Enhancing Term-Based Document Retrieval by Word Embedding and Transformer Models

Farzana, Sheikh Mastura
Matriculation number: 3276883
25 November, 2021

A thesis submitted in partial fulfillment for the degree of Master of Science

Institute of Computer Science

Supervisors:

Dr. Andreas Hamm, German Aerospace Center(DLR)
Md. Rashad Al Hasan Rony, University of Bonn

Examiners:

Prof. Dr. Jens Lehmann, University of Bonn
Dr. Andreas Hamm, German Aerospace Center(DLR)

INSTITUT FÜR INFORMATIK RHEINISCHE
FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Prüfungsausschuss Informatik

universität **bonn** · Institut für Informatik · 53012 Bonn

Vorsitzender des Prüfungsausschusses

Prof. Dr. Thomas Kesselheim

Prüfungsamt:
Judith König
Tel.: 0228/73-4418
Fax : 0228/73-4788
pa@informatik.uni-bonn.de
Postanschrift: 53115 Bonn
Endenicher Allee 19a

www.informatik.uni-bonn.de

Erklärung über das selbständige Verfassen einer Abschlussarbeit/ Declaration of Authorship

Titel der Arbeit/Title:

Enhancing Term-Based Document Retrieval by Word Embedding and Transformer Models

Hiermit versichere ich, Farzana (Name / name), Sheikh Mastura (Vorname / first name),

dass ich die Arbeit – bei einer Gruppenarbeit meinen entsprechend gekennzeichneten Anteil der Arbeit – selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Sheikh Mastura farzana
Unterschrift/signature

Bonn, den 24.11.2021
date

Acknowledgements

I would like to begin by extending utmost gratitude to my mentor, supervisor and one of the examiners, Dr. Andreas Hamm (Institute of Software Technology, Intelligent and Distributed Systems, German Aerospace Center(DLR)) for his continuous guidance, unwavering support and critical feedback in every part and aspect of this thesis.

I am grateful to Md. Rashad Al Hasan Rony (University of Bonn, Fraunhofer IAIS), my second supervisor, for his assistance with the thesis defense and for his valuable advice on the thesis report.

I wish to thank Prof. Dr. Jens Lehmann (University of Bonn, Fraunhofer IAIS) for agreeing to become the primary examiner of this thesis.

I am thankful Dr. Mark Azzam (Head of the Department, Think Tank, DLR) for giving me the opportunity of conducting my graduate thesis at DLR and for his valuable insights in the initial phase.

I am grateful to Mr. Andreas Schreiber (Head of the Department, Institute of Software Technology, Intelligent and Distributed Systems, DLR) for allowing me to continue my work in his department.

I would like to thank my friend, Ms. Aniqa Zaida Khanom for her help in making the cover page of this report.

Lastly, I am ever grateful to my Mother (Ms. Farhana Yesmin) and my late Grandmother (Mrs. Morium Khatun) for believing in me throughout my life.

Abstract

Document retrieval implies the process of obtaining most relevant documents based on some query from a large corpus of documents. Traditional document retrieval methods focus on the existence and/or non-existence of the query in a particular document to assess relevance of the document to the query terms. However, this approach does not guarantee relevance. For example, a document can be contextually relevant to some query without containing the exact query words or the document might contain the query term and still be about some completely different topic. Hence arises the need of context aware document retrieval systems. In this thesis, we focus on enhancing document retrieval methods in order to capture the contextual relevance of a document to a certain query. The primary components used to achieve our goals are word embedding models and transformer based pre-trained natural language models (details follow in later sections). Here, we have proposed three different approaches for enhancing document retrieval methods. We use three different datasets to evaluate our models and compare the results with classical document retrieval models. Our first method, zero-shot learning with transformer based pre-trained NLP models has the best scores. However, it is computationally very expensive. On the other hand, the proposed similarity score based relevance ranking method successfully surpasses the performance of traditional retrieval methods on multiple performance metrics. At the same time, it is significantly faster and robust. Our third approach, similarity score based logistic regression also performs better than the traditional models, however, on the basis of recall scores, the similarity score based relevance ranking model performs better.

Contents

1 Introduction

1.1 Motivation

The objective of document retrieval is to find documents that are most relevant to a certain query (information need) from within a large corpus of unstructured texts. In this thesis we focus on mapping unlabeled documents to the right subjects where we concentrate on the contextual (semantic) content of the documents.

The well established methods in the context of document retrieval usually uses Boolean combination of keyword searches and ranking functions based on Bag-of-words vector models such as TF-IDF [36, 27] or Okapi-BM25 [41]. However, estimating relevance of a document based on the existence and non existence of particular terms is not optimal for the following reasons:

1. Some documents might not contain a term but some semantically close word which is not contained in the query.

2. Some terms might be good indicators but no guarantee for relevance.

Some of the possible solutions would be to use some related terms along with the search term and use Boolean search with 'logical OR' to retrieve documents that does not contain the search term but are relevant. Alternative approach is doing a BM25 ranking using the actual term along with its related terms. Lastly, using a subject dependent supervised learning, this option is not viable at all as there will not be sufficient training data for all the subjects we want to deal with.

1.2 Problem Statement

In the context of subsection 1.1, the main problems of classical document retrieval techniques we want to address are,

1. Inability to retrieve documents that are contextually relevant but do not contain the search term.

2. Inability to assess relevance of documents that contain the search term.

In this thesis, we begin with assessing the viability of simple Keyword Search as well as Okapi-BM25 ranking. Then we move on to assess the performance of pre-trained transformer models for zero-shot classification, which can be used for document retrieval. Lastly, we use word embedding models to gather similarity scores between words of a document and search terms and use that score for document ranking using two different approaches. The results from all the mentioned methods has been discussed in section 5.

1.3 Contribution

The main contributions in this thesis are as follows:

1. Comparative study on transformer based different pre-trained and fine-tuned NLP models for document retrieval using zero-shot classification.

2. Brief study on the effect of different *Hypothesis* sentences used in the classification pipe-line of zero-shot classifiers.

3. A novel relevance ranking scheme using word embedding based similarity scores for keyword based document retrieval. Additional study on performance of different word embedding models.

4. Regression based document ranking method using similarity scores acquired from combinations of several word embedding approaches.

1.4 Report Organization

The thesis is organised in the following sections:

- Motivation and Problem statement. (Introduction) (section 1)
- Related work regarding relevance ranking and zero-shot learning. (Related Work) (section 2)
- Discussion on methods and models used in this project. (Background) (section 3)
- Elaboration on the datasets, data pre-processing and research methodologies used. (Methodology) (section 4)
- Model setup and related results. (Experiments) (section 5)
- Comparative summary of all results from different approaches. (Discussion) (section 6)
- Conclusion and suggested future work related to the thesis. (Conclusion) (section 7)

All the scripts related to this thesis can be found at `https://github.com/SheikhMasturaFarzana/Master-Thesis`.

2 Related Work

Document retrieval is an integral part of information storage and efficient information retrieval. Different document retrieval methods have been around for decades. Some of the earliest approaches of document retrieval are based on full text search. The approach implies that a certain query term is searched amongst all the available documents and the documents containing the query is retrieved. However, in 1986 Blair et al. argued that regardless of the apparent advantages of full text search, the effectiveness of the approach is quite poor [2].

In 1995 Chen et al. proposed a knowledge based fuzzy information retrieval method [7]. They based the concept on weighted queries and weighted interval queries to retrieve documents using transitive closure of concept matrices. The concept matrices contain a set of concepts and relevance value of those concepts to a set of documents. The user desired documents related to some concept C_j is assigned a strength value and the query is assigned a weight in accordance with concept C_j hence producing a weighted query. This fuzzy information retrieval method is capable of handling uncertain information unlike its predecessors.

The Probabilistic Relevance Framework (PRF) was conceptualised in the 1980s by Robertson et al [42, 41]. They came up with a theoretical framework for probabilistic models and can be used for information retrieval. PRF became the basis of many probabilistic relevance ranking models for information retrieval including the well know BM25 ranking model. PRF aims to estimate the probability of relevance of a document against some query. The BM25 document retrieval method ranks documents based on the appearance and frequency of a query term in each document of a corpus (details follow later subsection 3.3.)

Information retrieval functions based on the assumption of the existence and non-existence of a certain query term in a document, are not capable of extracting documents that are contextually relevant to the query but do not contain the query word itself. One possible solution in this direction is query expansion. In 2017, Liu et al presented their work for the enhancement of word embedding similarity measures using fuzzy query expansion rules [34]. Alongside the original query term, they also considered the $top - k$ similar words of query q from the word vector embedding space (generated using Word2Vec) using cosine similarity as the similarity measure. They later use fuzzy rules to re-weight the expanded queries based on the weight of the original query.

Qiu et al. proposed a word embedding based fuzzy information retrieval approach [39]. Instead of relying on query term frequency in a document, their model depends on word embedding model to calculate the similarity between each query term against each word in a document. Based on the similarity scores they assign a membership score to each document for some query and present the documents with higher scores as the retrieved documents.

We can comprehend from the above works that both relevance ranking and word similarity measurement based information ranking models can be useful for successful document retrieval. However, these techniques are not always capable of capturing the whole context of a document for a specific query. A combination of relevance ranking model and word embedding based similarity scores offers the possibility of handling the problem of retrieving contextually relevant documents. In later sections we have described our proposed models in this direction and compare our results with some of the established traditional document retrieval models.

One other interesting approach towards efficient document retrieval can be zero shot learning. Zero shot learning methods are capable of classifying classes that the model has not encountered during training. In 2013, Elhoseiny et al. proposed a zero shot learning method for image classification using textual description [12]. They adapted a regression model to classify new data based on the learned classification. Additionally, they also explored knowledge transfer from textual to visual domain.

In 2020, Shaheen et al. presented their work on large scale document classification using transformer models such as BERT, RoBERTa, XLNet etc [44]. With the combination of these classifier models along with strategies such as generative pre-training, gradual unfreezing and discriminative learning, they were able to outperform then state-of-the art models on two different datasets. In the same year, Pelicon et al. published their paper on cross-lingual

sentiment classification using zero-shot learning. The authors used an intermediate training step to improve the BERT model to get better input representations for sentiment classification. This improved model was tested by them in the cross-lingual zero-shot classification task without further training in the target language. The model was able to achieve improved results on both monolingual and cross-lingual sentiment classification task.

Adhikari et al. showed that their model DocBERT, based on a pre-trained BERT model (transformer based NLP model) is capable of reaching state-of-the-art performance on different models [1]. Although it can be argued that BERT has some drawbacks in this context such as a document can be larger than the BERT input size. However, their paper was the first indication that transformer based pre-trained can be successfully used for document retrieval. The PatentBERT model, proposed by Lee et al. is also based on the BERT model and is fine tuned to classify patent data [32]. The model is capable of surpassing state-of-the-art approaches in the same task.

It is evident that the transformer based models are powerful and capable of handling various NLP related problems including document retrieval. The combination of transformer based NLP models and zero-shot learning is the most promising approach for classifying large number of un-annotated text and has the the potential capacity of relevant document retrieval without the need of further training. In later sections, we experiment with zero-shot learning based document document retrieval approach with different transformer based pre-trained NLP models.

3 Background

In this section we explain the theoretical background of the computational architectures as well as algorithms used in this project.

We begin with discussing Natural Language Processing, document retrieval and move on to elaborate the mechanism of BM25, Word Embedding Models, Transformer Models and Dense Passage Retrieval (DPR),.

3.1 Natural Language Processing

Natural Language Processing (NLP) is a combined sub-field of Computer Science, Artificial Intelligence (AI) and Linguistic. The main goal of NLP is to automatically process, analyze and represent natural languages otherwise known as naturally evolved languages. In the present era of data, a large part of it consists of highly unstructured textual and audio data containing human language. The ability of properly processing these data will make trend analysis, different types of forecasting and interaction with artificially intelligent entities (eg. smart assistants, self-driving cars etc.) much easier. Some of the main challenges in NLP are, Natural Language Inference (NLI), Natural Language Understanding (NLU), Speech Recognition, Text Classification etc.

3.2 Document Retrieval

Document retrieval is a branch of information retrieval and by extension natural language processing, that aims at retrieving documents based on some query from a corpus of mainly unstructured texts. The main tasks of a document retrieval system are, identifying the documents that match a certain query, evaluate the results and rank them according to relevance. Form based retrieval focuses on finding text that contain specific syntactic features such as exact query sub-strings. The other form of document retrieval focuses on content of a document and focuses on semantic properties. In our project we explore possible improvements of the content based document retrieval approach.

3.3 Okapi-BM25

BM25 or Best Matching25 algorithm is a ranking function based on probabilistic retrieval framework (PRF) developed in the 1980s [41]. BM25 is a bag-of-terms retrieval method. Bag-of-term model regards multiplicity of words in a sentence but does not consider order of word appearance or grammar. There are several versions of the method with small differences in parameters. One of the notable instances of the function is explained with Equation 1. Given a query Q, containing keywords $q1, ..., qn$ the BM25 score of a document D is

$$Score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,D)(k+1)}{f(q_i,D) + k(1 - b + b(\frac{|D|}{avgdl}))} \tag{1}$$

In Equation 1,

- IDF or *Inverse Document Frequency* indicates presence of a query across the whole data corpus in consideration.

$$IDF(q_i) = \ln(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1) \tag{2}$$

  - N is the total number of documents.
  - $n(q_i)$ is the number of documents containing keyword q_i.

- $f(q_i, D)$ is the term frequency of q_i in document D.
- $|D|$ is the length of the document and $avgdl$ is the average length of documents in the document corpus.
- k is the saturation to avoid rewarding higher term frequency limitlessly.
- b is the length normalization term in order to keep into consideration that a certain query is more likely to appear in a longer document. Higher values of b makes it harder for term occurrences to weigh high.

3.4 Word Embedding

Word embedding is the representation of words in the form of real-valued vectors in multi-dimensional space. Although word embedding has been around for a while, semantic word embedding is a rather contemporary method. Semantic word embedding methods encode the meaning of the word in a way that the words that are similar in meaning are closer to each other in the vector space. Some of the popular techniques of producing word embedding are, dimensionality reduction, neural networks etc. Figure 1 shows an example of a 300 dimensional word embedding produced with the a pretrained fastText embedding model projected onto two dimensional plane using tSNE [47]. In the figure we can observe that words with similar context such as 'atmospheric' and 'tropospheric' are closer together and words such as 'resolution' and 'instrument' that have no contextual similarity are quite far apart.

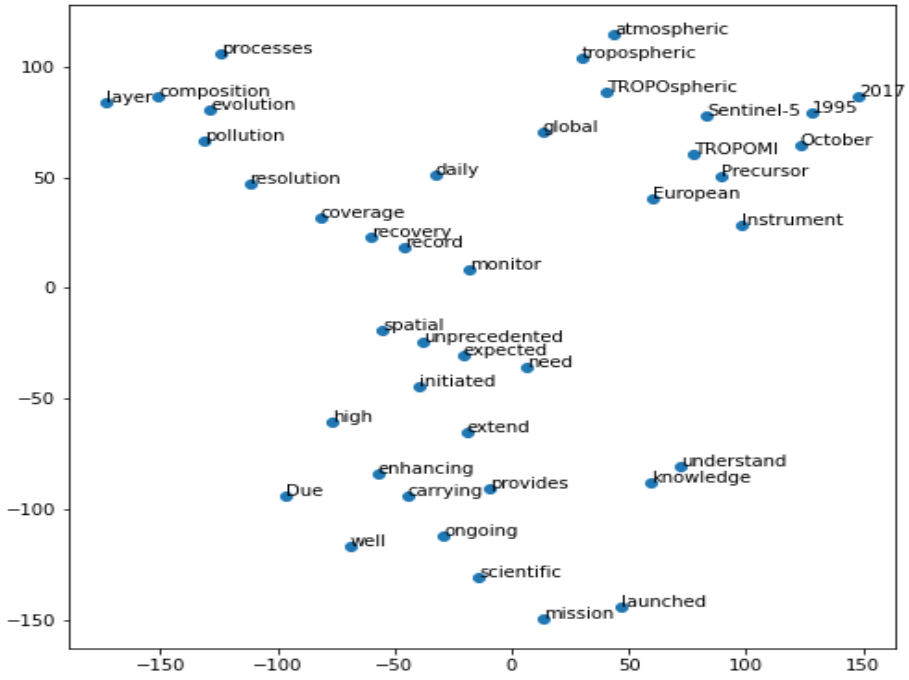


Figure 1: Word embedding visualization

Through word embedding human languages are made compre-

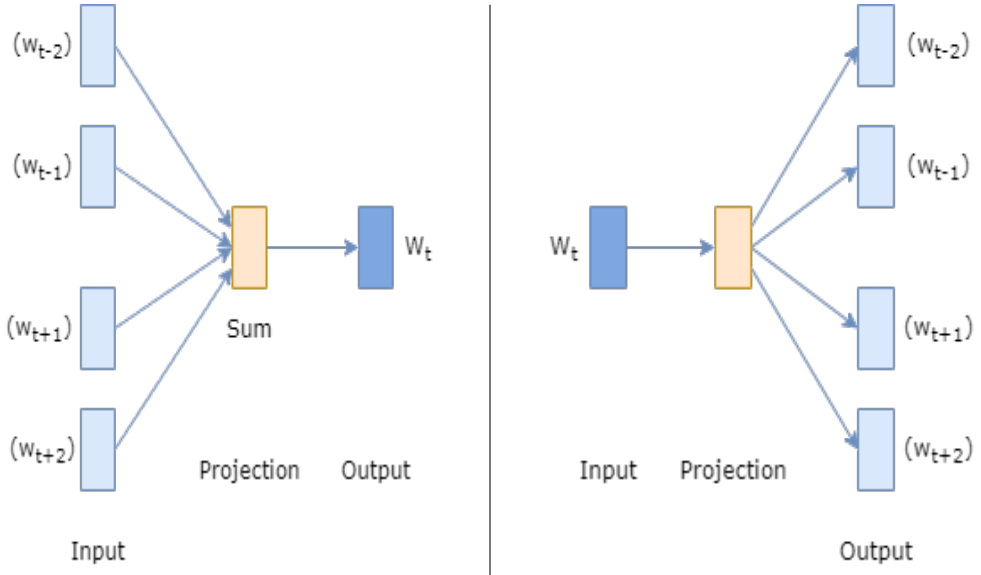
hensible to computers. Collobert et al. were the first to show the utility of pre-trained word embedding models in downstream tasks [9]. At present, for many NLP tasks such as text translation, recommendation systems and text similarity calculation, word embedding is a requirement in order to process the data further. Word embedding has enabled machine learning and deep learning models to employ vectors which are more efficient for neural networks to perceive rather than direct textual data. Here we discuss about three of the most prominent and recent pre-trained word embedding models available.

3.4.1 Word2Vec

Word2Vec learns distributed word associations using neural networks from a large text corpus. For a long time words have been regarded as atomic units without any notion of similarity amongst them, because it allowed simplicity and that simple models can perform well with bigger data corpus. However, for many NLP tasks such as automatic speech recognition, larger corpus is simply not available. Milkov et. al introduced two simple log-linear models for calculating continuous vector representations [37] which is the basis of Word2Vec. The model was developed with the idea that contextually similar words have similar meaning and should therefore have similar vector representations [20]. Word2Vec eliminates the need of requiring large annotated dataset for NLP tasks as it is capable of mapping a target word to its context words without the need of labeling. It uses a (shallow) neural net to train a mapping from unique words of a corpus vocabulary to an N-dimensional real vector space. Word2Vec exists in two versions depending on the training objective: Continuous Bag Of Words (CBOW) and Skip Gram (Figure 2). CBOW is faster and has slightly better performance on words that appear frequently while Skip Gram works better with infrequent words compared to CBOW [18].

3.4.1.1 Common Bag of Words In CBOW method, each target word is predicted by taking into account the surrounded context words. It is essentially a feed-forward neural network consisting of three layers, input layer, projection layer and output layer. The output is not influenced by the order of the context words. If we consider a simple sentence, *Apples, eggs and milk are good for health*, with a context window of size 2, the labeled training data will consist the following pairs ('context word', 'target word') : *([Apples, and, milk], eggs)*, *([Apples, eggs, milk, are], and)*, *([eggs, and, are, good], milk)* and so on. The model will then try to predict the target word from the context words. Figure 2a shows a primitive architecture of CBOW model.

3.4.1.2 Skip Gram Skip Gram has a neural network architecture similar to CBOW. The main difference is skip gram uses the current word to predict the surrounding context words within the context window. Considering the same sample sentence as CBOW, *Apples, eggs and milk are good for health* and a context window of size 2, for the target word *milk* its neighboring words are *eggs, and, are, good*. The input and target word pairs would be *(eggs, milk), (and, milk), (are, milk), (good, milk)*. Since the proximity of context words to the target word does not have any effect, all four context words will have same priority. Figure 2b depicts the basic Skip-Gram architecture.



(a) CBOW (b) Skip Gram

Figure 2: Neural Network architectures of Word2Vec

3.4.2 fastText

fastText is a word embedding and text classification library. The algorithm behind fastText is based on two publications from Facebook AI research teams [3, 28]. fastText is a significant improvement over Word2Vec in that it takes the morphological structure of words into account.Word morphology is significant when a single word has a large number of morphological forms, each of which might occur rarely. Word2Vec on the other hand produces embedding for each unique work without considering their morphology. fastText resolves the morphology issue by regarding each word as an aggregation of its sub-words, which are treated as the character n-grams of the base word. Then the word's vector is the sum of all vectors of its component characters' n-grams. This process allows fastText to support out-of-vocabulary words. The process of sub-word generation and skip gram with negative sampling is explained in the following sections.

3.4.2.1 Sub-word Generation If we consider *Apple* as an example word, character n-grams of length 3 is generated by sliding a window of 3 characters from the start to the end of the word. The resulting n-grams are <*Ap, app, ppl, ple, le*>, where the angular brackets denote beginning and ending of a single word. Producing character n-grams is useful in order for the model to understand prefixes, suffixes and shorter words within a word. Hashing is used to bound the memory requirements. fastText learns total 'b' embeddings where 'b' denotes the bucket size, which represents the array size allocated for all the character n-grams. The original paper used a bucket of a size of 2 million [3].

3.4.2.2 Skip Gram with Negative Sampling Negative sampling refers to using Sigmoid function to learn the differentiation between actual context words (positive) and false context words (negative) sampled from a noise distribution. Hence, words vectors are not needed to be learned by predicting context words of a target word. For example, in the sentence *The apple is red*, if the target word is *apple* and the context window is 1, the context words are, *The, is*. Embedding for the target word is calculated by taking a sum of vectors for the character n-grams and the whole word itself. For the context words, their word vector are taken from the embedding table without adding the character n-grams. Negative samples are drawn randomly with probability proportioned to the square root of the uni-gram frequency. For each positive context word, k random negative words are sampled where k is a hyper-parameter and can be tuned empirically. Sigmoid function is applie don the dot product between the target word and positive context words. Based on the loss, embedding vectors are updated with Stochastic Gradient Descent optimizer to bring actual context words (Sigmoid output close to 1) closer to the center word but increase distance to the negative samples (Sigmoid output close to 0). Figure 3 shows the process of producing embedding vectors where the negative context word sam-

ples are *Moon, Air*.

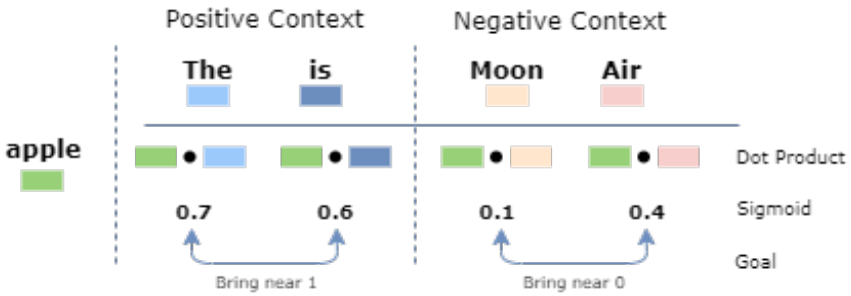


Figure 3: Skip Gram with Negative Sampling

3.4.3 ConceptNet Numberbatch

ConceptNet is a semantic network that provides different computations with word meaning including word embedding [46]. ConceptNet itself is a knowledge graph that connects natural language entities with weighted and labeled edges. In addition to its own knowledge about words, ConceptNet also represents links between various knowledge resources such as WordNet, Wiktionary, OpenCyc, and DBPedia.

A retrofitting approach is applied to combine ConceptNet with distributed semantically produced word embeddings (Word2Vec) performs better than distributional semantics alone. Numberbatch has better performance because it benefits from the semi-structured common sense knowledge from ConceptNet. Numberbatch is a combination of ConceptNet, Word2Vec, GloVe and OpenSubtitles [46]. The ConceptNet Numberbatch outperforms pre-trained word embedding models such as Word2Vec, GloVe, fastText on word similarity tasks.

3.5 Transformer Models

The Transformer architecture was first proposed in the paper *Attention Is All You Need* [48]. The transformer network is capable of solving sequence-to-sequence tasks with easier handling of long-range dependencies compared to Recurrent Neural Networks.

Python's Hugging Face library provides a selection of pre-trained models that have transformer networks in the core and are capable of various Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks [51]. In this project we employ some of these pre-trained models for zero-shot classification. In chapter subsection 4.7 we explain the process of *how* we have used the models for document retrieval. In this subsection, we first discuss the autoencoder and transformer architectures as part of preambles and move on to explain details of the specific pre-trained transformer models we have selected for our project.

3.5.1 Autoencoder

Autoencoder is a type of neural network that is able to encode and compress data in an unsupervised way then learns to reconstruct the input data from the compressed version with minimum error. The idea of Autoencoder has been around for many years [31, 4, 23]. In 2016, Ian Goodfellow comprehensively explained the basic structure and different variations of autoencoders in his book[17].

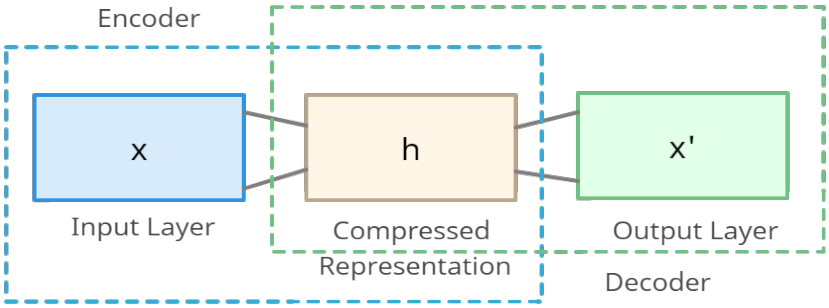


Figure 4: Simple Autoencoder

An autoencoder has two parts, the encoder side and the decoder side. Encoder function ($f(x)$) encodes input x into compressed representation h. Afterwards, decoder function $g(h)$ decodes h to a close representation (x') of input x. An autoencoder is intentionally re-

stricted from learning to copy input to output by adding 'noise' to the data which enables them to copy only approximately, resulting in the model learning important features of the input data.

3.5.2 Transformer

Transformers were proposed as a neural network model that is capable of processing sequential data (eg: natural language) for NLP tasks. However, transformer models employ attention mechanism to weigh the importance of each part of the input sequence. Whereas its predecessors such as Recurrent Neural Networks, long short-term memory [24] and gated recurrent neural networks [8] took into account the symbol position of input and output data. This hinders parallelization and becomes critical for processing longer sequences. Since, transformer takes into account the context of each word in a sequence, it does not need to maintain order which in turn reduces training time by allowing parallelization [48].

3.5.2.1 Encoder Stack Figure 5 shows the original transformer diagram depicted by the authors. The left portion of the diagram shows the encoder stack which consists of 6 identical layers. Each layer consists of two major components: a self-attention mechanism (explained in a later paragraph) and a feed-forward neural network. The sub layers have a residual connection around them followed by layer normalization. The output encodings are finally passed to the next encoder stack as its input, as well as the decoders. The first encoder stack takes positional information and embeddings of the input sequence as its input instead of encodings.

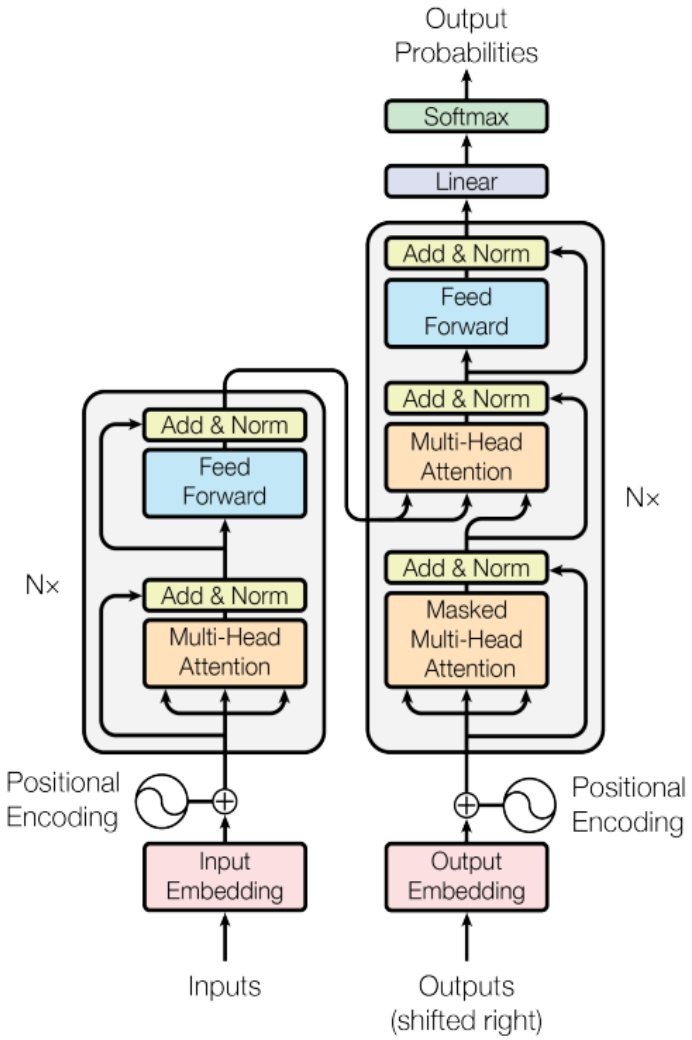


Figure 5: Transformer Architecture [48]

3.5.2.2 Decoder Stack In Figure 5 the right side shows the decoder stack. Similar to the encoder side, it also has 6 identical layers. Each layer consists of the following three sub-layers, the self-attention mechanism, a feed-forward neural network and a multi-head attention mechanism that applies multi head attention on the output received from encoder stack. Similar to the encoder stack, The sub layers have a residual connection around them followed by layer normalization. The self-attention sub-layer also encompasses a masking scheme along with the output embeddings being shifted right by one position to ensure that the prediction for a certain posi-

tion depends only on the knowledge of outputs from previous positions.

3.5.2.3 Attention Mechanism The authors of the original transformer paper have described attention mechanism as such, *An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by compatibility function of the query with the corresponding key.* [48].

The transformer architecture has two different types of attention mechanism, scaled dot product attention and multi head attention. The input for the scaled dot product attention has 3 components, queries and keys (dim d_k), and values (dim d_v). The dot product is then calculated by dividing the query by all the keys and diving the result by $\sqrt{d_k}$, afterwards softmax function is applied to get the final weights of the values.

For the case of multi head attention, the authors argued that is is beneficial to project the queries (dim d_k), keys (dim d_k) and values (dim d_v) linearly h times with different, learned projection of dimensions which is also linear. Afterwards, attention function can be applied in parallel to these components to get output values of dim d_v. Which are then concatenated and projected to get the final values. With multi head attention it is possible to process information from different subspaces at different postions in parallel.

3.5.3 Transformer Based Pre-trained NLP Models

The python Hugging Face library has a large number of pre-trained models based on transformer architecture that are capable of various downstream NLP tasks. For our project we have selected 7 of these models and employed them for document retrieval task. In this section we briefly discuss the architecture and training of these

models.

3.5.3.1 BERT BERT (Bidirectional Encoder Representations from Transformers) is a popular transformer based context aware language representation model used to solve several language processing tasks. Opposed to directional models, which reads the text input sequentially (left-to-right or right-to-left), BERT encoder reads the entire sequence of words at once. It is an advantage for tasks where information about later tokens are available while processing earlier tokens.

The multi-layer bidirectional model has the original transformer network at its core. It pre-trains deep bidirectional models from unlabeled input text by accounting both left and right context in all layers [11]. It was pre-trained with the following objectives, Masked Language Model and Next Sentence Prediction.

- Masked Language Model (MLM): The general assumption in the context of sequential models is that a bidirectional model would perform better than either left-to-right or right-to-left as well as simple integration of both. However, the bidirectional models poses the problem that a specific token of the input would have the context of itself and would be able to predict the target word with this knowledge. To avoid this issue, in the MLM issue some percentage of the input is masked during training [49]. In order to avoid conflict between pre-training and fine-tuning the authors of BERT chose 15% token positions at random and amongst these tokens 80% of the time are replaced with a mask, 10% of the time is replaced with a random token and 10% of the time remain unchanged.

- Next Sentence Prediction (NSP): NSP is the basis of many other NLP tasks such as language inference, question answering and so on. BERT was trained for binary next sentence prediction task in order to make it capable of doing sentence prediction

tasks. In the training data, half of the time the target sentence of a sentence pair is actually the sentence trailing the initial sentence and rest of the time it is replaced with a random sentence. The authors of BERT show that training towards NSP is helpful for sentence prediction tasks [11].

The pre-trained model can be easily fine-tuned by adding one extra output layer to produce models for several downstream NLP tasks such as text classification, question answering etc. In our task we use the Hugging Face *bert-base-uncased* model which has 12 - transformer blocks, 768 - hidden size, 12 - self-attention heads, $110M$ parameters and is trained on lower-cased English text [11].

3.5.3.2 RoBERTa RoBERTa (Robustly Optimized BERT Pretraining Approach) [35] as the name suggests is a study on the impact of hyperparameter choices on the original BERT model [11]. The authors of RoBERTa show that the original BERT model was undertrained and the improved version (RoBERTa) with slightly different design choices exceeds initial performance. Modifications on the original BERT architecture are as follows:

- Exclusion of the Next Sentence Prediction (NSP) objective.
- Increase of batch size, data and training time.
- Increase of sequence length.
- Dynamic change of the input text masking pattern.

The mentioned changes led to improved performance on downstream tasks and the new model (RoBERTa) was proven to be competitive with previously published similar approaches. In our project we use the *RoBERTa-large-mnli* version of the model which is fine-tuned on the MultiNLI dataset [10]. It has 24 transformer layers, 1024 hidden size, 16 attention heads and $355M$ parameters

3.5.3.3 DeBERTa DeBERTa [22] (Decoding-enhanced BERT with disentangled attention) uses disentangled attention mechanism and an enhanced masked decoder to improve the original BERT [11] and RoBERTa [35] models. In addition, fine-tuning was done using a virtual adversarial training method to improve model generalization. Here we briefly explain the disentangled attention mechanism and the enhanced masked encoder approach.

- Disentangled Attention Mechanism: In BERT encoder architecture, the input word vector is calculated as sum of the word's embedding and its position embedding. However, in DeBERTa model, each input word is represented using two separate vectors, one for the word content embedding and one for the word position embedding. For each word the attentions weights are calculated using disentangled metrics from their content and position. Therefore, the attention weight of word pairs get equal contribution from both content and relative position of each word.

- Enhanced Mask Encoder: The enhanced mask encoder is incorporated in the architecture to account for the absolute position of a word in a sentence. Using disentangled attention mechanism context and relative position of a word is considered however, syntactic roles of a word is more dependent on the absolute position of the word in a sentence. Enhanced mask encoder adds absolute word position embedding before the Softmax layer, from there the decoder predicts the target words based on the context and position embedding.

According to the authors the enhanced DeBERTa model perform better on downstream NLG and NLU tasks compared to original BERT or RoBERTa model [22](refer to original paper for detailed results). In our experiment we use the *deberta-base* model has 12 transformer layers and attention heads, hidden size 768 and approximately $140M$ parameters.

3.5.3.4 SqueezeBERT SqueezeBERT [25] is an improved version of the BERT [11] architecture that replaces computationally expensive self attention layers with grouped convolutions. The resulting model runs 4.3x faster than the original BERT model.

Inside the BERT encoder blocks, there is a self attention layer which accommodates multiple separate position -wise-fully-connected (PFC) layers. PFC layers generate the position wise activation vectors for feature embedding. In the paper, the authors of SqueezeBERT point out that upto 88.3% of the network latency. Furthermore, they observe that the Fully Connected layers in the BERT encoder can be considered as special case of non-grouped 1D convolution and the PFC layers' operation is same as a $kernelsize = 1$ convolution. Hence, the networks behavior and numerical features remain same while significantly decreasing run time.

3.5.3.5 BART The BART (Bidirectional and Auto-Regressive Transformer) proposed in 2019 is a denoising autoencoder model with standard transformer network at its core [33]. BART performs well on sentence generation as well as text comprehension tasks however, it can also be used for sentence classification the in the same way as other pre-trained transformer models. In the original paper, the authors have pointed out that, amongst many noising strategies, both shuffling the original sentence in random order and a infilling scheme (arbitrary spans of sentence masked with single mask token) work best. The transformer based machine translation model leverages BERT [48] and GPT (Generative Pre-trained Transformer) [40] models for training.

The BART auto-regressive model uses BERT for encoding to benefit from its bidirectional encoder. However, the authors of BART pointed out that, BERT is not a good model for sentence generation as it predicts masked tokens randomly.

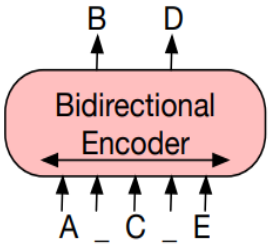


Figure 6: BERT Encoder [33]

On the other hand GPT has a left-to-right decoder and predicts tokens auto-regressively, making it useful for text generation. The drawback here is that GPT cannot learn bidirectional interactions because tokens have only leftward context.

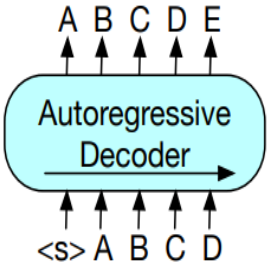


Figure 7: GPT Decoder [33]

The encoder input and decoder output are not required to be aligned which allows for random noise transformations. In the BART model, input sentence is corrupted by adding mask symbols as replacement for text spans and then it is fed to the bidirectional encoder (BERT). The target output is predicted by an auto-regressive decoder (GPT) from previous un-altered tokens and encoder output.

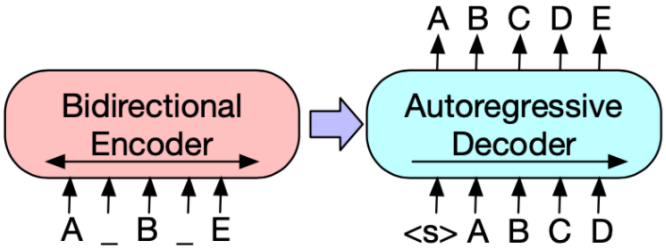


Figure 8: BART Architecture [33]

In Figure 8, original input is *'A B C D E'*, the span *C, D* is masked, at the same time an extra masked token is inserted before *B*. Hence, the input of the encoder becomes *A _ B _ E*.

In our experiment we use the *bert-large-mnli* [33] model, a BART-large base architecture that has been fine=tuned on the Multi-Genre Natural Language Inference (MultiNLI) dataset [50]. The model has an additional 2 layer classification head and $1M$ parameters.

3.5.3.6 BART-Yahoo In this version of BART, the *bert-large-mnli* [33] model has been further finetuned on the Yahoo Answers topic classification dataset [10]. Since the model has been fine-tuned on topic classification dataset, it presumably performs well on zero-shot classification tasks. Hence, we chose the model as one of our pre-trained transformer models for our document retrieval task.

3.5.3.7 DistilBART DistilBART [38] is the distilled or student version of the *bert-large-mnli* [33] model that used the *No Teacher Distillation* [45] approach. No Teacher Distillation is a 'shrink and fine-tune' (SFT) technique which is simply reducing the teacher model to a student model by copying parameters.

As an example, the authors described that, for a student BART with 3 layers of decoder, they copy full decoder layers, $0, 6 and 11$ from the original BART model. The encoder layers are copied entirely without exclusion. They also argue that, during ties, any of the layers can be arbitrarily chosen meaning instead of copying layer 6, layer 5 would work equally well. When only a single decoder layer is needed for the student mode, layer 1 is chosen.

According to the DistilBART paper, the student model performs better when initialized with Direct Knowledge Distillation (KD) as the student model tries to match the probability distribution of the teacher model over next target words by reducing KL-divergence [30, 43].

We chose the pre-trained *distilbart-mnli-12-3* model along with

the *bart-large-mnli* in order to observe the performance of a smaller version of *bart-large* model on downstream zero-shot classification task.

3.6 Dense Passage Retrieval

DPR or Dense Passage Retrieval is a retrieval method implemented using only dense representation where a simple dual encoder is used for the embedding of passages and questions to be learned [29]. The authors argue that the dense, latent semantic encoding based retriever is able to capture useful context of a text that is not possible using a inverted index based retrieval approach. DPR uses the BERT [11] pre-trained model along with a dual-encoder architecture [5].

The model is trained on mini-batches question and passage (answers) pairs. The authors observe that optimizing the embedding for maximizing inner products of question and relevant passage vectors, with the objective of comparing all question-passage pairs in a single batch gives better results. Additionally, they conclude that by only fine-tuning input text encoders on existing QA pairs can significantly outperform the standard BM25 model [41].

DPR uses two BERT models for encoding, one for *passage vector* encoding from the passages (answers) and the other one encodes questions into a *question vector*. Later during inference, the passage encoder is applied to all the passages and indexed using FAISS [26] which is a library for clustering dense vectors and similarity search tasks. In our project we use the Haystack [21] implementation of DPR using FAISS.

4 Methodology

4.1 Dataset

This thesis strives to enhance document retrieval for finding thematically relevant documents (where themes are expressed using keywords or some other form of query. The datasets used in this project are unstructured and diverse in content. However, all these datasets contain parts of text from scientific publications that are in some way mapped towards keywords or labels. This allows us to perform similar experiments on the datasets and evaluate the results using same metrics. There are in total 3 primary datasets. We have selected the datasets based on their relevance to the task, each of these datsets provide some advantages over other datasets.

4.2 Dataset Description

Here we discuss the source and content of our primary datasets.

- **ENX Dataset** : This dataset has been scraped from the engineering pre-print server engrXiv [14]. The corpus contains 1237 documents from various engineering backgrounds such as 'Biotechnology', 'Aerospace', 'Computer Science' etc. The scraped data was processed and formatted in a way that can be easily used. Afterwards, each document has the following list of attributes, *'_key'*, *'_id'*, *'rev'*, *'id'*, *'title'*, *'authors'*, *'abstract'*, *'doi'*, *'date'*, *'disciplines'*, *'tags'*, *'type_key'*, *'origin'*, *'link'* and *'searchtext'*. The useful features for our purpose are *'_key'*, *'title'*, *'abstract'*, *'disciplines'* and *'searchtext'* (combination of 'title' and 'abstract'). The disciplines attribute contains the thematic labels which can be used for measuring the success of retrieval methods. A relational data-frame has been produced mapping each discipline to a list of corresponding documents (*example: Chemical = Chemical Engineering + Chemical Kinetics + Other*

Chemical Engineering.)

The raw dataset has 113 unique disciplines which were later reduced to 80 by combining similar disciplines.

- **ESR Dataset** : ESR or Explicit Semantic Ranking dataset was curated for the paper Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding [52]. The raw dataset contains query log, relevance judgments, candidate documents (from Semantic Scholar) as well as ranking lists. We are using the candidate documents and query log for our experiment. There are 100 different query terms mostly related to computer science such as 'Deep Learning', 'Part of speech tags', 'Convolutional neural networks' etc. and 3251 corresponding documents. Each document contains the following attributes 'doc_no', 'title' and 'paperAbstract'. To maintain uniformity 'doc_no' has been changed to '_key' and 'searchtext' has been produced combining 'title' and 'paperAbstract'. The main advantage of the ESR dataset over the ENX dataset is that, the mapping between queries and documents has been done by experts which makes it more reliable.

- **EU-ENV Dataset** : It is a corpus of 1463 documents scrapped from the Science for Environment Policy News site [16]. The articles are mainly oriented towards environmental themes such as 'Climate change', 'Biodiversity', 'chemical' etc. Each document contains *'key', 'link', 'title', 'abstract', 'text', 'date', 'theme', 'source' and 'contact'*. We only use the fields 'key', 'theme' and 'text' for our experiments. There are also lists of documents (keys) for all thematic tags which are concerned with that tag and in total there are 31 unique tags. The headers 'key', 'text' and 'theme' has been changed into '_key', 'searchtext' and 'query' respectively. Similar to the ENV dataset, the tags to document mapping of EU-ENV Dataset has been done by the editors who

collected the articles for the website. At the same time each query in this dataset is mapped to a larger number of documents compared to ESR or ENX dataset, which makes performing some of our experiments easier (explained in chapter x).

- **Others** : We have used a corpus of scraped text from the ELIB publications of DLR [13] for some preliminary evaluations of relevance ranking models (further explanation on chapter x). The corpus contains 'abstracts' from nearly 18000 documents published on the DLR Electronic Library (ELIB).

4.3 Data Pre-processing

Data pre-processing is a crucial part of data centric projects. The goal of pre-processing step is to bring uniformity amongst different datasets as well as reduce redundancy from the raw data. Textual data pre-processing is a fairly straight forward process and has a well established set of techniques. We have followed the methods adapted by the authors of these Topic Discovery papers [19] with some changes.

Figure 9: Text pre-processing pipeline

Figure 9 illustrates the steps followed to prepare the raw data for experiments. Below are the explanations of the process:

- Tokenization : Raw text (concatenation of several sentences) is lower-cased and punctuation marks are removed. Then the strings are tokenized. We have used custom SpaCy tokenizer in order to not remove 'hyphens'. Our experiments show that word embedding models like fastText work better with hyphenated

words (eg. non-standard) rather than pair of words (eg. non standard).

- Non-ASCII Strings Removal : We remove all special characters as well as non-English letters.
- Stop-words Removal : We have used the default stop-word list from available with SpaCy. It has the largest (326 words) list of stop-words in comparison with similar text processing libraries.
- Short Words Removal : We remove all words that are less than 2 characters long.
- Lemmatization : Lemmatization is the technique to convert words to its contextually relevant base form. Hence same words can have different lemma depending on the context they have in a sentence. We removed redundancy by grouping different inflected forms of words into their base form using lemmatization.
- Re-join Tokens : Since the transformer models accept sentence as input, in our processed datasets we keep the text input as a single space separated string of tokens retrieved after the pre-processing steps.

Here is a sample raw text along with the processed version.

Raw Text : Semantic parsing has been a topic of great interest to researchers for a very long time. The drawback of these approaches was that the systems were specific domain dependant [3]. Some improved methods that take into consideration semantic aspects instead of only syntactic information are proposed in some publications [5, 6, 7]. In recent years neural sequence-to-sequence models have been proposed to deal with semantic parsing [1,8]. These systems have overcome the need for extensive feature engineering. Some sophisticated ideas to improve the outcome of these models are data augmentation [9] and transfer learning [10].

Processed text: semantic parse topic great interest researcher long time drawback approach system specific domain dependant improve method consideration semantic aspect instead syntactic information propose publication recent year neural sequence-to-sequence model propose deal semantic parsing system overcome need extensive feature engineering sophisticated idea improve outcome model data augmentation transfer learning

4.4 Similarity Score Calculation

In our proposed similarity based relevance ranking and logistic regression methods we require the similarity scores between two terms. In our case, the first term is individual words from documents and the second term is the keyword we use to retrieve documents. The similarity score is calculated as cosine similarity between the embedding vectors of the two terms. The idea behind the approach is that, word that are closer in the vector space are expected to be similar in meaning and have higher similarity score. Euclidean distance metric accounts the magnitude of the vectors as well as distance. Similarity score between two word vectors will then depend on their length and frequency along with their orientation. However, angular distance metrics such as cosine similarity depends only on the orientation of the words and provides more accurate semantic similarity scores between common and uncommon, frequent and less frequent words.

- **Cosine Score**: It is the cosine of the angle between two vectors, that gives the angular distance between the vectors. The formula to calculate the cosine distance of two vectors, A and B is as follows,

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3}$$

  Cosine is 1 at $\theta = 0$ and -1 at $\theta = 180$, hence for exactly same

words the similarity score is highest and for opposite words having opposite vectors it will be lowest.

In this thesis we use 4 different word embedding models to calculate the word vectors. They are as follows:

- Pre-trained fastText model ft-en-cc [15].
- fastText model trained on some data from the ELIB repository of DLR. We trained different embedding models on this data with different parameter values and chose the elib_model_s_100_e_5_w_5 model, denoted as , it is on 100 dimension with a window size of 5 and epoch 5.
- ConceptNet Numberbatch embedding model version one. Since the model is not capable of handling out-of-vocabulary words, we do a gradual removal of suffix characters and search for vectors for out-of-vocabulary words until a single character is left. The model is denoted as nb_vec henceforth.
- ConceptNet Numberbatch embedding model version two. Instead of gradual removal of suffix characters when out-of-vocabulary words are encountered, we instead use ft-en-cc to get the embedding vector of the out-of-vocabulary words. The model is denoted as nb_vec_ft henceforth.

4.5 Similarity Score Based Relevance Ranking

In this section, we explain our proposed novel relevance ranking method for term based retrieval of documents that are contextually related to the term regardless of the existence or non-existence of the term in the document. At its core the relevance ranking method has a complex relevance ranking equation which returns the ranking of a document corresponding to a specific term. Although, the equation is inspired from the Okapi-BM25 ranking approach, it has

many differing components. In the next part we have explained the relevance ranking equation along with its different components.

$$R(t) = KC(t) \sum_i G(w_i, t) F(w_i) W(w_i) \tag{4}$$

Here,

- w_i indicates a specific word in a document and t indicates the search term in consideration.
- K is an overall normalization constant which is chosen in order to keep the maximum of $R(t)$ at 1.
- $G(w, t)$ is based on the embedding cosine similarity of w and t ($sim(w, t)$).
  - $G(w, t)$ is 0 for word pairs having similarity scores below a certain threshold (this threshold is denoted as st henceforth), 1 for identical words and between 0 – 1 for some semantic relevance between the words.
  - We chose the following equation for $G(w, t)$ where θ is a threshold.

$$G(w, t) = \frac{q(sim(w, t) - \theta^2)}{(1 - \theta)^2} + \frac{1}{q} \tag{5}$$

- $W(w)$ is a function that is related to the frequency of w in the document.
  - We use the following expression for $w(w)$,

$$W(w) = \frac{x(k_w + 1)}{x + k_w} \tag{6}$$

    and

$$x = \frac{count_i}{(1 - b_w) + b_w * a_w} \tag{7}$$

  - $W(w)$ integrates length normalization, b_w and saturation, k_w.

- a_w is the total number of words in document i divided by the average number of words in all documents of the set.

- $F(w)$ is a function that measures the significance of w within the document.
  - We have used the posIDfRank method proposed by Hamm et al. to calculate these values [19].
  - The method combines the following: voting based on local word neighborhood associations; a weighting according to the absolute position in the text; counter balancing the influence of unspecific words by the inverse document frequency.

- $C(t)$ rewards situations where words that are semantically somewhat related to t cover a big portion of the document.
  - We use the following expression for $C(t)$,

$$C(t) = \frac{x(k_c + 1)}{x + k_c} \tag{8}$$

    and

$$x = \frac{count_i}{(1 - b_c) + b_c * a_c} \tag{9}$$

  - $C(t)$ integrates length normalization, b_c and saturation, k_c.
  - a_c is the number of unique words in document i divided by the average number of unique words in all documents of the set.

In total there are 7 parameters ($\theta, q, b_w, k_w, b_c, k_c$ and st) can be used for fitting to the labeled datasets.) in the relevance ranking equation that can has been tuned to successfully rank the relevance of a document against a particular term. After many experiments we set $q = \frac{3}{4}, b_w = 0.1, k_w = 2, b_c = 0.4$ and $k_c = 2$. In subsubsection 5.2.5 we show how different values of θ and st influence the relevance ranking score.

4.6 Similarity Score Based Logistic Regression

It is possible to use the similarity scores between the words in a document and keywords as features for a regression model to predict class probability. In our task, for each dataset, we gather similarity scores each word in a document with individual keywords. We have then processed these scores into 10 equidistant weighted histogram bins based on value. For the implementation of the weighted bins, whenever a word has a similarity with the keyword within the bin, instead of adding 1, we add the PIdfrank value of the word. In this way, words have more impact on the count if they are important words. Normalisation is done in a way that the weighted bin counts add up to 1 for each document. For each embedding model we then have a single dataframe containing 10 features apart from document key and label. For each dataset, apart from the 4 original dataframes from the 4 embedding models, we also create 4 other combinations of these dataframes by horizontally concatenating dataframes. Lastly, we apply logistic regression on the data to predict the document class. We use a 70 – 30 train-test split for the data. The 4 dataframes that are produced in combination of different dataframes are as follows:

- **ft_elib** : ft_en_cc + elib_model_s_100_e_5_w_5
- **ft_nb** : ft_en_cc + nb_vec
- **ft_nb_ft** : ft_en_cc + nb_vec_ft
- **ft_nb_elib** : ft_en_cc + nb_vec + elib_model_s_100_e_5_w_5

4.7 Zero-shot Learning with Transformer based pretarined NLP Models

Zero-shot learning, in the context of our task, is a classification strategy where the task is to predict classes that had not been included in the training. Natural Language Inference (NLI) is one of the

possible strategies for zero-shot learning. Example: The premise 'It is snowing' entails the hypothesis 'The temperature is below 10°C'. The premise 'Today is the 29th of February' contradicts the hypothesis 'It is the year 2021'. In the examples the relation between the sentence pairs can be inferred from the context. In subsection 3.5 we have explained the core Transformer architecture and discussed about the transformer based NLP models we use in this thesis.

The pre-trained transformer models available on the Hugging Face library are trained with many different objectives, some of these models can be used for downstream classification tasks. We employ the Hugging Face classifier pipeline for this purpose. The components of the classifier pipeline are:

- A language model (eg: bert-base-uncased) and a tokenizer.
- The text in to be classified, termed as premise. For our case this is the content of the document.
- A list of hypothesis, possible classes for the text. We use the list of keywords respective to each dataset as hypothesis.
- A hypothesis template, the default template is : this example is .
- A Boolean parameter, when set it allows to compute entailment probability score of a document for multiple possible classes.

The classifier pipeline returns the entailment probability of each class from the hypothesis, we choose the class with the highest probability as the predicted class for the document.

subsubsection 3.5.3 contains the list of pre-trained NLP models we use for the zero-shot learning task. Along with the pre-trained models, we have also fine-tuned 2 other models. We have fine-tuned the *bert-base-uncased* model with the help of the Trainer class from the Hugging Face transformers module.

- **BERT-ENX** : We considered 50% of the ENX data and did a 70 – 30 train-test split on it to fine-tune the *bert-base-uncased* model. Later, we make a subset corresponding to the keywords we used for ENX from the unused 50% data and use that during testing.

- **BERT-ESR** : We considered 30% of the ESR data and did a 70 – 30 train-test split on it to fine-tune the *bert-base-uncased* model. Similar to BERT-ENX, we make a subset corresponding to the keywords we used for ESR from the unused 70% data and use that during testing.

The Trainer Class requires a specially formatted NLI dataset. Therefore, we converted the data subsets used for fine-tuning accordingly. We set the document content as 'premise' and convert the keywords from single words to full sentences (eg. This is a document about chemicals.) to be the 'hypothesis'. Lastly, for each document only its true hypothesis is labeled as entailment and rest of the four hypothesis are labeled as contradiction.

5 Experiments

5.1 Evaluation Metrics

All the documents from the three datasets are mapped to a certain keyword or in other words have pre-assigned class label. In our different experiments, we essentially try to predict a class label for the documents. Therefore, evaluation metrics for classification tasks has been adapted as the evaluation metric. In this thesis, we use the standard metrics, precision, recall and F1 score. We have also used weighted average precision and weighted average recall to evaluate some results. Since, some of our data sub-classes are imbalanced, we do not use accuracy as it will not be an accurate reflection of

the results. Before we proceed with the explanation of the evaluation metrics, we describe the confusion metrics terminology (see Figure 10).

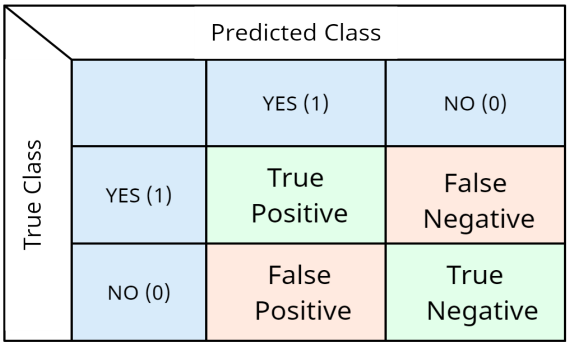


Figure 10: Confusion Matrix

- True Positive (TP) : For a specific class (keyword), the true label of a document and the predicted label of the document both are positive (1), successful prediction.
- True Negative (TN) : For a specific class (keyword), the true label of a document and the predicted label of the document both are negative (0), successful prediction.
- False Positive (FP) : For a specific class (keyword), the true label of a document is negative (0) but the predicted label is positive (1), false prediction.
- False Negative (FN) : For a specific class (keyword), the true label of a document is positive (1) but the predicted class is negative (0), false prediction.

We can now define our evaluation metrics using these terminologies.

- **Precision :** For a specific keyword, it is the ratio between correctly predicted observations and all predicted observations.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

- **Recall :** For a specific keyword, recall calculates the ratio between correctly predicted observations and all true observations. Recall is also known as sensitivity.

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

- **F1 Score :** It is a combination (harmonic mean) of *Precision* and *Recall* that takes into account both false positive and false negative. F1 Score is a good metric for imbalanced classes.

$$F1\_Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{12}$$

- **Weighted Average Precision :** In our experiments we have used a set of keywords, we compile results using only one of these keywords at a time. We can evaluate that result using standard *Precision* and *Recall*. However, we also try to assess the quality of the accumulated results on all keywords. The *Weighted Average Precision* is the method we use to achieve that. At first precision is calculated for each query separately then we calculate the weighted average, weights are proportional to the total number of documents related to the queries being used.

$$Precision_{wa} = \frac{\sum_q D_q * P_q}{\sum_q D_q} \tag{13}$$

  Here, D_q is the number of documents mapped to query q and P_d is the *Precision* calculated for query q.

- **Weighted Average Recall :** We calculate *Weighted Average Recall* in a similar manner as *Weighted Average Precision*. At first recall is calculated for each query separately then we calculate the weighted average, weights are proportional to the total number of documents related to the queries being used.

$$Recall_{wa} = \frac{\sum_q D_q * R_q}{\sum_q D_q} \tag{14}$$

Here, D_q is the number of documents mapped to query q and R_d is the *Recall* calculated for query q.

-

Weighted Average F1 Score : We calculate the weighted average F1 score with the same formula as standard F1 score with weighted average precision and recall values.

$$F1\_Score_{wa} = \frac{2 * (Precision_{wa} * Recall_{wa})}{Precision_{wa} + Recall_{wa}} \tag{15}$$

5.2 Model Setup and Results

For our experiments, we chose 3 sets of 5 keywords corresponding to each of our datasets. These are the queries we use to evaluate our models along with the benchmarks. Table 1 shows the list of keywords used for each dataset.

Table 1: List of keywords for each dataset.

Dataset	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$	$\mathbf{q_4}$	$\mathbf{q_5}$
ENX	biomedical	combustion	computer	aerospace	chemical
ESR	deep learning	question answering	computer vision	information geometry	cryptography
ENV	energy	biodiversity	soil	agriculture	chemicals

In the following subsections we discuss the approach and results of the different experiments that have been conducted.

5.2.1 Keyword Search

Keyword search refers to explicitly looking for a keyword in a set of documents. A certain document or text is retrieved only if the query term exists in the document. Keyword search in our experiment has been implemented by simply using Python's *'in'* command.

We calculate precision, recall and f1-score on each individual keywords and use the values to subsequently calculate the weighted average precision, recall and f1-score.

Table 2: Keyword Search Evaluation

Dataset	**precision$_{wa}$**	**recall$_{wa}$**	**f1-score$_{wa}$**
ENX	0.73	0.14	0.23
ESR	0.96	0.68	0.79
ENV	0.72	0.59	0.65

Table 2 contains the *Weighted Average Precision* and *Weighted Average Recall*. We can observe that the *recall* score is much lower compared to *precision* score, this is due to the fact that with keyword search we only extracted documents that contained the keywords. However, there were other relevant documents in the corpus that do not explicitly contain the keyword but is contextually associated with it. Hence, we can draw the conclusion that in order to retrieve documents that are contextually associated to the keyword but do not contain the keyword explicitly explicit keyword search is not very useful.

5.2.2 Okapi-BM25

We have explained in subsection 3.3 the principles behind the Okapi-BM25 algorithm. It is an important benchmark in relevance ranking tasks related to academic and commercial research. Hence, we have produced relevance ranking results using the Okapi BM25 in order to be able to compare the results of our approaches with an established method. Similar to the *Keyword Search* approach, we chose the same 3 sets of keywords corresponding to our datasets (Table 1). For each keyword, after applying BM25, we get a descending ordered list of scores corresponding to the documents. For

each keyword, we then consider the first (N+5) (N being the actual number of documents corresponding to the keyword in the dataset) documents as a match for the keyword. Afterwards, we compare the predicted list of documents with actual list of documents using standard precision and recall as well as weighted average precision and recall to analyze the outcome. The relevance ranking algorithm has been implemented using the Python library *rank_bm25* which has the *okapi-bm25* version of the BM25 algorithms along with some other variations of BM25 [6].

Table 3: Okapi BM25 Search Evaluation (Weighted Average Precision and Recall)

Dataset	**precision$_{wa}$**	**recall$_{wa}$**	**f1-score$_{wa}$**
ENX	0.30	0.33	0.32
ESR	0.81	0.92	0.86
ENV	0.55	0.58	0.56

Table 3 shows the weighted average precision, recall and f1-score values on all datasets. Here we can see that these evaluation scores on ESR dataset is comparatively better than it is on ENX and ENV dataset. Previously in the keyword search approach(Table 2), we observed that the precision score for ESR dataset is much higher compared to the other datasets. A closer look into the dataset content shows us that ESR dataset tends to contain many direct instances of the keywords in the searchtext which is not the case for ENX and ENV dataset. At the same time, the documents of ESR dataset do not have overlapping contextual similarities amongst them. Since the documents in ENX dataset match contextually to the keywords and in most cases do not actually contain the keyword itself, it is not possible to efficiently retrieve documents from such a corpus using Okapi-BM25. Another reason for the ESR scores being high is, BM25 searches documents individually for each sub-word of a

multi-word query (eg. deep learning) and combines the separate results for the final rank, making it easier for BM25 to identify documents that contain both sub-words together or separately (possibly out of context).

5.2.3 Dense Passage Retrieval

In subsection 3.6 we briefly explained the motivation and architecture of the dense passage retrieval method. The approach was proposed for open domain question answering. Therefore, logically it can be used for document retrieval in the sense that, we can think of our unlabeled document corpus as the data source and our queries as the questions and retrieve texts that are relevant using DPR. We directly use the Haystack [21] implementation of DPR for our project. For each dataset we first create the document FAISS store and from there we get a resulting list counting retrieved texts in descending order of relevance. For each query (q_i), we choose the first $N+5$ documents (N being the number of documents actually corresponding to query i) as the predicted documents matching (q_i) by DPR. Later, we use our evaluation metrics to analyze the results.

Table 4: Dense Passage Retrieval Evaluation

Dataset	**precision$_{wa}$**	**recall$_{wa}$**	**f1-score$_{wa}$**
ENX	0.47	0.50	0.49
ESR	0.55	0.62	0.59
ENV	0.56	0.59	0.57

In Table 4 we can observe that for ENX dataset we get significantly improved results using DPR. The score for ENV dataset also has some improvements. However, for ESR dataset the results depreciate. As explained before, BM25 algorithm treats muti-word query (eg. deep-learning) as separate words and culminates those

scores while ranking documents whereas DPR treats such queries as a single term.

5.2.4 Zero-shot Learning with Transformer based pre-trained NLP Models

In subsection 4.7 we have described the zero-shot learning method along with the pre-trained (subsection 3.5) and fine-tuned NLP models we are using. In this section we present the evaluation results we retrieved using the zero-shot learning approach. Along with the pre-trained models, we have also fine-tuned the *BERT (bert-base-uncased)* with ENX and ESR dataset.

During our initial experiments we observed that the *hypothesis template* criterion in the zero-shot classifier pipeline influences the results. Therefore, we ran small tests to evaluate the effects of different *hypothesis template* options for the classifier. The default hypothesis template is *This example is {}.*, where the *{}* is replaced by the hypothesis, in our case keywords. We present our findings regarding this analysis in this section. We have used 8 different *hypothesis templates* in combination with two different transformer models (pre-trained BART and fine-tuned BERT_ESR). We conduct this experiment on the following two datasets, ENX and ENV.

Table 5: Template Evaluation - ENX - BART

template	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
This is a document about {}.	**0.78**	**0.77**	**0.78**
This text is about {}.	**0.78**	0.76	0.77
This is a text about {}.	0.77	0.75	0.76
This text contains {}.	0.77	0.75	0.76
This topic is {}.	0.77	0.74	0.76
This is an example of {}.	0.75	0.72	0.74
This example is {}.	0.73	0.7	0.72
{}.	0.69	0.69	0.69

Table 6: Template Evaluation - ENX - BERT-ESR

template	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
This is an example of {}.	**0.7**	0.51	0.59
This is a document about {}.	0.69	0.56	0.62
This is a text about {}.	0.69	**0.61**	**0.65**
This text contains {}.	0.68	0.51	0.58
This text is about {}.	0.67	0.6	0.63
This topic is {}.	0.67	0.53	0.59
{}.	0.65	0.46	0.54
This example is {}.	0.6	0.51	0.55

Table 7: Template Evaluation - ENV - BART

template	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	**f1-s$_{wa}$**
This is a document about {}.	**0.86**	**0.86**	**0.86**
This is a text about {}.	**0.86**	0.85	0.85
This text is about {}.	**0.86**	0.85	**0.86**
This topic is {}.	0.85	0.84	0.84
This example is {}.	0.83	0.81	0.82
This text contains {}.	0.83	0.81	0.82
This is an example of {}.	0.79	0.79	0.79
{}.	0.67	0.65	0.66

Table 8: Template Evaluation - ENV - BERT-ESR

template	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	**f1-s$_{wa}$**
This text is about {}.	**0.8**	0.7	0.75
This is an example of {}.	0.79	**0.73**	**0.76**
This is a document about {}.	0.79	**0.73**	**0.76**
This is a text about {}.	0.79	0.72	0.75
This topic is {}.	0.79	0.71	0.75
This text contains {}.	0.76	0.7	0.73
This example is {}.	0.74	0.66	0.7
{}.	0.72	0.6	0.66

In tables (Table 5, Table 6, Table 7, Table 8) the template, {}. indicates that no template was used. In the tables we can clearly observe that changes in sentence pattern and words have significant effects

on the results. Almost in all cases, the other *hypothesis templates* have better scores than the default template (*This example is {}.*). Although, different words (i.e. document, example, topic etc.) meaning same context have different results, sentences made with different combinations of same words have quite similar results. Such as, the templates *This is a text about {}.* and *This text is about {}.*. After carefully observing the results, we can conclude that amongst these examples the following three, *This is a text about {}.*, *This is a document about {}.* and *This text is about {}.* consistently do better than other examples. An aspect of the example sentences is, we began structuring new sentences from the default sentence by first changing the structure then the words. Therefore we can only conclude that our example sentences work better than the default *hypothesis template* (*This example is {}.*), there may exist other sentences that could work better than our top examples.

In our subsequent experiments we work with the following two *hypothesis templates*: *This is a text about {}.* and *This is a document about {}.* . For all the pre-trained transformer models we use the same subsets of our dataset as the previous experiments.

From the evaluation results (Table 9, Table 10, Table 11) it is evident that zero-shot classification does significantly better than the other approaches described so far on keyword based information retrieval task. Although the pre-trained classifiers were trained on shorter sentences, they do very well with longer texts as well. Amongst the classifiers, the *BART* model has notably higher scores compared to the other models across all three datasets in general. One important aspect we can observe from these scores is that, even the models fine-tuned on the same dataset distribution do not do better than the available pre-trained models. Which implies that we can achieve good results by applying pre-trained transformer based zero-shot learning without explicit fine-tuning. We can also notice here that amongst the two *hypothesis templates*, the template *This*

is a document about {}. is consistently better than *This is a text about {}.* which is on par with our findings from the *hypothesis template* study.

Table 9: Zero Shot Classification - ENX

model	**template**	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	**f1-s$_{wa}$**
BART	This is a document about {}.	**0.78**	**0.77**	**0.78**
BART	This is a text about {}.	0.77	0.75	0.76
BART-Yahoo	This is a text about {}.	0.77	0.76	0.76
BART-Yahoo	This is a document about {}.	0.76	0.76	0.76
RoBERTa	This is a document about {}.	0.74	0.73	0.73
BERT	This is a document about {}.	0.73	0.56	0.64
RoBERTa	This is a text about {}.	0.73	0.73	0.73
BERT-ENX	This is a document about {}.	0.7	0.57	0.63
BERT-ENX	This is a text about {}.	0.69	0.62	0.65
BERT-ESR	This is a document about {}.	0.69	0.56	0.62
BERT-ESR	This is a text about {}.	0.69	0.61	0.65
BERT	This is a text about {}.	0.63	0.5	0.56
DistilBART	This is a document about {}.	0.56	0.48	0.52
DistilBART	This is a text about {}.	0.51	0.44	0.47
SqueezeBERT	This is a text about {}.	0.42	0.29	0.35
SqueezeBERT	This is a document about {}.	0.41	0.31	0.35
DeBERTa	This is a document about {}.	0.21	0.22	0.21
DeBERTa	This is a text about {}.	0.16	0.1	0.12

Table 10: Zero Shot Classification - ESR

model	template	p_{wa}	r_{wa}	f1-s_{wa}
BERT	This is a document about {}.	**0.96**	**0.95**	**0.95**
BERT-ENX	This is a document about {}.	0.95	**0.95**	**0.95**
BERT-ENX	This is a text about {}.	0.95	**0.95**	**0.95**
BERT-ESR	This is a document about {}.	0.95	**0.95**	**0.95**
BERT-ESR	This is a text about {}.	0.95	**0.95**	**0.95**
BART	This is a text about {}.	0.91	0.89	0.9
BART	This is a document about {}.	0.9	0.9	0.9
RoBERTa	This is a document about {}.	0.89	0.87	0.88
BERT	This is a text about {}.	0.88	0.88	0.88
RoBERTa	This is a text about {}.	0.87	0.85	0.86
BART-Yahoo	This is a document about {}.	0.81	0.71	0.76
BART-Yahoo	This is a text about {}.	0.8	0.7	0.75
SqueezeBERT	This is a document about {}.	0.76	0.24	0.37
SqueezeBERT	This is a text about {}.	0.71	0.25	0.37
DeBERTa	This is a text about {}.	0.43	0.36	0.39
DeBERTa	This is a document about {}.	0.22	0.16	0.19
DistilBART	This is a document about {}.	0.14	0.09	0.11
DistilBART	This is a text about {}.	0.11	0.07	0.09

Table 11: Zero Shot Classification - ENV

model	template	p_{wa}	r_{wa}	f1-s_{wa}
BART-Yahoo	This is a document about {}.	**0.87**	**0.86**	**0.86**
BART	This is a document about {}.	0.86	**0.86**	**0.86**
BART	This is a text about {}.	0.86	0.85	0.85
BART-Yahoo	This is a text about {}.	0.86	0.84	0.85
RoBERTa	This is a document about {}.	0.83	0.77	0.8
RoBERTa	This is a text about {}.	0.83	0.76	0.79
BERT-ENX	This is a document about {}.	0.79	0.74	0.76
BERT-ESR	This is a document about {}.	0.79	0.73	0.76
BERT-ESR	This is a text about {}.	0.79	0.72	0.75
BERT	This is a document about {}.	0.79	0.74	0.76
BERT-ENX	This is a text about {}.	0.78	0.72	0.75
BERT	This is a text about {}.	0.78	0.7	0.74
SqueezeBERT	This is a text about {}.	0.68	0.39	0.49
SqueezeBERT	This is a document about {}.	0.65	0.37	0.48
DistilBART	This is a document about {}.	0.54	0.44	0.48
DistilBART	This is a text about {}.	0.52	0.45	0.48
DeBERTa	This is a text about {}.	0.21	0.17	0.19
DeBERTa	This is a document about {}.	0.05	0.08	0.06

5.2.5 Similarity Score Based Relevance Ranking

In section subsection 4.5, the components of the following ranking equation for our relevance ranking method has been explained in details.

$$R(t) = KC(t) \sum_{i} G(w_i, t) F(w_i) W(w_i) \tag{16}$$

In this section we observe the effect of different values of θ and and the similarity threshold st. $sim(w, t)$ plays an important role as by restricting it with a threshold (st), we can reduce the number of words we are considering in a document. Hence, only the words with a value above the threshold are the ones considered in calculation. At the same time we also observe how different word embedding models affect (sim) values. A total of 18 combinations of θ $(0, 0.1, 0.5)$ and similarity thresholds (st) $(0.25, 0.3, 0.35, 0.4, 0.5, 0.6)$ have been used here.

For each term, t in a dataset, we retrieve a descending ordered list containing document ID and corresponding rank values. From that list we consider the first $(N + 3)$ (N being the number of true positives for q in that corpus) to be the documents predicted by the relevance ranking equation that correspond to t. Then we match the list of true positives with the list of predicted positives to gather the evaluation score.

We have used four different embedding models to acquire the cosine similarity values between keywords and individual words from searchtext. These four models are the pre-trained fastText model (ft_en_cc), fastText model trained on ELIB dataset (elib_model_s_100 _e_5_w_5), ConceptNet NumberBatch (nb_vec) and ConceptNet NumberBatch combined with ft_en_cc (nb_vec_ft). The acquisition, model descriptions and usage of these models for cosine similarity score calculation have been discussed in subsection 4.4.

5.2.5.1 Relevance Ranking - ENX Dataset The ENX keywords from Table 1 and corresponding subset of ENX corpus has been used here as well with pre-processing. Table 4 shows the evaluation results where, st_θ is the parameter dictating the different similarity threshold (st) and θ values. Figure 11 shows the visualisation of the evaluation scores.

Table 12: Relevance Ranking Evaluation - ENX

(a) ft_en_cc

st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.47	0.49	0.48
0.25_0.5	0.46	0.47	0.46
0.25_0	0.47	0.49	0.48
0.3_0.1	0.5	0.52	**0.51**
0.3_0.5	0.48	0.49	0.48
0.3_0	0.49	0.51	0.5
0.35_0.1	0.5	**0.52**	**0.51**
0.35_0.5	0.49	0.51	0.5
0.35_0	0.5	**0.52**	**0.51**
0.4_0.1	0.47	0.48	0.48
0.4_0.5	0.47	0.49	0.48
0.4_0	0.47	0.48	0.48
0.5_0.1	0.56	0.35	0.43
0.5_0.5	0.55	0.34	0.42
0.5_0	0.56	0.35	0.43
0.6_0.1	**0.67**	0.16	0.26
0.6_0.5	**0.67**	0.16	0.26
0.6_0	**0.67**	0.16	0.26

(b) elib_model_s_100_e_5_w_5

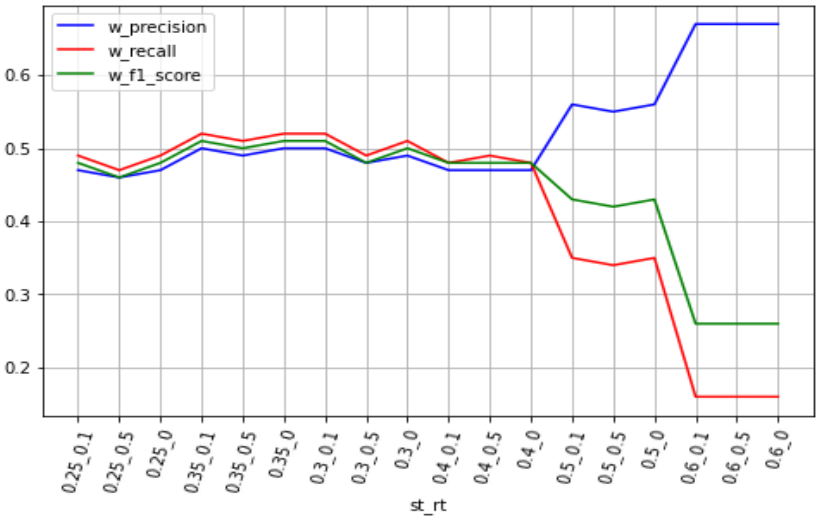
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.36	0.37	0.37
0.25_0.5	0.34	0.35	0.34
0.25_0	0.36	0.37	0.37
0.3_0.1	0.37	0.39	0.38
0.3_0.5	0.37	0.38	0.37
0.3_0	0.37	0.38	0.38
0.35_0.1	0.4	0.41	0.41
0.35_0.5	0.39	0.41	0.4
0.35_0	0.4	0.41	0.4
0.4_0.1	0.41	0.42	0.41
0.4_0.5	0.41	0.42	0.42
0.4_0	0.41	0.43	0.42
0.5_0.1	0.44	**0.46**	0.45
0.5_0.5	0.44	**0.46**	0.45
0.5_0	0.44	**0.46**	0.45
0.6_0.1	**0.45**	0.45	0.45
0.6_0.5	**0.45**	**0.46**	**0.46**
0.6_0	**0.45**	0.45	0.45

(c) nb_vec

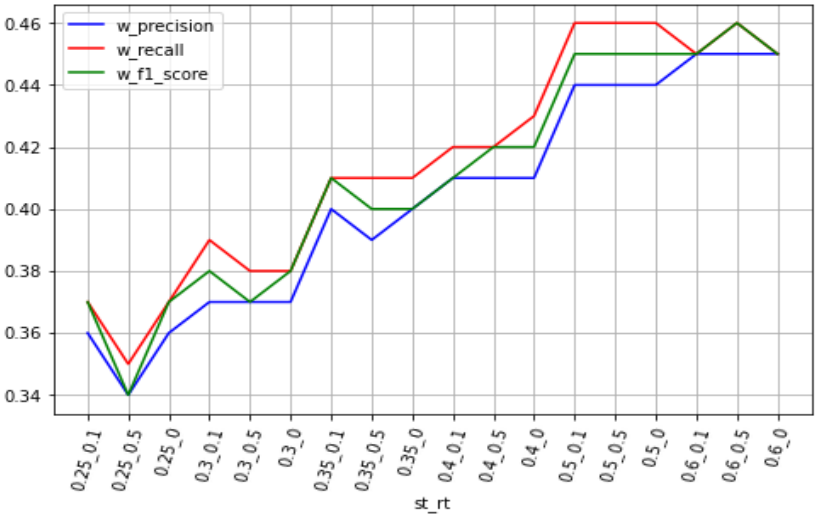
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.51	**0.53**	**0.52**
0.25_0.5	0.5	0.52	0.51
0.25_0	0.51	**0.53**	**0.52**
0.3_0.1	0.51	**0.53**	**0.52**
0.3_0.5	0.5	0.52	0.51
0.3_0	0.51	**0.53**	**0.52**
0.35_0.1	0.49	0.51	0.5
0.35_0.5	0.49	0.51	0.5
0.35_0	0.49	0.51	0.5
0.4_0.1	0.48	0.5	0.49
0.4_0.5	0.48	0.5	0.49
0.4_0	0.48	0.5	0.49
0.5_0.1	0.58	0.31	0.41
0.5_0.5	0.58	0.32	0.41
0.5_0	0.58	0.31	0.41
0.6_0.1	**0.71**	0.18	0.29
0.6_0.5	**0.71**	0.18	0.29
0.6_0	**0.71**	0.18	0.29

(d) nb_vec_ft

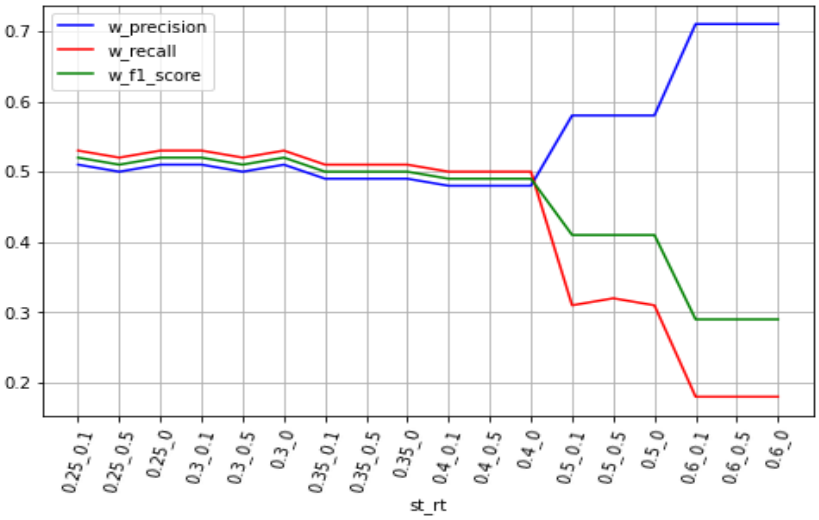
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.51	**0.53**	**0.52**
0.25_0.5	0.5	0.52	0.51
0.25_0	0.51	**0.53**	**0.52**
0.3_0.1	0.51	**0.53**	**0.52**
0.3_0.5	0.51	**0.53**	**0.52**
0.3_0	0.51	**0.53**	**0.52**
0.35_0.1	0.49	0.51	0.5
0.35_0.5	0.49	0.51	0.5
0.35_0	0.5	0.52	0.51
0.4_0.1	0.48	0.5	0.49
0.4_0.5	0.48	0.5	0.49
0.4_0	0.48	0.5	0.49
0.5_0.1	0.59	0.31	0.41
0.5_0.5	0.59	0.31	0.41
0.5_0	0.59	0.31	0.41
0.6_0.1	**0.73**	0.18	0.29
0.6_0.5	**0.73**	0.18	0.29
0.6_0	**0.73**	0.18	0.29



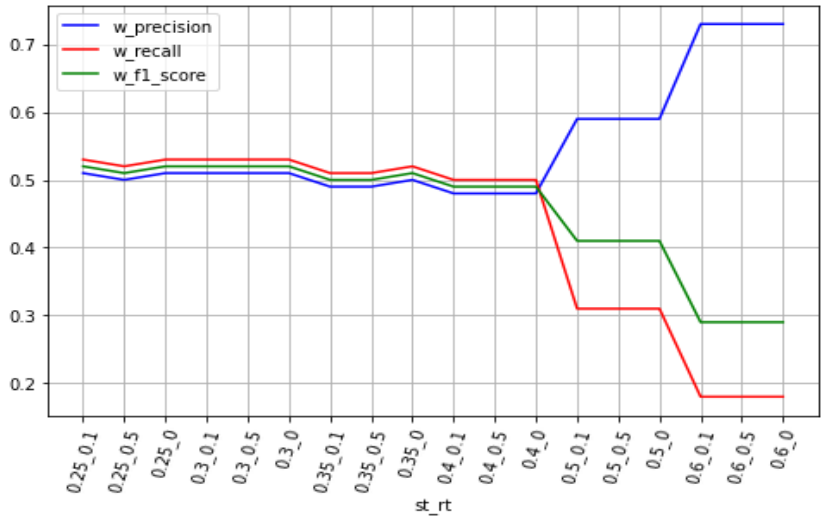
(a) ft_en_cc



(b) elib_model_s_100_e_5_w_5



(c) nb_vec



(d) nb_vec_ft

Figure 11: Relevance Ranking Results Visualisation - ENX

5.2.5.2 Relevance Ranking - ESR Dataset The ESR keywords from Table 1 and corresponding subset of the ESR corpus has been used with pre-processing. Table 5 shows the evaluation results and Figure 12 shows the visualisation of the evaluation scores.

Table 13: Relevance Ranking Evaluation - ESR

(a) ft_en_cc

st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.53	0.57	0.55
0.25_0.5	0.5	0.54	0.52
0.25_0	0.53	0.57	0.55
0.3_0.1	0.57	0.62	0.6
0.3_0.5	0.58	0.62	0.6
0.3_0	0.56	0.61	0.59
0.35_0.1	0.64	0.7	0.67
0.35_0.5	0.68	**0.73**	0.7
0.35_0	0.64	0.69	0.66
0.4_0.1	0.64	0.64	0.64
0.4_0.5	0.65	0.66	0.66
0.4_0	0.64	0.64	0.64
0.5_0.1	**0.83**	0.65	**0.73**
0.5_0.5	**0.83**	0.65	**0.73**
0.5_0	0.82	0.64	0.72
0.6_0.1	0.82	0.64	0.72
0.6_0.5	**0.83**	0.64	0.72
0.6_0	0.82	0.64	0.72

(b) elib_model_s_100_e_5_w_5

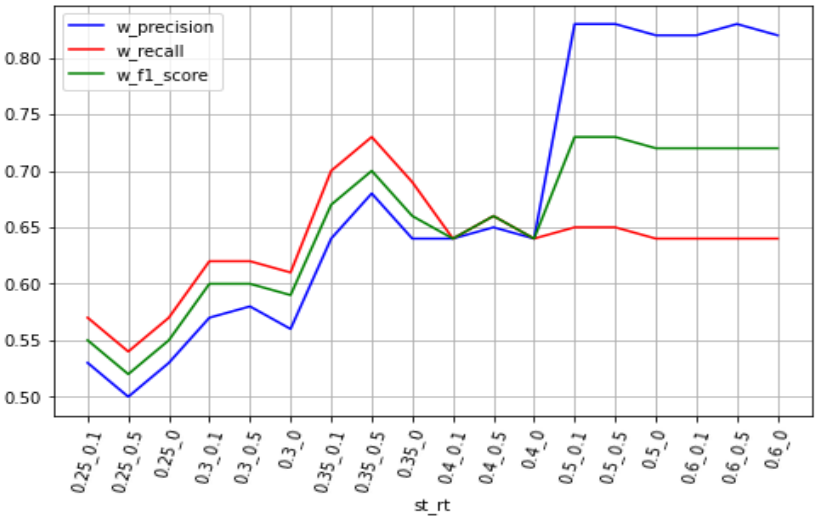
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.34	0.36	0.35
0.25_0.5	0.32	0.34	0.33
0.25_0	0.32	0.34	0.33
0.3_0.1	0.35	0.38	0.37
0.3_0.5	0.35	0.38	0.37
0.3_0	0.34	0.36	0.35
0.35_0.1	0.36	0.39	0.38
0.35_0.5	0.41	0.44	0.43
0.35_0	0.35	0.38	0.37
0.4_0.1	0.4	0.43	0.41
0.4_0.5	0.45	0.49	0.47
0.4_0	0.38	0.41	0.4
0.5_0.1	0.5	0.54	0.52
0.5_0.5	0.56	0.6	0.58
0.5_0	0.48	0.52	0.5
0.6_0.1	0.62	0.66	0.64
0.6_0.5	**0.68**	**0.74**	**0.71**
0.6_0	0.61	0.66	0.64

(c) nb_vec

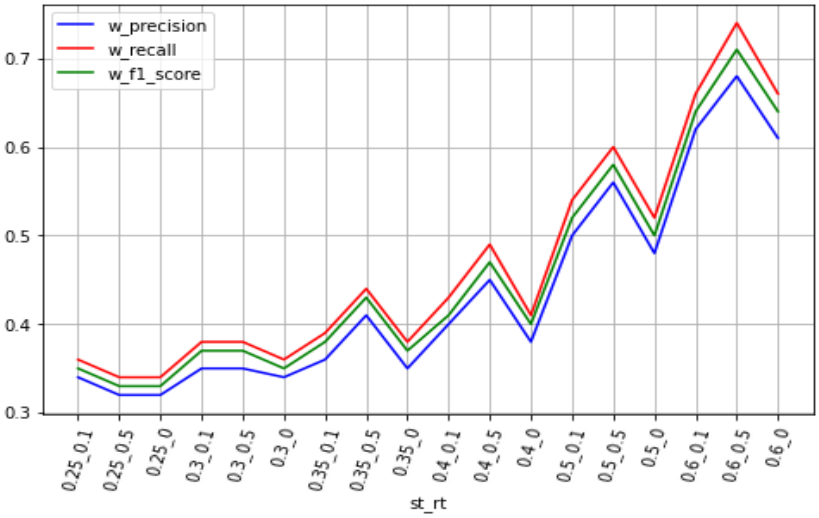
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.65	0.7	0.67
0.25_0.5	0.61	0.65	0.63
0.25_0	0.63	0.68	0.65
0.3_0.1	0.71	0.77	0.74
0.3_0.5	0.72	0.77	0.74
0.3_0	0.71	0.76	0.73
0.35_0.1	0.74	0.79	0.76
0.35_0.5	0.74	0.8	0.77
0.35_0	0.73	0.79	0.76
0.4_0.1	0.75	0.81	0.78
0.4_0.5	0.75	0.81	0.78
0.4_0	0.75	0.8	0.77
0.5_0.1	0.75	0.8	0.77
0.5_0.5	0.75	0.81	0.78
0.5_0	0.75	0.8	0.77
0.6_0.1	**0.78**	**0.82**	**0.8**
0.6_0.5	**0.78**	**0.82**	**0.8**
0.6_0	**0.78**	**0.82**	**0.8**

(d) nb_vec_ft

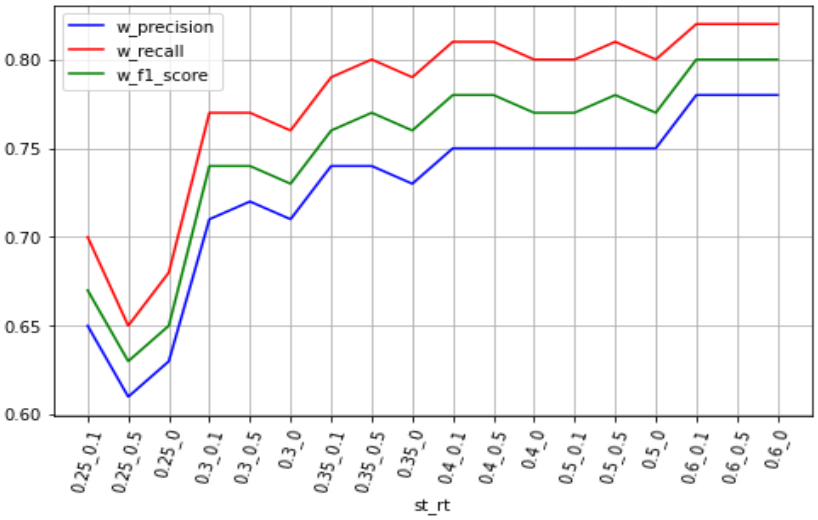
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.53	0.57	0.55
0.25_0.5	0.51	0.55	0.53
0.25_0	0.52	0.56	0.54
0.3_0.1	0.58	0.63	0.61
0.3_0.5	0.58	0.63	0.61
0.3_0	0.58	0.62	0.6
0.35_0.1	0.64	0.7	0.67
0.35_0.5	0.68	**0.74**	0.71
0.35_0	0.64	0.69	0.66
0.4_0.1	0.63	0.64	0.64
0.4_0.5	0.65	0.66	0.66
0.4_0	0.63	0.64	0.64
0.5_0.1	0.8	0.64	0.72
0.5_0.5	0.81	0.65	**0.73**
0.5_0	0.8	0.64	0.71
0.6_0.1	0.82	0.64	0.72
0.6_0.5	**0.83**	0.64	0.72
0.6_0	0.82	0.64	0.72



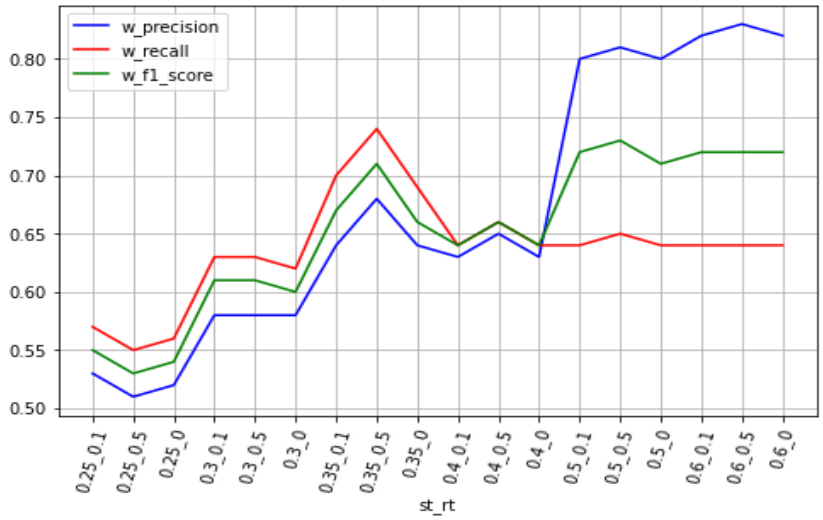
(a) ft_en_cc



(b) elib_model_s_100_e_5_w_5



(c) nb_vec



(d) nb_vec_ft

Figure 12: Relevance Ranking Results Visualisation - ESR

5.2.5.3 Relevance Ranking - ENV Dataset The ENV keywords from Table 1 and corresponding subset of the ENV corpus has been used with pre-processing. Table 6 shows the evaluation results and Figure 13 shows the visualisation of the evaluation scores.

Table 14: Relevance Ranking Evaluation - ENV

(a) ft_en_cc

st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.55	0.56	0.55
0.25_0.5	0.49	0.51	0.5
0.25_0	0.55	0.56	0.55
0.3_0.1	0.58	0.6	0.59
0.3_0.5	0.57	0.58	0.57
0.3_0	0.58	0.6	0.59
0.35_0.1	0.62	0.63	0.62
0.35_0.5	0.61	0.63	0.62
0.35_0	0.62	0.63	0.62
0.4_0.1	0.64	0.66	0.65
0.4_0.5	0.65	0.67	0.66
0.4_0	0.64	0.66	0.65
0.5_0.1	0.66	**0.68**	0.67
0.5_0.5	0.66	**0.68**	0.67
0.5_0	0.66	**0.68**	0.67
0.6_0.1	**0.74**	**0.68**	**0.71**
0.6_0.5	0.73	**0.68**	**0.71**
0.6_0	**0.74**	**0.68**	**0.71**

(b) elib_model_s_100_e_5_w_5

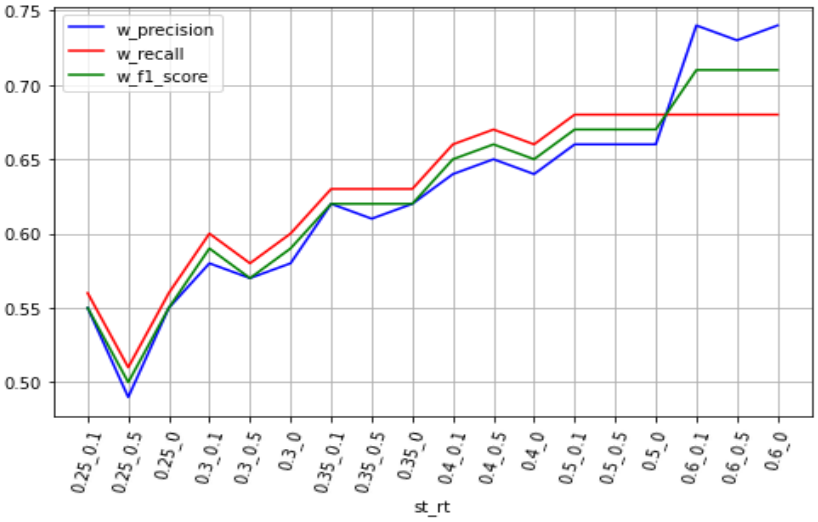
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.4	0.41	0.41
0.25_0.5	0.37	0.38	0.37
0.25_0	0.41	0.42	0.41
0.3_0.1	0.43	0.44	0.43
0.3_0.5	0.41	0.42	0.42
0.3_0	0.43	0.44	0.43
0.35_0.1	0.46	0.47	0.47
0.35_0.5	0.46	0.47	0.47
0.35_0	0.46	0.47	0.46
0.4_0.1	0.49	0.5	0.49
0.4_0.5	0.51	0.52	0.51
0.4_0	0.48	0.49	0.49
0.5_0.1	0.51	0.52	0.51
0.5_0.5	0.52	0.54	0.53
0.5_0	0.51	0.52	0.51
0.6_0.1	0.58	0.57	**0.58**
0.6_0.5	**0.59**	**0.58**	**0.58**
0.6_0	0.58	0.57	0.57

(c) nb_vec

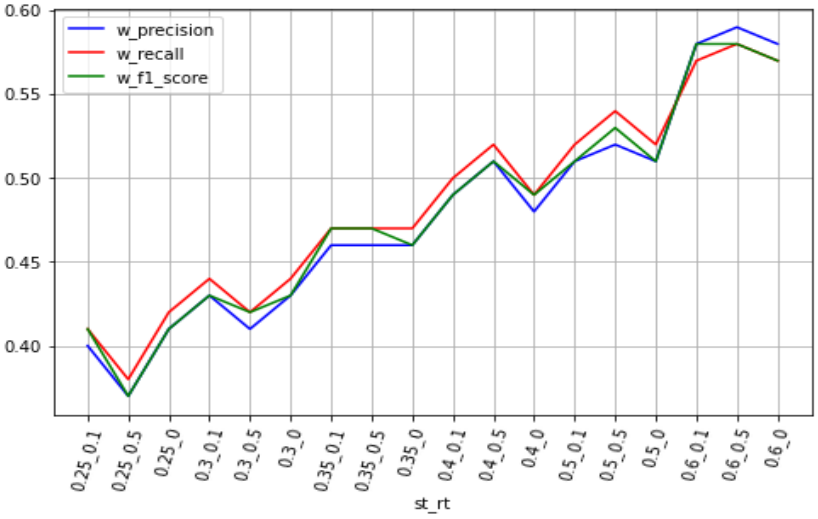
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.6	0.62	0.61
0.25_0.5	0.57	0.59	0.58
0.25_0	0.6	0.62	0.61
0.3_0.1	0.64	0.66	0.65
0.3_0.5	0.63	0.65	0.64
0.3_0	0.64	0.65	0.64
0.35_0.1	0.66	0.68	0.67
0.35_0.5	0.66	0.68	0.67
0.35_0	0.65	0.67	0.66
0.4_0.1	0.66	0.68	0.67
0.4_0.5	0.67	**0.69**	0.68
0.4_0	0.66	0.68	0.67
0.5_0.1	0.68	0.68	0.68
0.5_0.5	0.69	**0.69**	**0.69**
0.5_0	0.68	**0.69**	0.68
0.6_0.1	**0.72**	0.64	0.68
0.6_0.5	**0.72**	0.64	0.68
0.6_0	**0.72**	0.64	0.68

(d) nb_vec_ft

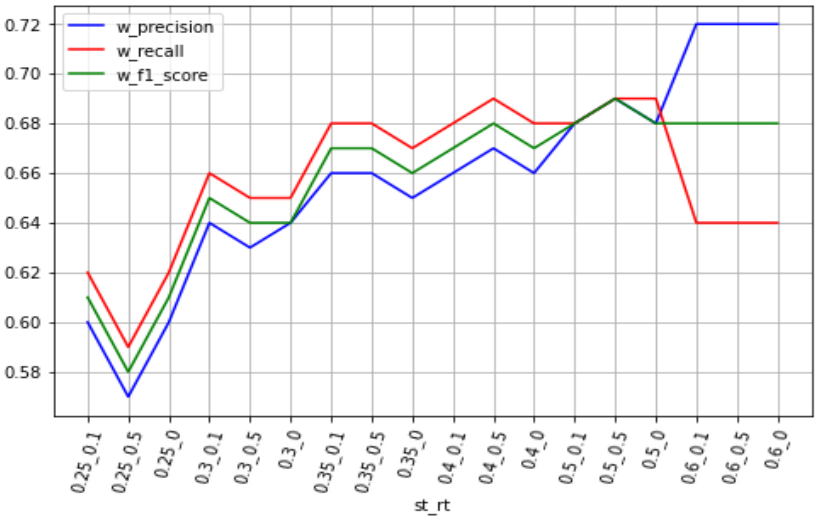
st_θ	**p_{wa}**	**r_{wa}**	**f1-s_{wa}**
0.25_0.1	0.61	0.62	0.61
0.25_0.5	0.57	0.59	0.58
0.25_0	0.6	0.62	0.61
0.3_0.1	0.64	0.66	0.65
0.3_0.5	0.63	0.65	0.64
0.3_0	0.64	0.66	0.65
0.35_0.1	0.66	0.68	0.67
0.35_0.5	0.65	0.67	0.66
0.35_0	0.66	0.68	0.67
0.4_0.1	0.66	0.68	0.67
0.4_0.5	0.68	**0.7**	**0.69**
0.4_0	0.66	0.68	0.67
0.5_0.1	0.68	0.68	0.68
0.5_0.5	0.69	0.69	**0.69**
0.5_0	0.68	0.69	0.68
0.6_0.1	**0.72**	0.64	0.68
0.6_0.5	**0.72**	0.64	0.68
0.6_0	**0.72**	0.64	0.68



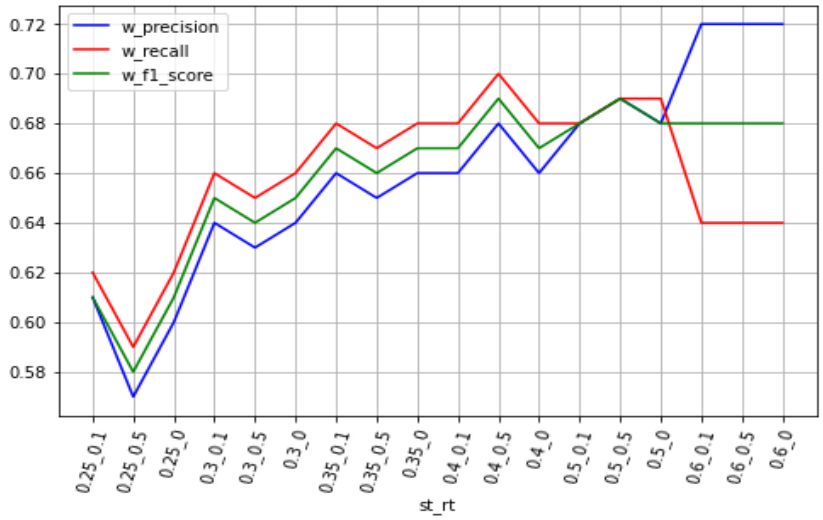
(a) ft_en_cc



(b) elib_model_s_100_e_5_w_5



(c) nb_vec



(d) nb_vec_ft

Figure 13: Relevance Ranking Results Visualisation - ENV

In the figures (Figure 11, Figure 12, Figure 13) we can observe that after a certain value of similarity threshold, if we keep increasing it although *weighted average precision* (p_{wa}) increases but *weighted average recall* (r_{wa}) and *weighted average f1-score* (*f1-s*$_{wa}$) starts to decrease which is not desirable. However, this tendency also reflects that with increasing similarity threshold, the precision increases but recall decreases and relevance ranking starts to behave somewhat like the keyword search approach. For the case of Enx dataset, the optimal value of the evaluation metrics can be between 0.35 – 0.5. Similarly, for ESR and ENV dataset, we can choose a value between 0.5 – 0.6. From the tables (Table 12, Table 13, Table 14) it is evident that the value of θ does not have much contribution to the evaluation scores rather the similarity thresholds (st) have the major contribution to the final evaluation scores. Another observation is that the fastText model trained on ELIB dataset (elib_model_s_100_e_5_w_5) does not do well compared to the other three embedding models. The evaluation results of ENX and ENV dataset has a significant improvement over Keyword Search (Table 2 and Okapi BM25 Search (Table 3 especially for *weighted average recall* (r_{wa}) and *weighted av-*

erage f1-score (f1-s_{wa}). However, for the case of ESR dataset, it already had very good scores with Okapi BM25 (Table 3). We do not observe improvement using relevance ranking on this dataset.

5.2.6 Similarity Score Based Logistic Regression

In subsection 4.6 we have described the motivation and strategy behind using word embedding in combination with logistic regression for document retrieval.

We use the same keywords and corpus subsets for all 3 datasets as the previous experiments. In the tables (Table 15, Table 16, Table 17) we can observe that the evaluation scores on ENX and ENV datasets show improvement over keyword search (Table 2) and Okapi BM25 search (Table 3). ESR dataset does not have improvement over Okapi BM25 search (Table 3) but it has better scores here than relevance ranking (Table 13). Amongst the different embedding models, the combination of ft_en_cc, nb_vec and elib_model_s_100_e_5_w_5 has better recall and precision scores whereas ft_en_cc has better precision scores. If we look at the keyword based evaluation graphs (Figure 15, Figure 17, Figure 19), there is no observable pattern to analyze which embedding models is better for a dataset as different keywords score better on different embeddings.

5.2.6.1 Logistic Regression - ENX Dataset Similarity Score base Logistic Regression on ENX dataset.

Table 15: Document Retrieval Evaluation - ENX

embedding	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	**f1-s$_{wa}$**
ft	**0.66**	0.3	0.42
elib	0.64	0.32	0.42
ft_elib	0.64	0.35	0.45
ft_nb_ft	0.64	0.34	0.44
nb_ft	0.63	0.34	0.44
ft_nb	0.63	0.38	**0.48**
nb	0.61	0.37	0.46
ft_nb_elib	0.61	**0.4**	**0.48**

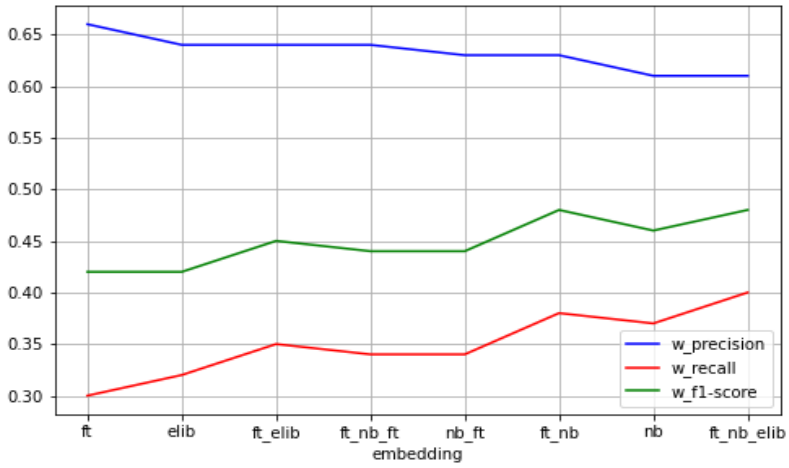
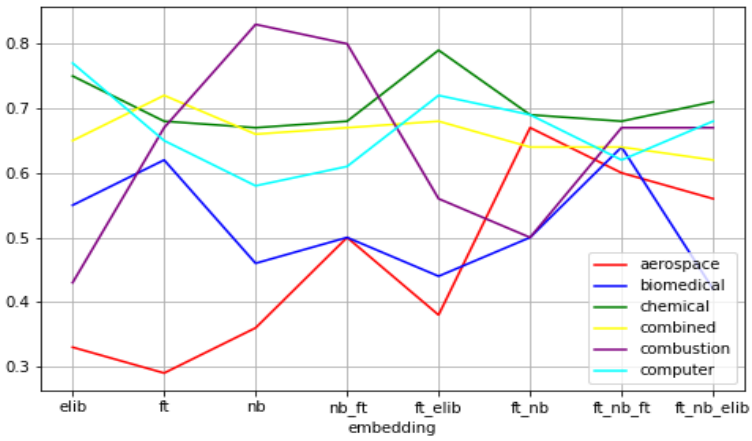
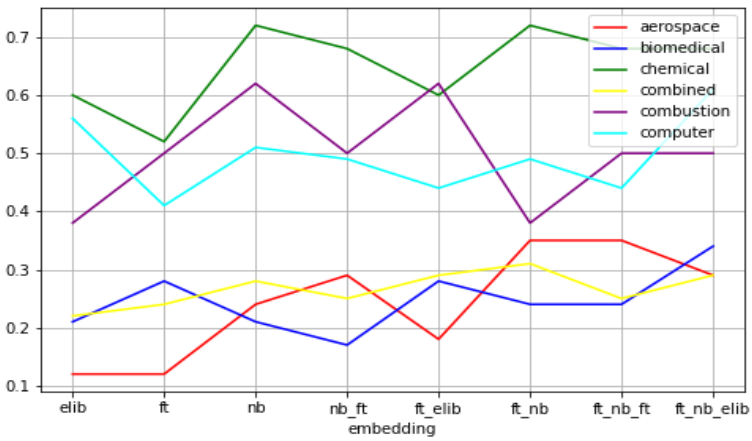


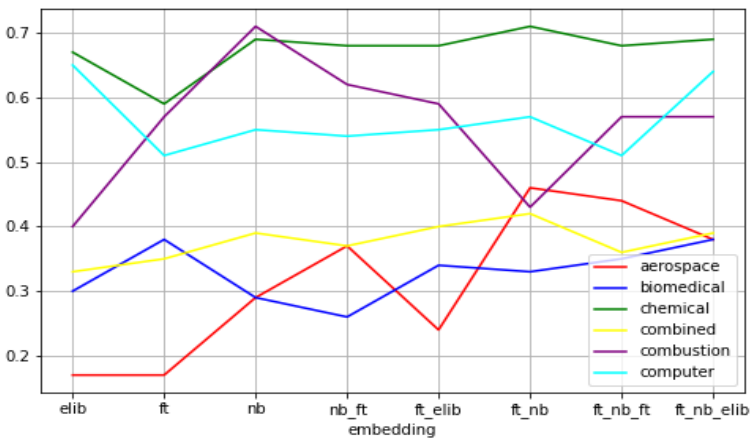
Figure 14: Logistic Regression Scores Visualisation (weighted) - ENX



(a) Precision



(b) Recall



(c) f1_score

Figure 15: Logistic Regression Scores Visualisation - ENX

5.2.6.2 Logistic Regression - ESR Dataset

Similarity Score base Logistic Regression on ESR dataset.

Table 16: Logistic Regression Evaluation - ESR

embedding	p_{wa}	r_{wa}	f1-s_{wa}
ft	**0.91**	0.64	0.75
nb_ft	**0.91**	0.63	0.75
elib	0.9	0.74	0.81
nb	0.9	0.76	**0.82**
ft_nb_ft	0.89	0.64	0.74
ft_nb	0.87	0.77	**0.82**
ft_elib	0.85	0.74	0.79
ft_nb_elib	0.78	**0.78**	0.78

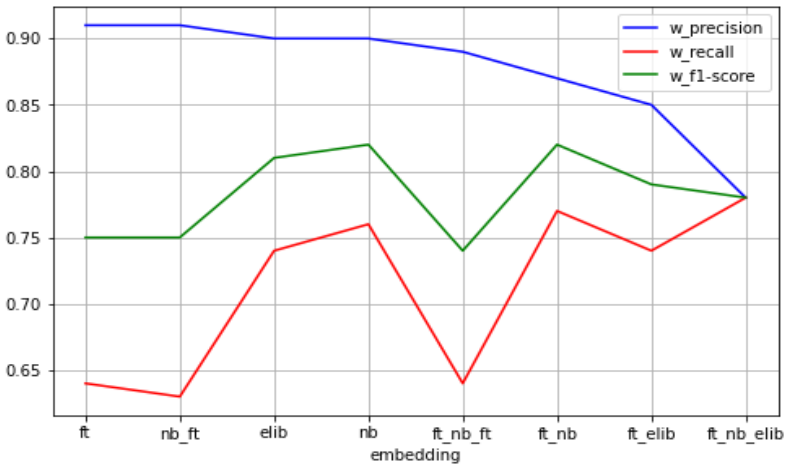
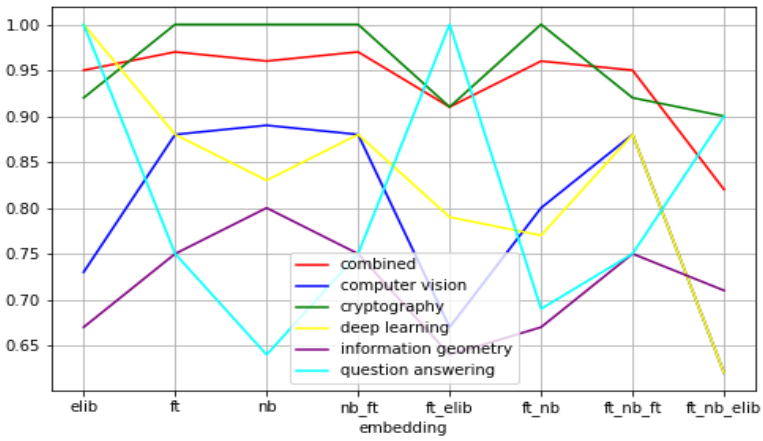
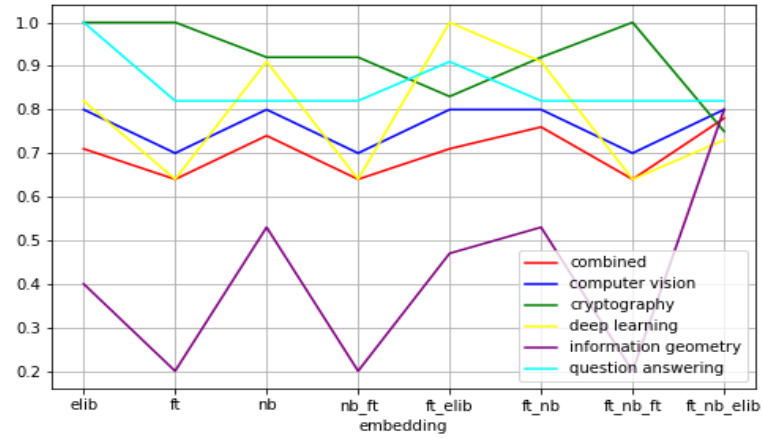


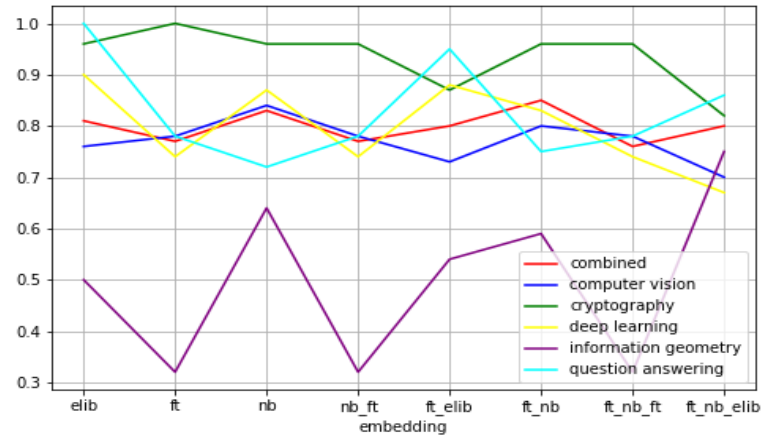
Figure 16: Logistic Regression Scores Visualisation (weighted) - ESR



(a) Precision



(b) Recall



(c) f1_score

Figure 17: Logistic Regression Scores Visualisation - ESR

5.2.6.3 Logistic Regression - ENV Dataset

Similarity Score base Logistic Regression on ENV dataset.

Table 17: Logistic Regression Evaluation - ENV

embedding	p_{wa}	r_{wa}	f1-s_{wa}
ft	**0.71**	0.41	0.52
nb	**0.71**	0.43	0.54
nb_ft	**0.71**	0.45	0.55
ft_elib	**0.71**	0.43	0.54
elib	0.7	0.31	0.43
ft_nb_ft	0.7	0.47	**0.56**
ft_nb_elib	0.7	0.45	0.55
ft_nb	0.69	**0.46**	**0.56**

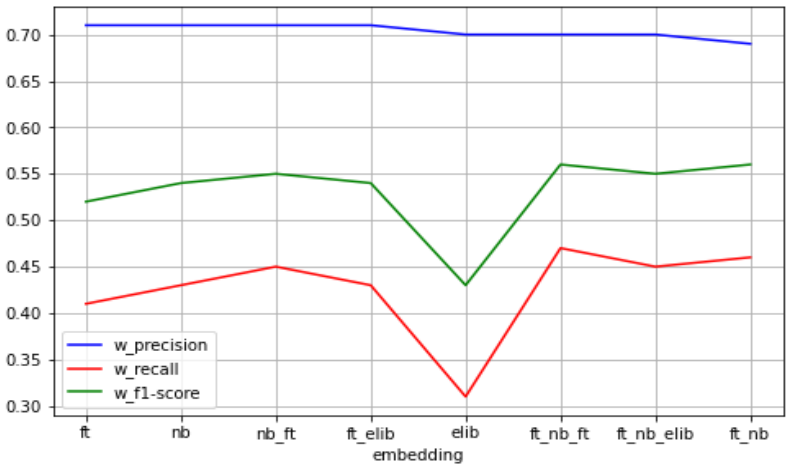
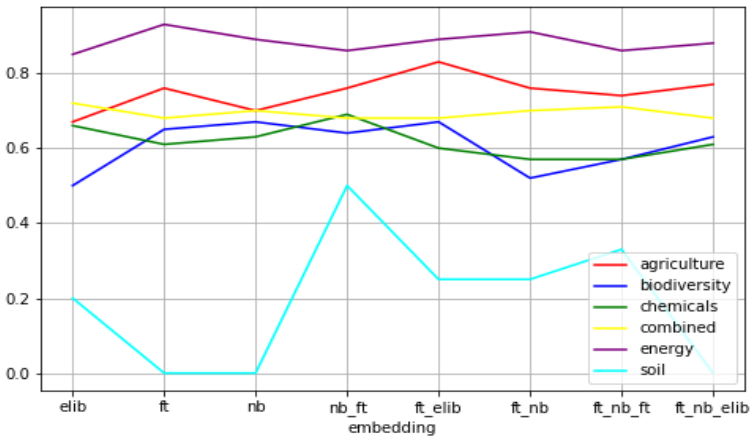
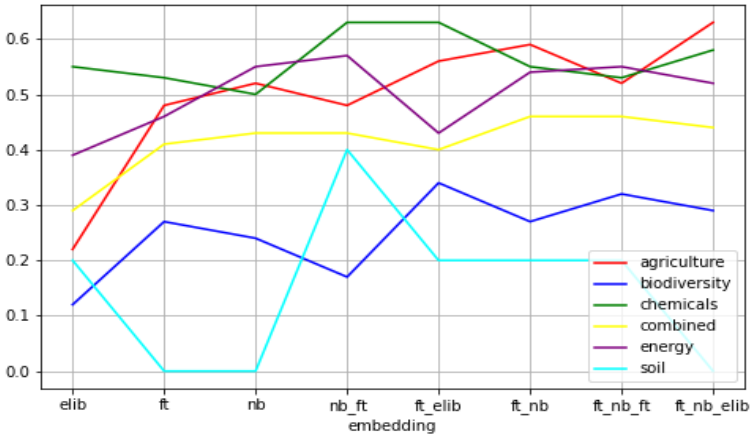


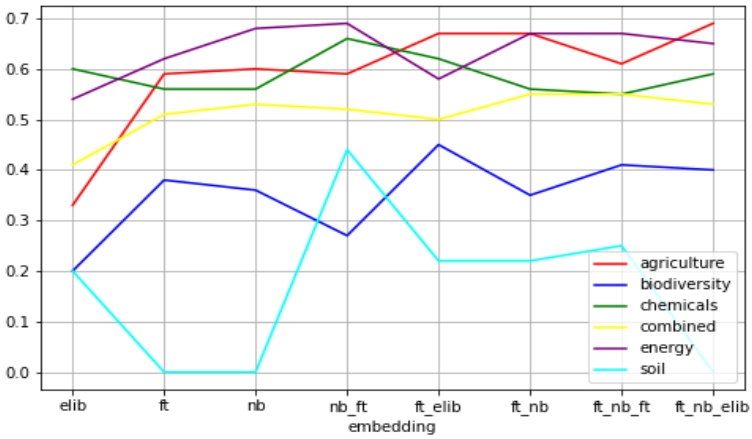
Figure 18: Logistic Regression Scores Visualisation (weighted) - ENV



(a) Precision



(b) Recall



(c) f1_score

Figure 19: Logistic Regression Scores Visualisation - ENV

6 Discussion

In this section we briefly present our best results from the different document retrieval approaches we have presented so far. Table 18 contains results from the **Keyword Search** (Table 2), **Okapi-BM25** (Table 3) and **Dense Passage Retrieval (DPR)** (Table 4) in the first three rows respectively. The last three rows contain the best models we have selected from the following three document retrieval methods proposed by us in this thesis, **Zero-shot Learning with Transformer based pre-trained NLP Models**, **Similarity Score Based Relevance Ranking** and **Similarity Score Based Logistic Regression** respectively. It is to be noted that from the many different variations of models we have experimented with in our proposed approaches, the model that performs relatively well on *all* three datasets has been picked as the most optimal model for its respective approach.

From the the **Zero-shot Learning with Transformer based pre-trained NLP Models** approach, we have chosen the **BART** pre-trained model as the most suitable transformer based pre-trained NLP model for document retrieval task. Amongst all the pre-trained NLP models, the **BART** model generally performs well on all three datasets (**subsubsection 5.2.4**). Between the two *hypothesis templates* we have experimented with, *This is a document about* works better than *This is a text about* for the **BART** model. In the **Similarity Score Based Relevance Ranking** method, the **nb_vec** model has comparatively good scores on all three datasets. There are other models that work particularly well on a specific dataset. However, in real world the data distribution will be unknown and we need a model that works well on different data distributions. Hence, **nb_vec** has been selected as the best embedding model for relevance ranking based document retrieval. Furthermore, from the tables presented in **subsubsection 5.2.5**, we can observe that the optimal *similarity*

threshold (st) for the **nb_vec** is 0.4 across all datasets.

Finally, from the **Similarity Score Based Logistic Regression** experiment, the combined dataframe **ft_nb** from the similarity scores of **ft_en_** and **nb_vec** embedding models performs good on all datasets.

Table 18: Summary of Results on all Datasets

	ENX			ESR			ENV		
method	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	$\mathbf{f_{wa}}$	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	$\mathbf{f_{wa}}$	$\mathbf{p_{wa}}$	$\mathbf{r_{wa}}$	$\mathbf{f_{wa}}$
Keyword Search	0.73	0.14	0.23	**0.96**	0.68	0.79	0.72	0.59	0.65
Okapi -BM25	0.3	0.33	0.32	0.81	**0.92**	0.86	0.55	0.58	0.56
DPR	0.47	0.5	0.49	0.54	0.61	0.58	0.56	0.59	0.57
Transformer Models **BART** (This is a docu -ment about)	**0.78**	**0.77**	**0.78**	0.9	0.9	**0.9**	**0.86**	**0.86**	**0.86**
Relevance Ranking **nb_vec (st = 0.4)**	0.48	0.5	0.49	0.75	0.81	0.78	0.67	0.69	0.68
Logistic Regression **ft_nb**	0.63	0.38	0.48	0.87	0.77	0.82	0.69	0.46	0.56

Both of our similarity score based approaches perform better on all three datasets as a whole compared to the baseline approaches (keyword search, Okapi-Bm25 and DPR). Looking at the recall scores

of these two methods, we can observe that the **Similarity Score Based Relevance Ranking** method has better performance than the **Similarity Score Based Logistic Regression**.

From Table 18 we can see that the textbfZero-shot Learning with Transformer based pre-trained NLP Models approach for document retrieval has best scores compared to all other methods. However, the computation time needed to classify each document in a dataset after loading a transformer based NLP model is approximately **22.5 minutes** (experimented on ENX dataset with 5 keywords as *hypothesis* on a setup with - Intel(R) Core(TM) i7-8850H CPU at 2.60GHz and 32GB RAM capacity). This value increases proportionately with the length of documents and the number of classes being considered.

On the other hand, after loading the vectors, the computational time required to calculate the similarity score of each word in a document against all 5 keywords is only approximately **8.8 seconds** with same technical setup. This makes our **Similarity Score Based Relevance Ranking** approach more useful compared to **Zero-shot Learning with Transformer based pre-trained NLP Models** for situations where a huge number of documents have to be processed and a quick response is expected .

7 Conclusion and Future Work

The objective of this thesis was to enhance keyword based document retrieval models. We conclude our experiments with three different proposed approaches for document retrieval, all of which perform better than the classical document retrieval methods we have considered as baseline. Amongst our proposed model the best performance is observed on the **Zero-shot Learning with Transformer based pre-trained NLP Models** approach. However, as explained before the pre-trained NLP models used in the method take significantly more computation time in comparison to word embed-

ding models calculating similarity scores on same datasets. Therefore, as mentioned in (section 6), **Similarity Score Based Relevance Ranking** is a more appropriate model for term based document retrieval. The model is highly capable of retrieving documents that are contextually relevant but do not contain the search term with good reliability.

We have already observed that embedding models (subsubsection 3.4.3) benefiting from both knowledge graph and distributed semantic word embeddings perform better than models that only contain the later. Therefore, the possible future direction from here would be to improve the use of such word embedding models for fast document retrieval. Alternatively, reducing the time need for computing class probability using transformer based NLP models will make the pre-trained NLP models highly advantageous for document retrieval tasks. Lastly, as observed in our experiments, since the *hypothesis template* needed in the transformer based zero-shot classifier pipeline affects the performance, developing a proper approach for finding the optimal *hypothesis template* will be beneficial not only for document retrieval tasks but for many other tasks where the classifier pipeline is used.

Appendix

7.1 Logistic Regression - Scores on individual keywords

Table 19: Similarity Score based Logistic Regression Evaluation - ENX

(a) Precision

embedding	q_1	q_2	q_3	q_4	q_5	q_{com}
elib	0.55	0.43	**0.77**	0.33	0.75	0.65
ft	0.62	0.67	0.65	0.29	0.68	**0.72**
nb	0.46	**0.83**	0.58	0.36	0.67	0.66
nb_ft	0.5	0.8	0.61	0.5	0.68	0.67
ft_elib	0.44	0.56	0.72	0.38	**0.79**	0.68
ft_nb	0.5	0.5	0.69	**0.67**	0.69	0.64
ft_nb_ft	**0.64**	0.67	0.62	0.6	0.68	0.64
ft_nb_elib	0.42	0.67	0.68	0.56	0.71	0.62

(b) Recall

embedding	q_1	q_2	q_3	q_4	q_5	q_{com}
elib	0.21	0.38	**0.56**	0.12	0.6	0.22
ft	0.28	0.5	0.41	0.12	0.52	0.24
nb	0.21	**0.62**	0.51	0.24	**0.72**	0.28
nb_ft	0.17	0.5	0.49	0.29	0.68	0.25
ft_elib	0.28	**0.62**	0.44	0.18	0.6	0.29
ft_nb	0.24	0.38	0.49	**0.35**	**0.72**	**0.31**
ft_nb_ft	0.24	0.5	0.44	**0.35**	0.68	0.25
ft_nb_elib	**0.34**	0.5	0.61	0.29	0.68	0.29

(c) f1_score

embedding	q_1	q_2	q_3	q_4	q_5	q_{com}
elib	0.3	0.4	**0.65**	0.17	0.67	0.33
ft	**0.38**	0.57	0.51	0.17	0.59	0.35
nb	0.29	**0.71**	0.55	0.29	0.69	0.39
nb_ft	0.26	0.62	0.54	0.37	0.68	0.37
ft_elib	0.34	0.59	0.55	0.24	0.68	0.4
ft_nb	0.33	0.43	0.57	**0.46**	**0.71**	**0.42**
ft_nb_ft	0.35	0.57	0.51	0.44	0.68	0.36
ft_nb_elib	**0.38**	0.57	0.64	0.38	0.69	0.39

Table 20: Similarity Score based Logistic Regression - ESR

(a) Precision

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	**1.0**	**1.0**	0.73	0.67	0.92	0.95
ft	0.88	0.75	0.88	0.75	**1.0**	**0.97**
nb	0.83	0.64	**0.89**	**0.8**	**1.0**	0.96
nb_ft	0.88	0.75	0.88	0.75	**1.0**	**0.97**
ft_elib	0.79	**1.0**	0.67	0.64	0.91	0.91
ft_nb	0.77	0.69	0.8	0.67	**1.0**	0.96
ft_nb_ft	0.88	0.75	0.88	0.75	0.92	0.95
ft_nb_elib	0.62	0.9	0.62	0.71	0.9	0.82

(b) Recall

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	0.82	**1.0**	**0.8**	0.4	**1.0**	0.71
ft	0.64	0.82	0.7	0.2	**1.0**	0.64
nb	0.91	0.82	**0.8**	0.53	0.92	0.74
nb_ft	0.64	0.82	0.7	0.2	0.92	0.64
ft_elib	**1.0**	0.91	**0.8**	0.47	0.83	0.71
ft_nb	0.91	0.82	**0.8**	0.53	0.92	0.76
ft_nb_ft	0.64	0.82	0.7	0.2	**1.0**	0.64
ft_nb_elib	0.73	0.82	**0.8**	**0.8**	0.75	**0.78**

(c) f1_score

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	**0.9**	**1.0**	0.76	0.5	0.96	0.81
ft	0.74	0.78	0.78	0.32	**1.0**	0.77
nb	0.87	0.72	**0.84**	0.64	0.96	0.83
nb_ft	0.74	0.78	0.78	0.32	0.96	0.77
ft_elib	0.88	0.95	0.73	0.54	0.87	0.8
ft_nb	0.83	0.75	0.8	0.59	0.96	**0.85**
ft_nb_ft	0.74	0.78	0.78	0.32	0.96	0.76
ft_nb_elib	0.67	0.86	0.7	**0.75**	0.82	0.8

Table 21: Similarity Score based Logistic Regression - ENV

(a) Precision

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	0.85	0.5	0.2	0.67	0.66	**0.72**
ft	**0.93**	0.65	0.0	0.76	0.61	0.68
nb	0.89	**0.67**	0.0	0.7	0.63	0.7
nb_ft	0.86	0.64	**0.5**	0.76	**0.69**	0.68
ft_elib	0.89	**0.67**	0.25	**0.83**	0.6	0.68
ft_nb	0.91	0.52	0.25	0.76	0.57	0.7
ft_nb_ft	0.86	0.57	0.33	0.74	0.57	0.71
ft_nb_elib	0.88	0.63	0.0	0.77	0.61	0.68

(b) Recall

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	0.39	0.12	0.2	0.22	0.55	0.29
ft	0.46	0.27	0.0	0.48	0.53	0.41
nb	0.55	0.24	0.0	0.52	0.5	0.43
nb_ft	**0.57**	0.17	**0.4**	0.48	**0.63**	0.43
ft_elib	0.43	**0.34**	0.2	0.56	**0.63**	0.4
ft_nb	0.54	0.27	0.2	0.59	0.55	**0.46**
ft_nb_ft	0.55	0.32	0.2	0.52	0.53	**0.46**
ft_nb_elib	0.52	0.29	0.0	**0.63**	0.58	0.44

(c) f1_score

embedding	**q_1**	**q_2**	**q_3**	**q_4**	**q_5**	**q_{com}**
elib	0.54	0.2	0.2	0.33	0.6	0.41
ft	0.62	0.38	0.0	0.59	0.56	0.51
nb	0.68	0.36	0.0	0.6	0.56	0.53
nb_ft	**0.69**	0.27	**0.44**	0.59	**0.66**	0.52
ft_elib	0.58	**0.45**	0.22	0.67	0.62	0.5
ft_nb	0.67	0.35	0.22	0.67	0.56	**0.55**
ft_nb_ft	0.67	0.41	0.25	0.61	0.55	**0.55**
ft_nb_elib	0.65	0.4	0.0	**0.69**	0.59	0.53

List of Figures

List of Tables

References

[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[2] David C Blair and Melvin E Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*, 28(3):289–299, 1985.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[4] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

[5] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[6] Dorian Brown. A collection of bm25 algorithms in python. `https://github.com/dorianbrown/rank_bm25`, 2021.

[7] Shyi-Ming Chen and Jeng-Yih Wang. Document retrieval using knowledge-based fuzzy information retrieval techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5):793–803, 1995.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[10] Joe Davison. bart-large-mnli-yahoo-answers. https://huggingface.co/joeddav/bart-large-mnli-yahoo-answers, 2021.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2584–2591, 2013.

[13] ELIB-DLR. Electronic library, dlr. https://elib.dlr.de/, 2021.

[14] engrXiv. engrxiv preprints. https://engrxiv.org/, 03 April, 2021.

[15] fastText. Word vectors for 157 languages. https://fasttext.cc/docs/en/crawl-vectors.html, 2021.

[16] EC Directorate General for Environment. Science for environment policy news site. https://ec.europa.eu/environment/integration/research/newsalert/, 2011-2020.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[18] Google. Google code archive - long-term storage for google code project hosting. https://code.google.com/archive/p/word2vec/, 2021.

[19] Andreas Hamm and Simon Odrowski. Term-community-based topic detection with variable resolution. *Information*, 12(6):221, 2021.

[20] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[21] Haystack. Better retrieval via "dense passage retrieval". https://haystack.deepset.ai/docs/latest/tutorial6md, 2021.

[22] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[23] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, 6:3–10, 1994.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Forrest N Iandola, Albert E Shaw, Ravi Krishna, and Kurt W Keutzer. Squeezebert: What can computer vision teach nlp about efficient neural networks? *arXiv preprint arXiv:2006.11316*, 2020.

[26] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[27] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[28] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[29] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[30] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[31] Yann Le Cun and Françoise Fogelman-Soulié. Modèles connexionnistes de l'apprentissage. *Intellectica*, 2(1):114–143, 1987.

[32] Jieh-Sheng Lee and Jieh Hsiang. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124*, 2019.

[33] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[34] Qian Liu, Heyan Huang, Jie Lut, Yang Gao, and Guangquan Zhang. Enhanced word embedding similarity measures using fuzzy rules for query expansion. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2017.

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[36] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[38] Suraj Patil. Distilbart. https://huggingface.co/valhalla/distilbart-mnli-12-3, 2021.

[39] Dong Qiu, Haihuan Jiang, and Shuqiao Chen. Fuzzy information retrieval based on continuous bag-of-words model. *Symmetry*, 12(2):225, 2020.

[40] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[41] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[42] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[43] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[44] Zein Shaheen, Gerhard Wohlgenannt, and Erwin Filtz. Large scale legal text classification using transformer models. *arXiv preprint arXiv:2010.12871*, 2020.

[45] Sam Shleifer and Alexander M Rush. Pre-trained summarization distillation. *arXiv preprint arXiv:2010.13002*, 2020.

[46] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[47] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[49] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[50] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[51] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[52] Chenyan Xiong, R. Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. *Proceedings of the 26th International Conference on World Wide Web*, 2017.