number of retrieved documents.

6.4.2 Results

After the data is prepared, we specify the Trec topics and the Trec model. First, we retrieve English documents for English queries. On Table 8, the results of all methods described in section 4.3.4 are listed. The columns show the four metrics MAP, MRR, Precision and nDCG. For Precision and nDCG we consider only the first five and ten retrieved documents as this is sufficient for our setting. The rows indicate the models that are available on the Terrier platform. If we look at the MAP values, we assume that the results of all exact-matching baselines are quite similar ranging in the interval of $MAP \in [0.443, 0.4544]$. Analogously, the other metrics show similar results across all methods as well. In comparison to the presented results in section 1.3.3, the Terrier methods increase the metrics MAP and MRR significantly. While in (1) the MAP has the value 0.131, the model DLH produced the best MAP value of 0.4529. We also improved the

MRR metric from the master thesis that has a value of 0.396. Hiemstra LM produced the best value for MRR with 0.4796. In expC2 is the best model referring P@10. Finally, the model PL2 outperforms all the other methods in P@5 (0.2871), nDCG@5 (0.4825) and nDCG@10 (0.5415).

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
BM25	0.4445	0.466	0.2836	0.1973	0.4706	0.5354
BB2	0.4503	0.4758	0.2809	0.1964	0.474	0.5402
DFR BM25	0.443	0.4651	0.2836	0.1969	0.4693	0.5336
DLH	**0.4529**	0.4773	0.2853	0.1964	0.4774	0.5409
DLH13	0.4475	0.4747	0.2809	0.1964	0.4702	0.5375
DPH	0.4478	0.4763	0.2862	0.1956	0.4772	0.5364
DFRee	0.4479	0.4727	0.2782	0.1951	0.4702	0.5356
Hiemstra LM	0.4518	**0.4796**	0.2836	0.1964	0.4751	0.5412
IFB2	0.443	0.4646	0.28	0.1969	0.4661	0.5339
In expB2	0.4445	0.4724	0.2836	**0.1978**	0.4718	0.5374
In expC2	0.4427	0.4685	0.2844	0.1937	0.4718	0.5352
InL2	0.4473	0.4757	0.2809	0.196	0.4719	0.5364
Lemur TF-IDF	0.4473	0.4705	0.2836	0.1969	0.4724	0.5377
LGD	0.4478	0.4767	0.2764	0.1956	0.4673	0.5364
PL2	0.4544	0.4812	**0.2871**	0.196	**0.4825**	**0.5415**
TF-IDF	0.4464	0.477	0.28	0.1951	0.4706	0.5354

Table 8: Trec eval results for Terrier on the English dataset

After processing the English dataset, we apply all methods on the translated Cranfield collection. The results are listed on Table 9. Not surprisingly, with the model BM25 we get the best MAP value of 0.4353. Lemur's TF-IDF model outperforms referring MRR with a value of 0.4873. Precision with the first ten retrieved documents reaches its best value off 0.1987 when we apply the model LGD. Analogously to the English dataset, PL2, again, outperforms all methods in P@5 (0.2889), nDCG@5 (0.4784) and nDCG@10 (0.5420).

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
BM25	**0.4353**	0.4623	0.2836	0.196	0.4604	0.526
BB2	0.4523	0.4870	0.2818	0.196	0.4742	0.4403
DFR BM25	0.4343	0.4609	0.2827	0.196	0.4587	0.5252
DLH	0.4323	0.4767	0.2853	0.1982	0.4634	0.5302
DLH13	0.4414	0.4819	0.2844	0.1978	0.4662	0.5344
DPH	0.4412	0.4744	0.2844	0.1973	0.467	0.5322
DFRee	0.4447	0.4789	0.288	0.1982	0.4729	0.5357
Hiemstra LM	0.4449	0.4806	0.2862	0.1973	0.4696	0.5357
IFB2	0.4297	0.4486	0.2738	0.1929	0.4495	0.5198
In expB2	0.4472	0.4782	0.2836	0.1969	0.471	0.5351
In expC2	0.4477	0.4803	0.2836	0.1964	0.4709	0.5353
InL2	0.4489	0.4859	0.2844	0.1964	0.4736	0.5365
Lemur TF-IDF	0.4476	**0.4873**	0.2827	0.1964	0.4717	0.5362
LGD	0.4488	0.4834	0.2853	**0.1987**	0.4754	0.5398
PL2	0.4514	0.4857	**0.2889**	0.1982	**0.4784**	**0.5420**
TF-IDF	0.4464	0.477	0.28	0.1951	0.4706	0.5354

Table 9: Trec eval results for Terrier on the German dataset

Both the metrics for the English as well as for the German dataset outperform the metrics MAP and MRR from the master thesis. The highest German metrics results, except P@10, are slightly higher than the highest English metrics results. Precision for ten retrieved documents solely is better on the English dataset. Nevertheless, the results are all quite high.

6.4.3 Conclusion

In this section we introduced the results of the exact matching methods that were available on the platform Terrier. We noticed that all metrics resulted from all these methods outperformed the metrics results from the master thesis presented in section 1.3.3. On the English dataset, the models DLH13, Hiemstra LM, IN expC2 and PL2 performed the best. Surprisingly, BM25 is not included among the models on the English dataset. For the German dataset, the models BM25, Lemur's TF-IDF, LGD and again PL2 show the highest results. In addition to that, in four of five metrics, the German results are higher than the English ones. Solely in Precision for the first ten documents, the English metric shows a slightly higher value. In conclusion, the methods from Terrier show for both English and German dataset high metric results even though they are exact matching methods. We compare these results in section 6.6 where a neural approach is introduced.

6.5 Word2vec Phrase detection

6.5.1 Model description

Word2vec published a C code that detects phrases as described in section 3.1.1. The commented and unaltered code was taken from Github (12) in order to train it on the queries and documents. The model takes as an input the following two parameter: a raw text, i.e. the query, and a threshold. Last mentioned is set to 100 as default. The higher the threshold, the less phrases are built. This is relevant for short text as they might contain phrases that do not occur often in general natural language. The output of the model is the text with phrases marked with an underscore between the words. As an example we consider the phrase "heat transfer" as one parameter. In the output text, the model would then, if the phrase score is higher than the threshold, return the words as a single one: "heat_transfer". We used the word2phrase.c code in order to detect the phrases.

6.5.2 Preprocessing

Before we add the phrases to our vocabulary, we first remove stop words and lemmatize as well as tokenize the queries and documents. With nltk, we can tokenize and lemmatize the English dataset. Additionally, the toolkit provides a set of stop words for English and German. When it comes to the lemmatization and tokenization of the German dataset, we use the open source library spaCy as nltk does not provide both for German. For both datasets we evaluated once on the raw dataset and once on the lemmatized and tokenized dataset where stop words are removed.

6.5.3 Parameters

In section 4, we introduced some Terrier methods in order to retrieve the documents. As these methods are count-based, we use word2vec's phrase detection model on both sides, the queries and the documents. Referring the parameters, we tried out several thresholds for queries and documents.

Train	The training data
Output	The output file
min-count	This parameter sets a default limit of 5. every word that appears less than this limit is discarded.
threshold	Varies according to the train data. In total, we have 255 queries that are mostly one sentence long. Lots of questions start with a question word and a verb such as *What is* or *Can a* that should not be detected as a phrase. With both limitations, we cannot set the threshold very high. Therefore, we tried to set the threshold to 20, 30, 40 and 50. With a threshold of 50, the model detects no phrases in the queries. Lastly, the information retrieval techniques show the best results with a **threshold of 40 for the**

queries.
As the documents dataset is larger, we can set the threshold higher than for the queries. Our experiments relied on the thresholds 80, 150 and 300. The best results with the chosen information retrieval techniques are based on a **threshold of 150 for the documents**.

debug Shows more information during training. The default of 2 works well for our case.

6.5.4 Results

Referring the information retrieval methods, we chose to process the best Terrier methods for the according language mentioned in section 6.4.2. Since precision does not consider the rank, we focus on the models that performed the best for MAP, MRR and nDCG. For the English dataset, these are the methods DLH, Hiemstra_LM and BM25. As the methods did not prove to have significantly different result on the original dataset, we depict the results of the lemmatized and tokenized dataset where stop words are removed. In Table 10 you can see the results in respect to the metrics introduced in section 5. The results of the documents and queries with phrases do not differ a lot the original results that were applied on the dataset without phrases. All metrics slightly decreased or remained the same in their values. An exception is the metric Precision for the first five retrieved documents. The model PL2 shows a small increase for this metric with the highest value for P@5 so far: 0.2989. The strongest models in this section are Hiemstra LM and the PL2 outperforming DLH in MAP, MRR and nDCG. This was not the case with the original dataset. Unfortunately, adding phrases to both queries and documents does not return more relevant document as expected. Therefore we conclude that phrase detection does not improve the information retrieval results for the English dataset.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
DLH	0.4392	0.4628	0.2853	**0.1964**	0.4676	0.5307
Hiemstra LM	0.4457	**0.4759**	0.2836	0.1956	0.4721	**0.5363**
PL2	**0.4458**	0.4666	**0.2898**	0.1942	**0.4761**	0.5328

Table 10: Trec eval results with word2vec's phrase detection on the English dataset (with lemmatization, stop words removal and tokenization)

For the german dataset, we went through the same process, but used the Terrier methods BM25, DLH and PL2 as they performed the best on the original dataset. Below, you can find the evaluation for these models. Table 11 shows the results for the methods on the raw dataset, that means without lemmtization and stop words removal. We can clearly see that PL2 outperforms the other methods with the highest values in MRR, P@5 and NDCG. Furthermore, on Table 12 we listed the results for the lemmatized dataset where stop words are removed. Comparing the two tables, we can see that in MAP, MRR and nDCG@10, the lemmatized dataset always has better results. For precision and nDCG@5 some metrics show slighty lower values. As we do not focus on precision, we can say that overall the phrase detection is advantageous if we lemmatize the data and remove its stop words first. In the following, we compare Table 12 with the original results of Table 9. In contrast to the English results, the German phrase detection sometimes show an increase in the metric values. For example BM25 always outperforms the original BM25 results except for precision. Unfortunately, for the Lemur's version of TF-IDF we always have slightly lower values except for precision and nDCG@10. PL2 solely shows for MAP and precision at five documents higher metric values. With phrase detection, BM25 clearly outperforms the other two methods for the German dataset. This is contrary to the original results where BM25 solely outperformed for MAP, Lemur's TF IDF is the best in regard to MRR and then PL2 taking the third place among the methods.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
BM25	**0.4554**	0.466	0.2836	**0.1973**	0.4706	0.5353
Lemur TF-IDF	0.4473	0.4705	0.2836	0.1969	0.4724	0.5377
PL2	0.4544	**0.4812**	**0.2871**	0.196	**0.4825**	**0.5415**

Table 11: Trec eval results with word2vec's phrase detection on the German dataset (without lemmatization and stop word removal)

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
BM25	**0.4607**	**0.4888**	0.2818	0.1956	**0.479**	**0.544**
Lemur TF-IDF	0.455	0.4855	**0.2844**	0.1956	0.4772	0.54
PL2	0.4599	0.4858	0.28	**0.1969**	0.4763	0.5438

Table 12: Trec eval results with word2vec's phrase detection on the German dataset (with lemmatization)

6.5.5 Conclusion

Applying the phrase detection from word2vec for the English dataset, we did not gain benefits to the information retrieval results, i.e. the metrics remain about the same. In contrast, the results for the German dataset the metrics showed slightly higher values. We expected that state-of-the-art methods may improve by including brigram search. Contrary to our expectations, we solely achieved small improvements. This may reasons on the limited size of the Cranfield dataset. On the one hand, the documents are relatively short and on the other hand, the dataset does not provide enough documents in order conclude if the phrase detection could improve information retrieval. With a bigger dataset, running count-based methods on bigrams may be more advantegous and can even be extended to trigrams.

There is one drawback of exact matching methods when dealing with phrases: Humans do not annotate phrases uniformly. This way, some phrases are marked with a line and some are not marked at all, having a space between the phrasal words. Consider the phrase "boundary layer". Word2vec's annotation for this bigram would be "boundary_layer". After investigation, we noticed that this phrase is sometimes separated through a space or a line, e.g. "boundary-layer". Exact matching methods would then not match the word2vec bigram to that phrase even though they contains the same words. Having a uniform representation of a phrase would simplify the information retrieval.

6.6 Transfer Learning

We experiment with two different settings of transfer learning. In the first setting we train plain Duet model without adversarial discriminator on each of three datasets separately and test it on Cranfield dataset. In the second setting we combine all three datasets to one big shuffled dataset and train Duet model with adversarial discriminator applied to it.

6.6.1 Preprocessing

As datasets used for training are in different formats we have to preprocess them, so that all queries are of the same length, which is required for the Duet model. Queries, that are shorter than maximum length are padded with a dummy word. The same padding is applied to documents analogously. Moreover, each of the datasets has its specific preprocessing steps that are listed below.

After preprocessing, the desired dataframe have these columns: query id, document id, and their binary relevance. As we use a convolutional network, the batch input data would have the same length. Therefore we need to truncate the length of queries and documents to the 95% quantile of the corresponding length in the target dataset - Cranfield, which is 25 for the queries and 720 for the documents respectively.

WebAP The candidate documents in WebAP dataset do not contain relevance annotations (only paragraphs of these documents were annotated). Hence we split all documents into paragraphs and treat them as query-paragraph pairs for training. Moreover, WebAP dataset contains non-binary relevances for paragraphs. As our Duet model of adversarial domain adaptation method works with binary relevances only, we had to come up with conversion strategy. All paragraphs of WebAP dataset, which were labelled as "irrelevant" were set to relevance 0, all others were marked as relevant and got relevance value 1.

InsuranceQA The insuranceQA(22) corpus provides questions and answers of different insurance domains. In total, there are 27,413 answers with 3,065,492 running words of answers. For each question, there is a ground truth answer. Irrelevant answers of a question are sampled from a pool given the pool size.

In our case, we choose the pool size of 100 with in total more than 2000000 question answer pairs, which is already large enough for our training purpose.

Yahoo L4 The Yahoo dataset contains questions, answers, category information and a context. As the dataset is formatted in XML the answer texts can include html tags, e.g. fat marked text, that are not necessary for our use case. We converted the XML file to dataframe where each question, answer pair is modeled in one row. There are no specific relevance annotations in the dataset but the ranked answers. Therefore, In order to assign binary relevance to the answers, we map the best answer of a question to a relevance of 1. As we only have 5.74 ranked answers per question in average, we map all other answers to a relevance of 0.

6.6.2 Results

The results of transfer learning approach are presented in Table 13. We trained every dataset on one epoch as the resulting difference to five epochs is not significant. We can see that, for the approach without adversarial discriminator, the best results were achieved with the Yahoo L4 dataset. The model trained on Yahoo L4 is stronger than other models on all metrics except for precision at 10, where the insuranceQA dataset was stronger. Training on the dataset WebAP shows the lowest results. This can be reasoned on the small siize of the dataset.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
WebAP	0.4047	0.4259	0.2587	0.1920	0.4191	0.4994
InsuranceQA	0.4062	0.4411	0.2622	0.**1929**	0.4259	0.5043
Yahoo L4	**0.4198**	**0.4564**	**0.2649**	0.1889	**0.439**	**0.5099**

Table 13: Transfer learning results on Duet Model

In Table 14, the results of the adversarial discriminator and the best scores of the separate training are listed. The training dataset is now the stacked dataset of WebAP, insuranceQA and Yahoo L4. Comparing the results on the single datasets with the adversarial discriminator results, we can see that the Duet model with domain regularization significantly outperformed other models on all metrics. The strongest difference was achieved on MAP, MRR and NDCG. This means that adversarial training is more effective for transfer learning on completely different domain.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
Duet adversarial network	**0.4403**	**0.4831**	**0.2658**	**0.1951**	**0.4537**	**0.5308**
Best scores in separate training	0.4198	0.4654	0.2649	0.1929	0.4139	0.5099

Table 14: Adversarial learning results on the stacked dataset

6.6.3 Comparison to exact matching methods

After retrieving the results from the duet adversarial network, we want to compare the neural approach with the exact matching models. On Table 15 you find the results of the two best exact matching models DLH and Hiemstra LM. As our focus referring the state-of-the-art baseline was originally the model BM25, the result for this retrieval method is listed as well. Additionally, you find the results of the duet adversarial network in order to directly compare the two baselines. Generally, the metrics show overall similar results for both the exact matching and the neural method. Nevertheless, the neural approach only shows better values with one metric that is the MRR of 0.4831. In all other metrics one of the three exact matching baselines outperformed the duet adversarial network. The DLH method clearly show the best results for MAP (0.4529), P@5 (0.2853) and nDCG@5 (0.4774). DLH and Hiemstra LM have both the best results for precision at ten documents with a value of 0.1964. Hiemstra LM also outperforms all other models in the metric nDCG@10 with a value of 0.5412. Surprisingly, BM25 turns out to be the weakest model in regard to its low metric values among the four listed on Table 15. In conclusion we can say that the exact matching baselines are definitely comparable to a neural approach and even outperform in most cases.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
DLH	**0.4529**	0.4773	**0.2853**	**0.1964**	**0.4774**	0.5409
Hiemstra LM	0.4518	0.4796	0.2836	**0.1964**	0.4751	**0.5412**
BM25	0.4445	0.466	0.2836	0.1937	0.4706	0.5354
Duet adversarial network	0.4403	**0.4831**	0.2658	0.1951	0.4537	0.5308

Table 15: Results of the best exact matching methods and the duet adversarial network

6.6.4 Conclusion

Based on the results we can conclude, that adversarial discriminative domain adaptation approach is successful in learning domain invariant features and achieves better results compared to transfer learning without adversarial discriminator. Comparing neural network approaches for transfer learning with exact matching and topic modelling we observe, that deep learning outperformed other models only on MRR (0.4831 vs. 0.4796), which means, that it was better at finding most relevant document. The biggest disadvantage of this approach is the need for time-consuming and expensive training, while exact matching models require no training and are extremely fast.

6.7 Comparison of all information retrieval techniques

In summary, Table 16 lists the results of the information retrieval techniques on the Cranfield dataset that we introduced in this report. All of our approaches exceeded the baseline model in all evaluation metrics, among which the LDA topic model performs the best. The exact matching approach with the Terrier models is some of the state-of-the art information retrieval model. They achieved substantial improvement to the baseline model on MAP and MRR. The duet adversarial network also achieves comparable results with those off-the-shelf models, improving the baseline model by 0.31 on MAP and 0.09 on MRR respectively. However, the neural network approach still have potential to improve, if we incorporate sophisticated feature engineering and hyper-parameter searching methods.

	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
DLH	0.4529	0.4773	0.2853	0.1964	0.4774	0.5409
Hiemstra LM	0.4518	0.4796	0.2836	0.1964	0.4751	0.5412
BM25	0.4445	0.466	0.2836	0.1937	0.4706	0.5354
LDA JSD	**0.5097**	**0.5518**	**0.304**	**0.2004**	**0.5333**	**0.5863**
Duet adversarial network	0.4403	0.4831	0.2658	0.1951	0.4537	0.5308
Baseline model	0.131	0.396	/	/	/	/

Table 16: Results of the all information retrieval techniques

7 Future work

In this work, we have seen exact based matching models as well as a neural approach for the cross lingual semantic search. Both approaches show similar results and are comparable regarding the metrics MAP and MRR and nDCG. We improved the MAP and MRR results of Balabel (1) for both approaches. Nevertheless, the resulting documents are different in exact based models and in the neural ranking, e.g. figure 17 gives an overview of top-5 results for the first query. For more informative comparison we evaluate the relevance of top results using original non-binary relevance in Cranfield dataset. The documents with highest relevance have score 5, with lowest -1. As it can be seen from the figure, only one same document was selected in top-5 for both of the models. Similar situation happens for all other queries.

Rank	BB2 doc_id	True relevance	Duet doc_id	True relevance
1	13	4	462	4
2	486	-1	184	2
3	56	3	30	3
4	142	4	66	3
5	184	2	121	3

Table 17: Comparison of top-5 predictions for exact matching and neural network model

Therefore, it could potentially be interesting to apply a mixed approach that combines the advantages of the exact matching methods and the neural model. Simple linear combination with equal weights does not bring any improvement, so more sophisticated combination methods are to be developed.

In addition to that, we dealt with the phrase detection tool of Word2Vec (2). As the Cranfield dataset is quite small, we solely considered bigrams. Unfortunately, the bigram detection did not result in high improvements as we expected. In the future, given a bigger dataset, one can detect longer phrases such as trigrams in order to increase exact matching results.

Another approach is to have a deeper look at the document structure, given a larger document set. We can develop a hierarchical data structure to preserve this document tree with following reasons.

First, The title of a chapter might be an abstraction of the text under its domain. This is beneficial to information retrieval because we can give weights to the word with regard to their document level. That means, a word that appears in the title of a chapter is more likely to relate to the topic than its equivalence in a single paragraph.

Second, this tree structure can accelerate searching. When we search for text related to a specific topic, we do not need to scan the document from the very beginning. Instead, we search the higher level of the document. Those texts in higher document level might contain more conclusive words which may help to narrow down the search area. The search algorithm can therefore unfold the sub-documents based on similarities between the query and the text in high level. As we iteratively perform beam search, we will finally go down to several paragraphs that might contain the information we want.

Third, this structure also allows a user to specify the granularity of the search result. We can stop Breadth First Searching at whatever document level and return either the whole document or a single page or simply one paragraph.

In section 4, we introduced BM25 and the Terrier IR Platform. Terrier also provides a BM25 extension called BM25F which is a per-field normalization model based on BM25. The idea of field-based weighting model is similar to our conception. For instance,if a query term occurs once in the body of the document, there is only a small chance that the document is really related to that term. However, if the term occurs in the title of the document, this chance is greatly increased. This structure might facilitate the analysis.

References

[1] M. Balabel, "Cleisst: a cross-lingual engine for informed semantic search in the technical domain," Master thesis, Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung, Germany, 2018.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] B. Mitra and N. Craswell, “Neural models for information retrieval,” *arXiv preprint arXiv:1705.01509*, 2017.

[4] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, “Semantic matching by non-linear word transportation for information retrieval,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 701–710.

[5] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson, “Terrier information retrieval platform,” in *European Conference on Information Retrieval*. Springer, 2005, pp. 517–519.

[6] D. Cohen, B. Mitra, K. Hofmann, and W. B. Croft, “Cross domain regularization for neural ranking models using adversarial learning,” in *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, ser. SIGIR ’18. New York, NY, USA: ACM, 2018, pp. 1025–1028. [Online]. Available: http://doi.acm.org/10.1145/3209978.3210141

[7] B. Mitra, F. Diaz, and N. Craswell, “Learning to match using local and distributed representations of text for web search,” in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW ’17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 1291–1299. [Online]. Available: https://doi.org/10.1145/3038912.3052579

[8] B. Mitra, E. Nalisnick, N. Craswell, and R. Caruana, “A dual embedding space model for document ranking,” *arXiv preprint arXiv:1602.01137*, 2016.

[9] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, “Retrofitting word vectors to semantic lexicons,” *arXiv preprint arXiv:1411.4166*, 2014.

[10] X. Rong, “word2vec parameter learning explained,” *arXiv preprint arXiv:1411.2738*, 2014.

[11] R. Řehůřek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[12] C. McCormick, “Github. chrisjmccormick/word2vec_commented.” [Online]. Available: https://github.com/chrisjmccormick/word2vec_commented

[13] C. McCormick, "Github. chrisjmccormick/inspect_word2vec." [Online]. Available: https://github.com/chrisjmccormick/inspect_word2vec

[14] Z. Lu and H. Li, "A deep architecture for matching short texts," in *Advances in Neural Information Processing Systems*, 2013, pp. 1367–1375.

[15] M. Steyvers and T. Griffiths, "Probabilistic topic models," *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

[16] N. I. of Standards and Technology, "Trec eval. usnistgov/trec_eval." November 2018. [Online]. Available: https://github.com/usnistgov/trec_eval

[17] C. Wang, J. Paisley, and D. Blei, "Online variational inference for the hierarchical dirichlet process," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 752–760.

[18] C. Macdonald and J. P. R. McCraedie, "Terrier ir platform - homepage," November 2018. [Online]. Available: http://terrier.org

[19] D. Cohen, B. Mitra, K. Hofmann, and W. B. Croft, "Cross domain regularization for neural ranking models using adversarial learning," *arXiv preprint arXiv:1805.03403*, 2018.

[20] M. Keikha, J. H. Park, W. B. Croft, and M. Sanderson, "Retrieving passages and finding answers," in *Proceedings of the 2014 Australasian Document Computing Symposium*, ser. ADCS '14. New York, NY, USA: ACM, 2014, pp. 81:81–81:84. [Online]. Available: http://doi.acm.org/10.1145/2682862.2682877

[21] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou, "Applying deep learning to answer selection: A study and an open task," *arXiv preprint arXiv:1508.01585*, 2015.

[22] M. Feng, "Github. shuzi/insuranceqa," October 2018. [Online]. Available: https://github.com/shuzi/insuranceQA

[23] Apache, "Apache solr home," February 2019. [Online]. Available: http://lucene.apache.org/solr/

[24] Y. Research, "Yahoo language data." [Online]. Available: https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

[25] Yahoo, "Yahoo answers," February 2019. [Online]. Available: https://answers.yahoo.com/

[26] E. L. Bird, Steven and E. Klein, "Natural language processing with python. o'reilly media inc." 2009. [Online]. Available: http://nltk.org/

[27] C. Sievert and K. Shirley, "pyldavis library." [Online]. Available: https://github.com/bmabey/pyLDAvis

[28] J. D. L. David M. Blei, "Topic models," 2009.