KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Document Expansion for Swedish Information Retrieval Systems

TOBIAS HAGSTRÖM

Stockholm, Sweden, 2023

Document Expansion for Swedish Information Retrieval Systems

TOBIAS HAGSTRÖM

Degree Project in Computer Science and Engineering
Date: June 28, 2023

Supervisors: Johan Boye, Richard Hartman
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Host company: Findwise
Swedish title: Dokumentexpansion för svenska informationssökningssystem

d|

Abstract

Information retrieval systems have come to change how users interact with computerized systems and locate information. A major challenge when designing these systems is how to handle the vocabulary mismatch problem, i.e. that users, when formulating queries, pick different words than those present in the relevant documents that should be retrieved. With recent advances in artificial intelligence and the emergence of transformer-based language models, new methods have been proposed to alleviate this problem. One such method is the usage of document expansion models which append words to each document that are likely to be part of users' queries. As previous research on document expansion models has been focused on English-language applications, this thesis investigates the effectiveness of one such model for Swedish applications. Although no improvement was found when using this method, the result is likely to be a consequence of dataset quality and domain rather than the method itself.

Keywords

Information retrieval, Natural language processing, Deep learning

Sammanfattning

Informationssökningssystem har förändrat hur användare interagerar med datorsystem och lokaliserar information. En betydande utmaning när dessa system designas är hur det s.k. ”vocabulary mismatch”-problemet ska hanteras, d.v.s. att användare väljer andra söktermer än de som förekommer i de relevanta dokumenten som söksystemet ska hitta. Nya framsteg inom artificiell intelligens och utvecklingen av transformer-baserade språkmodeller har lett till att nya metoder har föreslagits för att mildra det här problemet. En sådan metod är att använda dokumentexpansionsmodeller som lägger till ord till varje dokument som är sannolika att förekomma som söktermer. Då tidigare forskning på dokumentexpansionsmodeller har fokuserat på engelskspråkiga tillämpningar fokuserar det här arbetet i stället på hur väl sådana modeller fungerar för svenskspråkiga tillämpningar. Även om ingen förbättring observerades när denna metod tillämpades är resultatet sannolikt en konsekvens av datamängdens kvalitet och domän snarare än metoden i sig.

Nyckelord

informationssökningssystem, språkteknologi, djupinlärning

Acknowledgments

I want to start by thanking my supervisor Johan Boye and examiner Olov Engwall for providing me with valuable feedback and guidance throughout this project. Thank you Richard Hartman and all my other amazing colleagues at Findwise for supporting and encouraging me, and for welcoming me to your team.

Stockholm, June 2023
Tobias Hagström

Contents

Chapter 1

Introduction

1.1 Background

The internet age has brought an explosion in the amount of information being produced, recorded and published. In 2022, there were an estimated 2 billion websites on the internet [1]. Every minute, more than 500 hours of video are uploaded to Youtube alone [2]. Wikipedia has more than 61 million articles in over 270 different languages [3]. In comparison, the library of Alexandria, considered the epitome of human knowledge at its time, held an estimated 40,000 to 400,000 scrolls, that is between 0.7% and 0.07% of the number of Wikipedia articles [4].

As the amount of available information has grown, so has the need for systems that can index, sort, and search through this enormous amount of data. Having come a long way since 1940s, when the first computer-based information retrieval systems were introduced [5], such systems are today not only ubiquitous, but a necessity for any modern internet user. Unsurprisingly, it is not a newspaper, an encyclopedia or even a social media that is the most-visited website online, but Google Search, which gets over 90 billion visits each month [6].

Effective information retrieval requires systems that can understand the information needs of their users and provide them with relevant documents in response. While parsing a sentence is usually straightforward for a human, building computerized systems that can do so effectively and efficiently is far from it. Recent advances in natural language processing, especially the stunning success of transformer-based language models, has brought new ways to process human language that were previously impossible [7].

1.2 Problem

A way to improve information retrieval effectiveness using language models is document expansion, i.e. to expand documents with words that are likely to be part of users' queries. One method for document expansion is "doc2query", originally proposed by Nogueira *et al.*, in which documents are expanded with predictions of user queries [8]. While the initial results are promising, more research is needed in order to explore the effectiveness and limits to this method so that its usefulness for real-life applications can be evaluated. A limitation of Nogueira *et al.,* 's research is that their most effective method is only evaluated on one dataset, in one language [9]. This paper sets out to explore the effectiveness of doc2query on a different, Swedish-language dataset in order to extend the knowledge base and provide organizations interested in implementing this method in their own systems with additional information on its effectiveness.

1.3 Goals

The goal of this project is to evaluate the effectiveness of document expansion with doc2query in a Swedish language context. This has been divided into the following three sub-goals:

1. Locating or producing suitable datasets in Swedish for model training and system evaluation.

2. Setting up a doc2query-based system for ad-hoc retrieval.

3. Evaluating the effectiveness of doc2query compared with a baseline system.

1.4 Delimitations

This paper will focus on evaluating doc2query in a Swedish language context for ad-hoc retrieval. In addition, methods for producing suitable datasets for training and evaluation will be discussed. The doc2query method itself as described by Nogueira *et al.,* [8] will be adapted to work with Swedish collections but will not be further changed or developed.

1.5 Structure of the Thesis

Chapter 2 presents relevant background information about information retrieval, evaluation methods and language models, in addition to introducing the doc2query method. Chapter 3 presents the methodology and method used to answer the research question. Chapter 4 presents the results, which are further discussed in chapter 5. Finally, chapter 6 presents conclusions and gives suggestions for future work.

Chapter 2

Background

2.1 Information Retrieval

Information retrieval is the academic field of study of developing and applying methods to locate relevant pieces of information from a large set of unstructured data [10]. While a piece of information is usually a textual document coming from a large collection, information retrieval can also be applied to other forms of data, such as collections of images (e.g. Google Images) and sounds (e.g. Shazam). Most, if not all, internet users will have encountered information retrieval systems in the form of search engines.

The information retrieval process starts with a user's information need; we assume that there exists certain information that the user wishes to find. While other forms of information retrieval exist, this article will assume that all information in the system, as well as all user interaction with the system, is text-based.

In ad-hoc retrieval, a typical task, the user will, to the best of their ability, try to communicate their information need to the system by formulating a query. Based on this query, the system will return a list of relevant and/or non-relevant documents. Relevant here is defined as material satisfying the user's explicit or implicit information need [10].

2.1.1 The Information Retrieval Process

When the user enters a query into the system, the query will be transformed into a format that allows the system to match it with documents from the collection in which search is performed. Similarly, the system can index, i.e. transform the document collection in a way that allows for for fast retrieval, in

advance.

Documents consist of tokens, i.e. words. A class of identical tokens is called a type, and after being pre-processed and saved to a system's dictionary, a type becomes a term [10].

A basic information retrieval problem can be formulated as follows: given an unstructured collection of documents d, and a query q, return all documents from d containing any terms present in q. Typically, an inverted index would be used for this purpose. An inverted index is made up of two parts: a dictionary in which terms are stored, and postings which point to documents that contain these. This data structure allows for fast retrieval of documents from a collection [10].

2.1.2 Tokenization

Before constructing an inverted index, documents need to be tokenized (split into tokens). It is also beneficial to conduct linguistic preprocessing on the tokens before they are indexed. Both of these processes are language-specific. While tokenization is straightforward when sentences can be split on whitespace only ("museums in Paris" would become ["museums", "in", "Paris"]), this rule breaks down when whitespaces should be contained in the final terms ("New York" should ideally be one term). Other languages (e.g. Swedish and German) instead benefit from the splitting of compound words, while languages such as Chinese do not use spaces between words and thus call for more advanced tokenization methods [10].

In many situations, we also want two or more different types to map to the same term. For instance, we probably want searches for "jeopardy" to match the type "Jeopardy!", or "usa" to match "U.S.A.". Token normalization commonly includes steps such as case-folding (turning capital letters into lowercase), removing accents and diacritics, stemming, in which suffixes are removed ("cats" becomes "cat", "cities" becomes "citi"), and lemmatization, which uses natural language processing techniques to find the lemma (dictionary form) of each word. User queries entered at search-time should then go through the same pipeline in order to correctly match documents in the index [10].

2.1.3 Ranked Retrieval

Given a large collection of documents, a search can easily return an excess of hundreds or thousands of results. This makes it essential to have effective

ranking algorithms in place, so that the documents that are most likely to be relevant are returned first to the user. Doing this involves scoring each document in terms of its predicted relevance to the user's information need, and sorting the list of documents by their scores. The scoring is done by a ranking function, which takes in a document and a query and returns a numeric score [10].

2.1.4 tf-idf

One widely used statistic for scoring documents is tf-idf, an abbreviation of term frequency–inverse document frequency. The idea behind tf-idf is to prioritize documents which contain more occurrences of the query terms, while putting higher weight on rare words which are likely to be more informative. The first part, term frequency (tf) is a weight, given by the number of occurrences that a query term has in the document. In its simplest form, it is given by counting the number of times a term occurs in a given document. However, this count is often adjusted to account for varying document lengths [10].

Inverse document frequency accounts for the fact that some terms are more informative than others. For example, consider the query "cities in Saskatchewan". While many documents will contain the words "cities" and "in", few will contain "Saskatchewan". Thus, it is more likely that an article containing "Saskatchewan" is relevant to the user, and this term should therefore be given a higher weight than the others. Inverse document frequency, idf, is a weight which accounts for the differing informativeness of different terms. It is given by dividing the total number of documents in the corpus by the number of documents where the term appears, and taking the logarithm of this quotient to moderate the weight [10].

To obtain the tf-idf, simply calculate the product of tf and idf. By doing this for each word in the corpus, we obtain a document vector with a length equal to the number of unique words in the corpus. Note that this bag of words representation is agnostic to the order in which the terms occur [10].

tf-idf can thus be defined as follows [11]:
$tf\text{-}idf_{t,d} = tf_{t,d} * log(\frac{N}{DF_t})$

2.1.5 The Vector Space Model

To score each document in relation to a query, the vector space model can be used. The vector space model incorporates vectors representing different documents in a common vector space, and is widely used for information retrieval tasks. To compute the similarity between a document and a query, we can take the cosine similarity between their respective vectors, given by taking the dot product of the two vectors and dividing by their lengths. It is defined as [10]:

$$\text{sim}\left(d_1, d_2\right) = \frac{\vec{V}\left(d_1\right) \cdot \vec{V}\left(d_2\right)}{\left|\vec{V}\left(d_1\right)\right| \left|\vec{V}\left(d_2\right)\right|}$$

Cosine similarity is the cosine of the angle between the vectors. The denominator normalizes the vectors, so that short documents are given equal priority to longer ones. Higher scores are given to document vectors which are close to the query vector, representing a high degree of similarity between document and query. To rank the search results, each document is scored by its cosine similarly to the query, with high similarity to the query estimated as more relevant [10].

tf-idf has empirically been shown to work well for most applications and was the standard ranking algorithm in Apache's widely used Lucene library (and in turn Elasticsearch and Solr) until 2015 [12]. However, one major shortcoming of this method is its handling of saturated terms (terms occurring many times) in documents, as the tf-idf score increases linearly as the document frequency of a term increases. Intuitively, it is unlikely that a document containing a term twice as many times is twice as relevant to the user. In addition, there is no reward for documents containing disparate query terms, meaning that a document which is over-saturated with one query term can be ranked higher than documents containing all query terms but with lower frequency. To compensate for this, Apache Lucene introduced a coordination factor, which increases the score of documents based on how many query terms that they contain [13].

2.1.6 Probabilistic Information Retrieval

Probabilistic information retrieval was born from applying probability theory to the information retrieval problem, giving this approach a theoretical

foundation that tf-idf lacks. We start by making two assumptions about document relevance [14]:

"1. Relevance is assumed to be a property of the document given information need only, assessable without reference to other documents; and 2. The relevance property is assumed to be binary."

The probability ranking principle, which gives a probabilistic foundation for document ranking, asserts the following [15]:

"If retrieved documents are ordered by decreasing probability of relevance on the data available, then the system's effectiveness is the best that can be obtained for the data."

In other words, we want to rank documents by their $P(relevant|d, q)$, i.e. by their probability of being relevant to the user given a query. We assume that terms are conditionally independent, i.e. $P(ab|c) = P(a|c) * P(b|c)$, and that non-query terms are not affecting relevance. Using probability theory, we can under these two assumptions derive the following scoring function, which will rank documents in the same order as $P(relevant|d, q)$:

$$P(\text{relevant}\,|d, q) \propto_q \sum_{\mathbf{q}, tf_i > 0} w_i,\ w_i := W_i(tf_i)$$

where α_q denotes rank equivalence, tf_i is the frequency of term t_i, and

$$W_i(x) = \log \frac{P\left(TF_i = x \mid \text{relevant}\right) P\left(TF_i = 0 \mid \text{not relevant}\right)}{P\left(TF_i = x \mid \text{not relevant}\right) P\left(TF_i = 0 \mid \text{relevant}\right)}$$

This is the so-called basic model for probabilistic information retrieval [14].

2.1.7 BM25

BM25, originally proposed by Robertson *et al.,* [16] at City University London, was developed from the basic model and further accounts for document length and term eliteness (if a term is elite to a document, it provides useful information about its content). Like the basic model, BM25 assumes document relevance to be binary.

Unlike tf-idf, which increases linearly with term frequency, BM25 approaches an asymptotic limit as the term frequency approaches infinity. This property is one of the major advantages of BM25 as it mirrors common assumptions made regarding term eliteness [14].

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

The BM25 scoring function is summed over the query terms, where N is the number of documents, df_t is the number of documents containing term t, tf_{td} is the term frequency of term t in document d, L_d is the current document length and L_{avg} is the average document length [16].

The formula contains two parameters, k_1 and b, that can be tuned. b is a document-length normalization factor, where $b = 1$ will lead to full normalization and conversely, setting $b = 0$ will turn off normalization. k_1 controls saturation, i.e. how much one term can contribute to the document score. A high k_1 increases the effect that an increasing term frequency has on the document score. Real-world experiments show that values between $0.5 < b < 0.8$ and $1.2 < k_1 < 2$ work fairly well in most situations [14].

2.1.8 Evaluation

In order to determine how effective an information retrieval system is, we need methods for evaluation. The standard way to measure ad-hoc retrieval effectiveness is to use a test collection consisting of documents, information needs, and document relevance judgements for each information need. Relevance judgements are based on whether a document satisfies the described information need, and not whether they contain the terms from the query [11].

Two basic measures of information retrieval effectiveness are precision and recall, defined as [11]:

$$\text{Precision} = \frac{\text{number of relevant items retrieved}}{\text{number of retrieved items}} = P(\text{ relevant } \mid \text{ retrieved })$$

$$\text{Recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items}} = P(\text{ retrieved } \mid \text{ relevant })$$

However, these measures are mainly useful for evaluating unranked retrieval systems since they do not take into account the ordering of the search results. One possible extension is to calculate precision for the first k results. This is useful in situations when the users are assumed to only be interested

in the first page of search results. However, precision at k does not average well over different searches as the total number of relevant documents in the collection will have a strong effect on this measure [11].

Document	Document is Relevant	Number of Relevant Documents Retrieved	Number of Retrieved Documents	Precision
D1	Yes	1	1	1,00
D2	Yes	2	2	1,00
D3	No	2	3	0,67
D4	No	2	4	0,50
D5	Yes	3	5	0,60
D6	Yes	4	6	0,67
D7	Yes	5	7	0,71
D8	Yes	6	8	0,75
D9	No	6	9	0,67

Table 2.1: Example of an average precision calculation.

An alternative measure for ranked retrieval is average precision, which is the mean of the precision at every level of the list of retrieved documents where the document is deemed relevant (highlighted in red). In the example in Table 2.1, the average precision is thereby calculated as $\frac{1}{6}(1.00 + 1.00 + 0.60 + 0.67 + 0.71 + 0.75) = 0.79$. By taking the average precision over several queries, we get the mean average precision (MAP), defined as [11]:

$$\text{MAP}(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\text{Precision}\left(R_{jk}\right)$$

$\{d_1, \ldots d_{m_j}\}$ is the set of relevant documents for an information need $q_j \in Q$ and R_{jk} is the set of ranked retrieval results from the top result to document d_k.

A similar measure is the reciprocal rank which only considers the position of the first retrieved relevant document. If the relevant retrieved document is in position one, the reciprocal rank is 1.0, if in position two it is 0.5 etc. Observe that the reciprocal rank is the same as average precision in the case of only one relevant document being retrieved. By taking the mean of the reciprocal rank of several queries, we get the mean reciprocal rank (MRR) [17], defined as:

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

where Q denotes the information needs in the test set [10].

2.2 Document Expansion by Query Prediction

Doc2query is a method proposed by Nogueira *et al.,* [8] to mitigate the vocabulary mismatch problem - that users use different words in their queries than the words used in documents that would satisfy their information needs. As most information retrieval systems rely on matching terms between queries and documents rather than considering the semantics of each word, this is a problem which can adversely impact their effectiveness. For example, consider the document "London is the capital and largest city of England and the United Kingdom, with a population of just under 9 million" and the query "how many people live in London". While "a population of" and "how many people live in" refer to the same concept, a system purely concerned with lexical matching would not account for this. As the document only has one word in common with the query, it is likely that other, possibly less relevant documents would be ranked higher. For instance, the document "What is it like to live in London?" would receive a higher score than the first document by both the tf-idf and the BM25 function. More examples of query predictions, generated by a doc2query model from [9], are shown in Table 2.2.

Document	**Prediction**
The presence of communication amid scientific minds was equally important to the success of the Manhattan Project as scientific intellect was. The only cloud hanging over the impressive achievement of the atomic researchers and engineers is what their success truly meant; hundreds of thousands of innocent lives obliterated.	what was important to the success of the manhattan project
Phloem is a conductive (or vascular) tissue found in plants. Phloem carries the products of photosynthesis (sucrose and glucose) from the leaves to other parts of the plant. The corresponding system that circulates water and minerals from the roots is called the xylem.	where is phloem found

Barley (Hordeum vulgare L.), a member of the grass family, is a major cereal grain. It was one of the first cultivated grains and is now grown widely. Barley grain is a staple in Tibetan cuisine and was eaten widely by peasants in Medieval Europe. Barley has also been used as animal fodder, as a source of fermentable material for beer and certain distilled beverages, and as a component of various health foods.	where does barley come from

Table 2.2: Examples of query predictions.

Nogueira *et al.,* [8]'s method utilizes a sequence-to-sequence language model to generate predictions of user queries for each document in a collection. Each document is then expanded by appending a number (10-40 in Nogueira *et al.,* [9]) of model-generated queries. When searching through the collection, the system will both consider the original documents as well as the expansions. Since not all of the words generated by the model will occur in the original documents, this method could alleviate the vocabulary mismatch problem and lead to better search results [8]. For example, in the example document above, we would wish that the query "how many people live in London" is generated. The system would then give a high rank to this document when a user searches for "how many people live in London" even though "how many people live in" is not part of the original document.

The main advantage of doc2query compared to alternative machine learning approaches to information retrieval is that this technique does not add a significant computational cost at query-time, as inference is done in advance. This is in contrast to, for example, BERT-based passage re-ranking [18]. As the system architecture stays the same as before with the exception of adding document expansion to the pre-processing step, doc2query can easily be integrated in existing information retrieval systems.

The original doc2query model was based on a BERT (transformer based, encoder-only) language model, and fine-tuned on 500k query-document pairs from the MS MARCO ranked retrieval dataset [8]. In a subsequent article, Nogueira *et al.,* [9] replaced the BERT model with a T5 model which improved the effectiveness of their method. Their T5 method was only evaluated on the MS MARCO document retrieval task, which improved MRR@10 from 0.186 to 0.272 compared to their BM25 baseline [9].

2.3 Language Models

As explained in the previous section, doc2query uses a language model to generate query predictions. Language models are, at their foundation, probabilistic descriptions of languages. Simple examples of these are n-gram models, utilizing the Markov assumption that each state is influenced only by the immediately preceding states. Accordingly, a bi(2)-gram model will give the probability of observing a word given the preceding word [19]. Obviously, this assumption is not true for natural languages such as English or Swedish, necessitating a large n in order to model these well. However, as the distance between two related words can be arbitrarily large, such models are by nature limited.

Before the introduction of transformers in 2017, the state of the art in natural language processing was the use of recurrent neural networks. These are variations of artificial neural networks which allow for sequential inputs. Their "memory" means that the current output is influenced not only by the current input, but by prior inputs in the sequence. This property made them useful for natural language processing, as language is dependent on word order [20]. However, recurrent neural networks still struggled with mapping relationships in large text sequences as their "memory" gradually faded. Another issue with the sequential property of recurrent neural networks was that it made them difficult to train as it limited the possibility of parallelization. This prevented them from being trained on very large collections of text [7].

2.3.1 Word Embeddings

Word embeddings are vector-based mathematical representations of lexical semantics (word meaning). In the same way related words tend to occur in the same contexts, related words should have embeddings that are located close to each other in space (see Figure 2.1 for an example [10].

One of the first influential word embedding algorithms was skip-gram with negative sampling, part of the word2vec package [10]. Based on the idea that "a word is characterized by the company it keeps" [21], the skip-gram algorithm learns embeddings from running text in a self-supervised process by modeling the likelihood of words to occur in the same context. The skip-gram embeddings are detailed and complex enough to enable calculations such as Paris - France + Italy = Rome [22]. These embeddings are static, meaning that each word type is represented by a fixed embedding. In the case of transformers, the word embeddings are learned as part of the training

process [23].

Figure 2.1: A 2-d projection of embeddings for selected words and phrases. Inspired by Jurafsky and Martin [10].

2.3.2 Transformers

The field of natural language processing was revolutionized with the introduction of transformers in 2017, developed by researchers at Google [23]. These models replaced the use of sequential inputs with positional encodings and uses an attention mechanism to model dependencies in the input sequence. While the attention mechanism was introduced earlier by Bahdanau *et al.,* [24] (and further developed by Sutskever *et al.,* [25] and Cheng *et al.,* [26] among others) to improve the handling of long-range dependencies by recurrent neural networks, the transformer architecture made it possible to process the entire input in one go, making it possible to train larger models on more data.

Transformers are built with the architecture shown in Figure 2.2.

2.3.3 Attention

Attention is a mechanism which lets the model focus on certain parts of the input at each step. Attention can thereby reveal contextual relationships between different words in the input. Scaled dot product attention, the attention mechanism used in the transformer architecture, works with three parts: query, key and value. The query is the word currently being processed

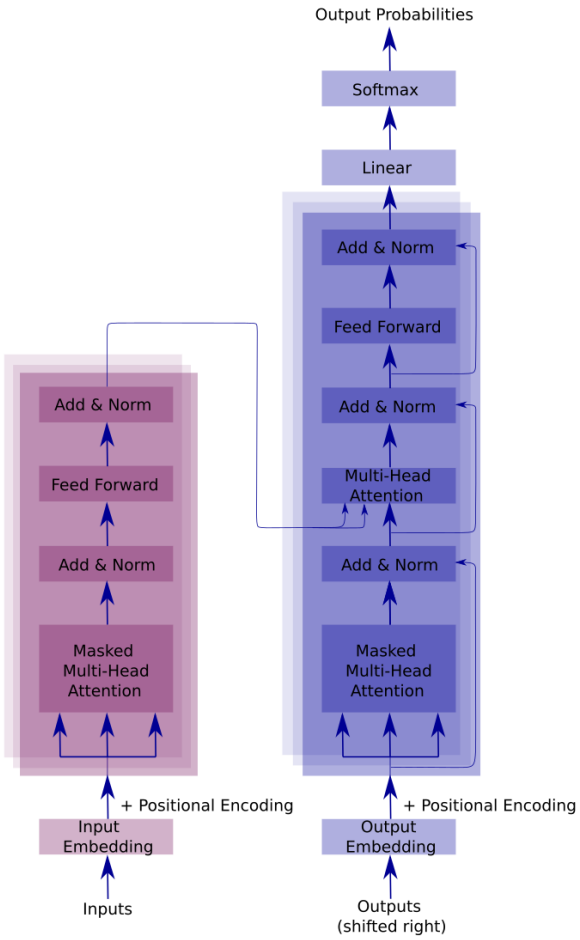

Figure 2.2: Transformer architecture. Inspired by Vaswani *et al.,* [23].

and we wish to find out its relationship to other words in the input. Keys are the words that we compare the query to. Finally, the value is used to calculate the output from the query-key product that indicates how strong the relationship is between each preceding word and the current word (note that "current" is used for explanation, and in practice all of this is computed in parallel) [23].

Queries, keys and values are linear transformations of the input vectors. The next step of the attention calculation is taking the dot products of the queries and keys. Then, we scale the resulting elements by dividing them by the square root of the length of the query/key vector. This is to avoid too large values in the matrix, which would give the softmax function exceedingly small gradients and obstruct model training. In the case of masked attention, used in order to not leak information that the model should predict, all elements in the

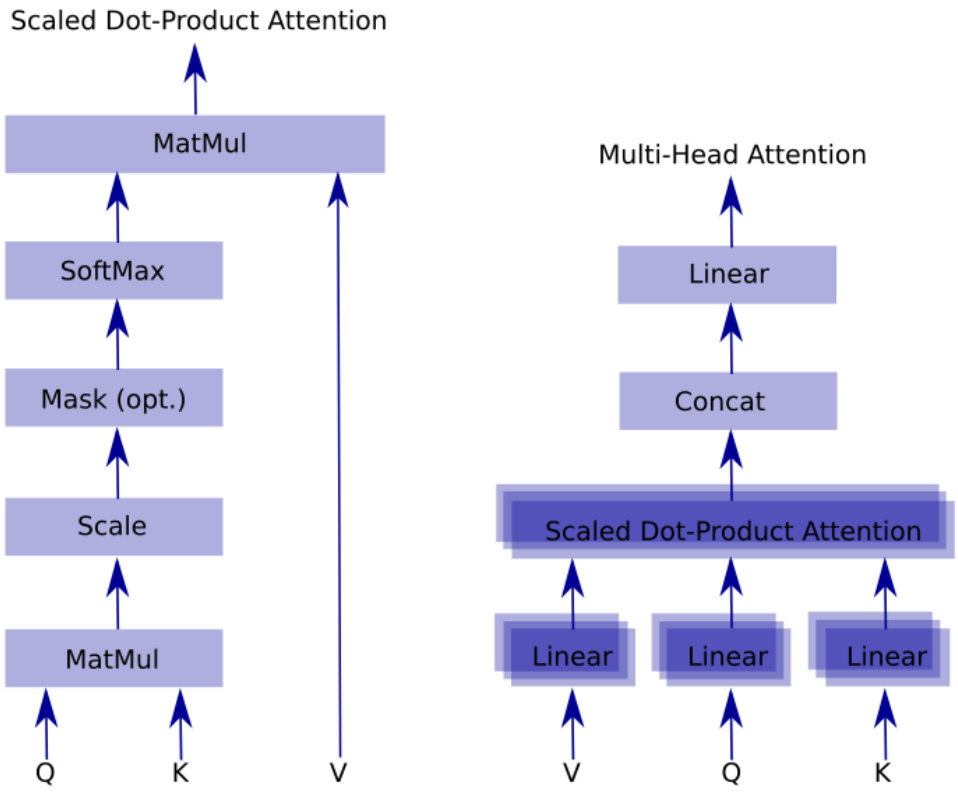

Figure 2.3: Scaled dot-product attention and multi-head attention. Inspired by Vaswani *et al.,* [23].

matrix which denote subsequent words are set to -inf. This allows the model to ”look backward” but not forward in the sequence. Next, we pass the elements to a softmax function to get values between zero and one. The resulting matrix is the attention filter, which for each pair of words describe the strength of the relationship between them. Finally, the attention filter is multiplied with the value matrix [23].

The transformer architecture uses eight parallel attention layers (“heads”) which can attend to different features of the input. While the attention layers use the same architecture, the parameters will differ between the layers. In order to keep matrix dimensions constant, the multi-head attention mechanism adds linear layers before and after the attention layers which project the input and output values [23].

2.3.4 Positional Encoding

Unlike recurrent neural networks, transformers can process all input tokens in parallel [23]. This necessitates encoding the input tokens with positional information as information would otherwise be lost if word order was not preserved. For example, the sentence "Your dog bit my brother" has a very different meaning from "My brother bit your dog".

In the original transformer architecture, positional information is encoded with sine and cosine functions, added to each element in the input vectors.

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$

where pos is the position and i is the dimension [23]

Many subsequent models (including T5) instead use relative position embeddings, introduced by Shaw *et al.,* [27], which consider distances between elements in the input sequence. The self-attention formulas are modified to account for the relative position representations, which are learned parameters of the model.

2.3.5 Encoder/Decoder

The transformer is made up of an encoder and decoder stack, where each stack consists of several identical layers. Each encoder layer contains two sub-layers: the first with multi-head attention as previously described, and the second with a connected feed-forward neural network. To preserve additional information between the sublayers, each is followed by an addition and normalization layer which combines the sublayer's output with output from residual connections. The residual connection simply outputs the sublayer's input, bypassing the sublayer [23]. The function of the encoder is to create a representation of the input.

The decoder layer contains two multi-head attention sublayers - one normal and one masked, in addition to a feed-forward network layer. Inputs to the decoder are the output from the encoder as well as embeddings of the output that the decoder has previously generated [23].

The final piece of the transformer is another linear layer and a softmax layer which provides output probabilities for each word in the vocabulary [23].

While the original transformer architecture is composed of both a encoder and decoder, many successive models [28] [29] [30] have only used the

encoder or decoder part of the architecture.

2.3.6 mT5

mT5 is a multilingual encoder-decoder transformer-based model developed by Google. The model was trained with a "span-corruption" objective, in which parts of text were masked and the model's objective was to complete the sentences with the masked-out words. mT5 was trained with the mC4 dataset, based on cleaned up web scrapes by Common Crawl. This dataset covers over 6.6 billion pages in 101 languages, including Swedish which makes up 1.6% of the dataset. Google has released five different variants of mT5 of varying size, of which the base model has 580 million parameters and the largest "XXL" model has 13 billion parameters. These models can then be further fine-tuned to improve their performance on specific downstream tasks [31].

Chapter 3

Method

3.1 Training and Testing Datasets

Creating a doc2query document expansion model necessitates a suitable dataset for fine-tuning. As we want the model to generate a query given an input document, the training dataset should follow the same structure. That is, it should contain documents with matching queries. Nogueira *et al.,* used the MS MARCO passage ranking dataset for model training, in which queries are matched with short passages ordered by relevance (although Nogueira *et al.,* only retained one passage per query in their training dataset) [9]. MS MARCO was created from real Bing user queries and human-created answers extracted from online sources.

As far as I am aware, there are no similar Swedish datasets of sufficient size publicly available. Therefore, this study uses the same MS MARCO passage ranking dataset as Nogueira *et al.*, but machine translated (using Google's translation service) into Swedish.

For evaluation, a dataset from CLEF (Cross Language Evaluation Forum)'s 2003 competition, compiled by the Swedish Institute of Computer Science, was used. The dataset consists of 142,833 news articles published by TT Nyhetsbyrån during the years 1994-1995. Additionally, it contains 49 information needs and (binary) human relevance judgements for the corresponding articles.

By using training and testing data from two different datasets, this study attempts to answer how well the method performs in a real-life scenario when indexed documents may come in many different types and from different sources, rather than from the same dataset that the model was trained on.

Document	Query
Ackley, Iowa. Ackley is a city in Franklin and Hardin Counties in the U.S. state of Iowa. The population was 1,589 at the 2010 census.	what county is ackley iowa in
Thin cut pork chops should be baked at 350 degrees for about 20 minutes or until the internal temperature is 145 degrees to ensure meat is fully cooked. 2 people found this useful.	how long do you bake thin pork chops in the oven
Aluminium (British and IUPAC spelling) or aluminum (American spelling) is a chemical element with symbol Al and atomic number 13. It is a silvery-white, soft, nonmagnetic, ductile metal in the boron group.	element al
Definition of inherent. : involved in the constitution or essential character of something : belonging by nature or habit : intrinsic risks inherent in the venture.	inherent legal definition
Tax definition, a sum of money demanded by a government for its support or for specific facilities or services, levied upon incomes, property, sales, etc. See more.	what does tax mean

Table 3.1: Sample of MS MARCO items.

3.2 Elasticsearch

Elasticsearch was chosen as the search engine to base the retrieval system on. Based on the Apache Lucene library, it is developed by Elastic BV and freely available under the Server Side Public License [32]. Elasticsearch is widely used in the industry and is currently the most widely used enterprise search engine [33].

Lucene-based search engines use the following, slightly modified version of the BM25 algorithm as their standard scoring function [34]:

$$\sum_{t \in q} \log\left(1 + \frac{N - df_t + 0.5}{df_t + 0.5}\right) \cdot \frac{tf_{td}}{k_1 \cdot \left(1 - b + b \cdot \left(\frac{Ldlossy}{Love}\right)\right) + tf_{td}}$$

Two modifications have been made to Robertson *et al.,* 's original formula. First, a constant (1) has been added to the first component to avoid negative values when $df_t > N/2$. Secondly, the document length, here denoted as L_{dlassy} is rounded to the nearest one byte value, which allows for the pre-

computation of all possible values of $k_1 \cdot (1 - b + b \cdot (L_{dosey}/L_{avg}))$, saving time and resources at search-time [34].

Elasticsearch organizes data into shards, which can hold the index in part or in entirety. Each shard functions internally as its own search engine, which is then presented in aggregated form to the user interacting with the software [35]. It is important to note that Elasticsearch calculates document scores on a per-shard basis by default, meaning that documents will not be ranked identically across different Elasticsearch configurations [36]. In real-world use, this is a reasonable trade-off to make, since the scoring will be approximately correct while the efficiency will be much better. In order to provide consistent scores across shards, Elasticsearch provides a "dfs_query_then_fetch" setting which gathers term frequencies across all shards before computing scores, and was used in this study.

3.3 Selecting a Document Expansion Model

While Nogueira *et al.,* [9]found the T5 model to perform well as a document expansion model after fine-tuning, T5 has only been trained on English data [37] and is therefore less suitable for generating expansions for Swedish documents. Instead, the multilingual mT5 model was used, as this model has been trained on a multitude of languages, including Swedish.

3.4 Fine-Tuning Procedure

The dataset was randomly split into a training and evaluation set containing 90% and 10% of the documents, respectively. The model was fine-tuned for 7 hours after which no further decrease in validation loss was observed, on a virtual machine with one NVIDIA Tesla V100 GPU and 8 Intel Xeon Broadwell E5-2686 v4 vCPUs. Inference was performed on the same machine and took 12 hours to complete. Huggingface Transformers and Pytorch Lightning libraries were used to load the training dataset and for model training and inference.

For performance reasons, the maximum sequence length was limited to 512 tokens meaning that documents over that length were truncated.

3.5 Information Retrieval System Implementation

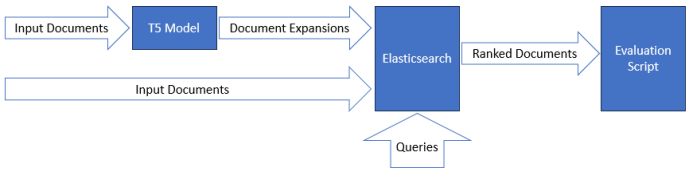

Figure 3.1: System design.

The evaluation system was set up as follows: First, the fine-tuned mT5 model was used to generate 10 document expansions for each document in the test dataset. The test-set documents were then indexed in Elasticsearch along with their respective document expansions. As they were indexed, the documents were separated into title, body and expansion, which allowed search to be performed on requested fields only.

The provided queries from the test collection were used to evaluate the search engine. In order to provide a baseline which to compare doc2query to, a first evaluation was performed using the original documents without expansions. Secondly, the search engine was evaluated again, but now with the expansions included in the retrieval. This allows for comparison between the two results. For each information need, evaluation metrics were calculated with the first 100 ranked results.

3.6 Choice of Evaluation Metrics

Suitable evaluation metrics for this study should take document ranking into consideration, since BM25 with/without document expansion are both methods for ranked retrieval. Therefore, (mean) average precision will be used to compare the rankings, which is widely used in research articles and provides a straightforward measure to compare different models. For a more complete picture, the (mean) reciprocal rank will also be presented.

Chapter 4

Results and Analysis

4.1 Average Precision

Average precision varied widely between different queries. For 12 of the queries, average precision for the system was higher with documents expansion appended. Interestingly, the average precision was higher for the baseline system (no document expansions) for 16 of the queries, and for 21 of the queries average precision was the same across both scenarios. Mean average precision was 0.33 for the document expansion system and marginally higher 0.34 for the baseline system. While average precision was roughly the same for most queries, it was significantly different for others. For example, average precision for query 16 increased from 0.39 to 0.60 when expansion were added. Conversely, for query 29 average precision decreased from 0.72 to 0.36. This would indicate that the document expansions do have an effect on the effectiveness of the system, even if the effect is not always positive. The reciprocal rank paints a similar picture, with the document expansion system receiving a better score for 9 queries versus 10 for the baseline, and the mean being the same across the two.

Query Number	Average Precision (No Exp.)	Average Precision (With Exp.)	Reciprocal Rank (No exp.)	Reciprocal Rank (With exp.)
1	0.28	0.28	1,00	1,00
2	0.33	0.38	1,00	0,50
3	0.82	0.82	1,00	1,00
4	0.36	0.45	0,50	0,50
5	0.05	0.03	0,05	0,03
6	0.00	0.00	0,00	0,00
7	0.52	0.32	1,00	1,00
8	1.00	1.00	1,00	1,00
9	0.27	0.33	0,50	1,00
10	0.29	0.29	1,00	1,00
11	0.91	0.60	1,00	1,00
12	0.60	0.57	1,00	1,00
13	0.11	0.00	0,25	0,00
14	0.07	0.07	0,04	0,00
15	0.00	0.00	0,00	0,00
16	0.39	0.60	1,00	1,00
17	0.22	0.30	0,14	0,50
18	0.49	0.27	1,00	1,00
19	0.89	0.89	1,00	1,00
20	0.00	0.00	0,00	0,00
21	1.00	1.00	1,00	1,00
22	0.17	0.17	0,50	0,50
23	0.01	0.08	0,01	0,14
24	0.15	0.14	0,33	0,33
25	0.00	0.00	0,00	0,00
26	0.37	0.35	0,33	0,33
27	0.02	0.00	0,02	0,00
28	0.39	0.45	0,50	1,00
29	0.72	0.36	1,00	0,50
30	0.00	0.00	0,00	0,00
31	0.01	0.03	0,01	0,03
32	0.06	0.06	0,05	0,04
33	0.78	0.81	1,00	1,00
34	0.45	0.40	1,00	1,00
35	0.25	0.27	1,00	1,00
36	0.00	0.00	0,00	0,00
37	0.03	0.02	0,00	0,02
38	0.21	0.07	0,50	0,08
39	0.31	0.22	0,50	0,17
40	0.92	0.92	1,00	1,00
41	0.02	0.02	0,00	0,01
42	1.00	1.00	1,00	1,00
43	0.07	0.04	0,11	0,05
44	0.25	0.25	0,25	0,25
45	0.04	0.12	0,03	0,14
46	0.64	0.74	0,50	1,00
47	0.00	0.00	0,00	0,00
48	0.56	0.53	1,00	1,00
49	0.82	0.82	1,00	1,00
Mean	**0.34**	**0.33**	**0,51**	**0,51**

Table 4.1: Average Precision and Reciprocal Rank for each information need.

4.2 Analyzing Expansions

Looking further at the document expansions generated by the model, these could be grouped into four categories by whether the expansion is correct or hallucinatory and whether it improves precision or not.

Correct Expansion Improving Precision	**Correct Expansion Not Improving Precision**
Hallucinatory Expansion Improving Precision	**Hallucinatory Expansion Not Improving Precision**

Figure 4.1: Expansion categorization.

While the desired behavior of the model is the generation of correct expansions that improve precision, it was observed that many of the expansions did not improve, or even lower precision. Many of these were hallucinations, queries that the documents did not provide answers to, while others were correct but anyway resulted in irrelevant documents being boosted in the ranking. Interestingly, some hallucinations also resulted in improved precision as they boosted keywords from the documents while the expansions themselves were nonsensical or unanswered in the corresponding documents.

For instance, to an article about the nomination of Erkki Liikanen to the post of Finland's EU commissioner, the model generated queries like ”vem är Finlands kommissionär?” (who is Finland's commissioner?). This was a correct expansion which improved precision. Examples of hallucinations which improved precision were those generated to an article about Sweden being overruled in the EU of whether to label ketchup made from genetically modified tomatoes. Here, some of the queries generated by the model were ”vad är genteknik” (what is genetic engineering?), ”definition av genteknik” (definition of genetic engineering) and ”varför uppfanns genteknik?” (why was genetic engineering invented?), neither was answered by the article and these were thus classified as hallucinations. In fact, 7 out of 10 generated expansions to this document were hallucinations. However, since the model correctly

identified genetic engineering as a keyword in the article, the expansions boosted the article's ranking in regards to an information need about genetic engineering to which this article was relevant.

Even though an expansion is correct, it does not necessarily increase the system's score during evaluation. An example of this is the expansion ”hur många som har gjort en säkerhetsanalys” (how many have done a security analysis), generated for an article about IT security in the Swedish public sector. While this was a correct expansion, all query words were already present in the article and the expansion did not affect the article's ranking. It is however imaginable that this expansion could improve precision given a different query.

Finally, some expansions were hallucinations that did not improve, and often decreased effectiveness. An example of the latter is the expansions for an article about Ukraine transferring nuclear warheads to Russia as part of an arms control treaty, which included queries like ”kärnvapenstrider i Ukraina” (nuclear battles in Ukraine) and ”vad var kärnvapenstriden i Ukraina?” (what was the nuclear battle in Ukraine?). Given an information need about nuclear arms control in Ukraine, these expansion lowered the documents' ranking by several spots and thus decreased precision during evaluation.

Chapter 5

Discussion

5.1 Factors Affecting the Result

Based on previous research, one could have expected to see an improvement when applying document expansion. Instead, the system's effectiveness was roughly the same as the baseline system. There are however several factors which could explain this discrepancy between expectations and results, which will be explained in this section.

5.1.1 Dataset Differences

One of the goals of this thesis was to investigate the effectiveness of the doc2query document expansion model under less-than-ideal conditions in regards to testing and training dataset differences. Much research has been based on the MS MARCO dataset, and although this is a dataset with diverse types of text of high quality, a real-world search engine will inevitably have to index and search through collections of documents which may or may have similar characteristics to MS MARCO. A language model's effectiveness will differ depending on which type of text it is evaluated on, which is especially pronounced in the case of specialized language. Fine-tuning language models on domain-specific data has been shown to result in significantly better performance than using a model trained only with general domain corpora [38]. Therefore, it would be questionable to assume that a document expansion model would be equally effective for any given real-life application as when being tested and trained on text from the same collection. With this in mind, it is not surprising that testing on a different dataset than was used for model training would result in lower effectiveness than when both testing and training

data is from the same collection.

5.1.2 mT5 for Swedish Language Applications

While mT5 is a multilingual model that has been trained on Swedish text, this only represents 1.6% of the training data. Moreover, as this text originates from a web scrape, it is not necessarily of high quality. Dataset size and quality is in turn likely to affect the model's performance on downstream tasks.

Although there exist models trained on high-quality Swedish text only [39][40], there were several reasons to choose the mT5 model for this study. First is the improved performance of T5 compared with a BERT model as described by Nogueira *et al.,* [9]. Secondly, the encoder-decoder architecture used in mT5 was deemed more suitable for this summarization-like task than an encoder- or decoder-only architecture. Thirdly, language models tend to show strong transfer learning capabilities, which is why we can take a pre-trained language model and fine-tune it for a different downstream task than it was originally trained on. It has been shown that language models also show generalization abilities between languages, even for languages that these have not been trained on [41]. Additionally, language models generally benefit from being trained with more data. It is therefore be reasonable to believe that a model trained on a large multilingual dataset could outperform a model trained on a small monolingual dataset.

5.1.3 Fine-Tuning on Machine Translations

A requirement for training any language model is having access to a suitable dataset of sufficient size. While there are plenty of high-quality datasets available in English for a variety of machine learning tasks, the availability of Swedish datasets is extremely limited, especially within the field of information retrieval. Training a document expansion model using the procedure described in this article necessitates the availability of a labeled dataset with documents and related queries. To my knowledge, no such datasets are publicly available in Swedish. Thus, the decision was made to translate an existing dataset from Swedish to English using machine translation methods.

While translation opens up the possibility of using a wide range of high-quality datasets available in English when training models in other languages, it also introduces errors in the training data which can adversely affect model quality. There is currently no consensus among researchers on how to

Table 5.1: Examples of erroneous translations.

Type of Error	**Original Sentence**	**Swedish Translation**
Missing translation	The Triple-A affiliate of the Washington Nationals, the team plays in the International League (IL).	The Triple-A affiliate av Washington Nationals, laget spelar i International League (IL).
Name translation	Fine, New York. Fine is a town in St. Lawrence County, New York, United States.	Bra, New York. Fine är en stad i St. Lawrence County, New York, USA.
Mistranslation	Setbacks along state, provincial, or federal highways may also be set in the laws of the state or province, or the federal government.	Bakslag längs statliga, provinsiella eller federala motorvägar kan också fastställas i lagarna i staten eller provinsen, eller den federala regeringen.
Non-idiomatic language	September sees temperatures that are slightly milder than preceding months in Sorrento, Italy, although it is still extremely warm.	September ser temperaturer som är något mildare än föregående månader i Sorrento, Italien, även om det fortfarande är extremt varmt.
Grammatical	First passed in the World War II era, the Defense Base Act is the federal law requiring workers' compensation coverage for the overseas employees of U.S. government contractors and subcontractors.	Först antogs under andra världskrigets era, Defence Base Act är den federala lagen som kräver arbetskompensation för utomeuropeiska anställda hos amerikanska statliga entreprenörer och underleverantörer.

approach this problem; while some [42] point to positive results when using machine translated datasets for model training, others [43] caution against the use of translated data for training or fine-tuning models for downstream deployment due to a high occurrence of errors in translated datasets.

In this study, although most translations were satisfactory, there were also many errors observed in the translated datasets. These included missing translations, mistranslations, names being inappropriately translated, literal translations/non-idiomatic language and various grammatical errors (some examples are shown in Table 5.1).

5.1.4 Small Testing Dataset

It should be noted that the testing dataset was relatively small, with only 49 information needs. Although this is approximately the same number as the sufficient amount of 50 recommended by Manning *et al.,* [11] for evaluating information retrieval systems, a small testing dataset increases the probability of random variations and makes it hard to draw any conclusion from small differences between results.

5.1.5 Irrelevant Queries and Hallucinations

Hallucination (in the context of language models) is the behavior of generating factual information that is not supported by the input document(s). Hallucinations are commonly categorized as being either intrinsic or extrinsic; intrinsic hallucinations misrepresent factual information from the input documents while extrinsic hallucinations add information (which may or may not be factually correct) that is not present in the input. Tolerance for hallucinations differ between applications: in open-domain dialogue generation some amount of hallucination might be acceptable or even desirable, while, for example, a model summarizing medical records could be very dangerous if such behavior is displayed. Generally, hallucination is seen as a major concern for deep learning models, affecting both their safety and performance [44].

Not unexpectedly, doc2query document expansion models also display hallucinatory behavior by generating queries that are irrelevant or cannot be answered by the document. The consequence is that both retrieval effectiveness and efficiency of the system are lowered, as documents are expanded with misleading information that distort the rankings and increase index size. Gospodinov *et al.,* [45] showed that removing up to 70% of generated queries increased the effectiveness of their search engine with 16% (albeit with a significantly larger number of generated expansions per document of n=80, compared to n=10 used here and in Nogueria *et al.,* 's original paper [8]). Hallucinations by the document expansion model were observed in this study as well, and is a factor likely decreasing its performance.

5.2 Ethical Considerations

5.2.1 Environmental Sustainability

Machine learning models, and large language language models in particular, are a small but growing indirect cause of greenhouse gas emissions. Training and inference require significant computational resources, usually in the form of GPUs or other power-hungry hardware. Although the exact number is subject to debate, the share of global greenhouse gas emissions caused by the ICT (Information Communication Technology) sector is estimated to be between 1.8-3.9% [46]. Without significant efforts to reduce the carbon intensity of the electricity grid used to power servers and other hardware, this share is only likely to increase with the increasing adoption of AI technology.

The use of language models in information retrieval systems increases the computational cost of indexing and retrieving documents, and can therefore be considered to have a negative climate impact. However, one of the major advantages of document expansion is that the computational cost at query-time is not significantly impacted, possibly making this a greener option than other language model-based techniques such as indexing document embeddings directly. Further, it is reasonable to believe that future advances within the field of natural language processing will yield more efficient models and fine-tuning procedures, allowing the method discussed in this paper to be utilized in a less computationally expensive and more environmentally friendly manner. In conducting work for this thesis, I exclusively used cloud providers with renewable energy initiatives, meaning that 95-100% of the electricity used for powering datacenters came from renewable sources. The estimated carbon footprint of model training and inference was 2.11 kg. These emissions have been wholly offset through Vi-skogen.

5.2.2 Stereotypical Bias in Language Models

As large language models (such as mT5 used in this paper) are trained on large text collections, mainly gathered from online sources, they are known to reflect stereotypical biases present in their training data. Examples of this can include biases concerning gender, ethnicity and religion [47].

In the context of document expansion, it is not unlikely that the model would be prone to generating biased queries to input documents. This could in turn affect the ranking of search results in a way that reflects stereotypical biases.

As language models are prone to such unwanted behavior, it is crucial to carefully study their tendency to display biases, and if necessary, deploy mitigative measures (or if even reconsider their use) before deploying them to production environments.

Chapter 6

Conclusions and Future work

6.1 Conclusions

While this study did not show any improvement by using document expansion for Swedish document collections, there are several conclusions that can be drawn from these results.

Firstly, it is important to consider both the quality and domain of any dataset used for model fine-tuning. It is not obvious whether a high-quality dataset that has been automatically translated into another language is of sufficient quality for fine-tuning models for downstream tasks, and previous research has yielded mixed results in regards to this question. The results of this study would advise against using translated data for fine-tuning, although machine translation methods are likely to improve further in the years ahead. Additionally, the use of a multilingual model which mostly has been trained on text in other languages might have compounded the effect of using a translated dataset for training.

Secondly, these results highlight the need to evaluate new methods in the real-world systems that they will be implemented in, and not just rely on previous research. As language models' performance is affected by the domain they operate in, it might be very different when implemented in new applications. Ideally, language models should be fine-tuned on datasets that are representative of how the data they will be used on. Although often costly, producing suitable datasets and fine-tuning models can lead to significant performance increases compared to using an off-the-shelf general model.

Thirdly, this thesis highlights the need for more research within the field of information retrieval on non-English languages, and especially languages with relatively few speakers. In addition, there is a need for models that perform

equally well in Swedish or Swahili as they do in English or Chinese. Likewise, suitable large-scale and high-quality datasets are needed in order to allow for this progress to be made. As AI starts to rapidly change our world, it is of utmost importance that progress reaches all countries and peoples, and that less spoken languages do not get left behind.

6.2 Future work

There are many avenues for further research about document expansion techniques in Swedish. While this thesis attempts to bring some new knowledge to this field, there are many possible improvements and extensions to this work that could not be realized due to limitations in resources and time.

One possible extension would be to try other models than mT5; especially interesting would be models developed specifically for Swedish use. However, there exist very few of these, and have other properties explained earlier that could make them less suitable for this purpose.

It would also be of interest to train and evaluate the model on a variety of different datasets: for example, how would the model perform when evaluated on a translated MS MARCO dataset (i.e. coming from the same collection as its training data), or trained with news articles? This would give clues to whether the low effectiveness shown in this paper is mainly due to the different datasets used for training/testing or to low-quality training data.

Finally, one alternative method for document expansion proposed by MacAvaney *et al.,* [48] models the importance of terms and appends individual tokens instead of whole queries to the documents. This approach has shown to achieve better effectiveness than doc2query and could allow for training on other types of datasets.

References

[1] J. Fedewa, "How many websites are on the internet?" Nov 2022. [Online]. Available: https://www.howtogeek.com/845500/how-many-websites-are-on-the-internet/ [Page 1.]

[2] [Online]. Available: https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/ [Page 1.]

[3] May 2023, page Version ID: 1152561319. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Size_of_Wikipedia&oldid=1152561319 [Page 1.]

[4] W. A. Wiegand, *Encyclopedia of Library History*. Taylor and Francis, 2015. ISBN 9781322878713 [Page 1.]

[5] M. Sanderson and W. B. Croft, "The history of information retrieval research," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1444–1451, 2012. doi: 10.1109/JPROC.2012.2189916 [Page 1.]

[6] [Online]. Available: https://www.statista.com/statistics/1201880/most-visited-websites-worldwide/ [Page 1.]

[7] J. Lin, R. Nogueira, and A. Yates, *Pretrained Transformers for Text Ranking: BERT and Beyond*, ser. Synthesis Lectures on Human Language Technologies. Cham: Springer International Publishing, 2022. ISBN 9783031010538. [Online]. Available: https://link.springer.com/10.1007/978-3-031-02181-7 [Pages 1 and 14.]

[8] R. Nogueira, W. Yang, J. Lin, and K. Cho, "Document Expansion by Query Prediction," *arXiv e-prints*, p. arXiv:1904.08375, Apr. 2019. doi: 10.48550/arXiv.1904.08375 [Pages 2, 12, 13, and 32.]

[9] R. Nogueira and J. Lin, "From doc2query to doctttttquery," *Online preprint*, vol. 6, 2019. [Pages 2, 12, 13, 21, 23, and 30.]

[10] D. Jurafsky and J. Martin, "Speech and language processing (3rd ed. draft)," Jun. 1991. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ [Pages 5, 6, 7, 8, 11, 14, and 15.]

[11] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. New York: Cambridge University Press, 2008. ISBN 9780521865715 [Pages 7, 10, 11, and 32.]

[12] D. Turnbull, "Bm25 the next generation of lucene relevance," Oct 2015. [Online]. Available: https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/ [Page 8.]

[13] [Online]. Available: https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/Similarity.html [Page 8.]

[14] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, p. 333–389, 2009. doi: 10.1561/1500000019. [Online]. Available: http://www.nowpublishers.com/article/Details/INR-019 [Pages 9 and 10.]

[15] K. Sparck Jones, S. Walker, and S. Robertson, "A probabilistic model of information retrieval: development and comparative experiments: Part 1," *Information Processing Management*, vol. 36, no. 6, pp. 779–808, 2000. doi: https://doi.org/10.1016/S0306-4573(00)00015-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457300000157 [Page 9.]

[16] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at trec-3," in *Overview of the Third Text REtrieval Conference (TREC-3)*. Gaithersburg, MD: NIST, January 1995, pp. 109–126. [Online]. Available: https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/ [Pages 9 and 10.]

[17] N. Craswell, *Mean Reciprocal Rank*. Boston, MA: Springer US, 2009, pp. 1703–1703. ISBN 978-0-387-39940-9. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_488 [Page 11.]

[18] R. Nogueira and K. Cho, "Passage re-ranking with bert," 2020. [Page 13.]

[19] M. J. Hofmann, C. Biemann, and S. Remus, "10 - benchmarking n-grams, topic models and recurrent neural networks by cloze completions, eegs and eye movements," in *Cognitive Approach to Natural Language Processing*, B. Sharp, F. Sèdes, and W. Lubaszewski, Eds. Elsevier, 2017, pp. 197–215. ISBN 978-1-78548-253-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978178548253350010X [Page 14.]

[20] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," 08 2013. doi: 10.13140/2.1.2755.3285 [Page 14.]

[21] J. R. Firth, "A synopsis of linguistic theory, 1930–1955," in *Studies in Linguistic Analysis*. Oxford: Blackwell, 1962, {:original-date:1957}. [Page 14.]

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv e-prints*, p. arXiv:1301.3781, Jan. 2013. doi: 10.48550/arXiv.1301.3781 [Page 14.]

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf [Pages 15, 16, 17, and 18.]

[24] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016. [Page 15.]

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 3104–3112. [Page 15.]

[26] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," 2016. [Page 15.]

[27] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," 2018. [Page 18.]

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, p. arXiv:1810.04805, Oct. 2018. doi: 10.48550/arXiv.1810.04805 [Page 18.]

[29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," *arXiv e-prints*, p. arXiv:2005.14165, May 2020. doi: 10.48550/arXiv.2005.14165 [Page 18.]

[30] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. Sankaranarayana Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "PaLM: Scaling Language Modeling with Pathways," *arXiv e-prints*, p. arXiv:2204.02311, Apr. 2022. doi: 10.48550/arXiv.2204.02311 [Page 18.]

[31] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021. doi: 10.18653/v1/2021.naacl-main.41 pp. 483–498. [Online]. Available: https://aclanthology.org/2021.naacl-main.41 [Page 19.]

[32] [Online]. Available: https://www.elastic.co/what-is/elasticsearch [Page 22.]

[33] [Online]. Available: https://db-engines.com/en/ranking/search+engine [Page 22.]

[34] C. Kamphuis, A. P. de Vries, L. Boytsov, and J. Lin, "Which bm25 do you mean? a large-scale reproducibility study of scoring variants," in *Advances in Information Retrieval*, J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, Eds. Cham: Springer International Publishing, 2020. ISBN 978-3-030-45442-5 pp. 28–34. [Pages 22 and 23.]

[35] Dec 2022. [Online]. Available: https://www.elastic.co/blog/how-many-shards-should-i-have-in-my-elasticsearch-cluster [Page 23.]

[36] Apr 2018. [Online]. Available: https://www.elastic.co/blog/practical-bm25-part-1-how-shards-affect-relevance-scoring-in-elasticsearch [Page 23.]

[37] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2020. [Page 23.]

[38] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," 2020. [Page 29.]

[39] Jul 2020. [Online]. Available: https://github.com/af-ai-center/SweBERT [Page 30.]

[40] M. Malmsten, L. Börjeson, and C. Haffenden, "Playing with words at the national library of sweden – making a swedish bert," 2020. [Page 30.]

[41] M. Artetxe, S. Ruder, and D. Yogatama, "On the cross-lingual transferability of monolingual representations," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.421. [Online]. Available: https://doi.org/10.18653%2Fv1%2F2020.acl-main.421 [Page 30.]

[42] L. Bonifacio, V. Jeronymo, H. Q. Abonizio, I. Campiotti, M. Fadaee, R. Lotufo, and R. Nogueira, "mmarco: A multilingual version of the ms marco passage ranking dataset," 2022. [Page 31.]

[43] T. Isbister and M. Sahlgren, "Why not simply translate? a first swedish evaluation benchmark for semantic similarity," 2020. [Page 31.]

[44] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, mar 2023. doi: 10.1145/3571730. [Online]. Available: https://doi.org/10.1145%2F3571730 [Page 32.]

[45] M. Gospodinov, S. MacAvaney, and C. Macdonald, "Doc2query–: When less is more," 2023. [Page 32.]

[46] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair, and A. Friday, "The real climate and transformative impact of ict: A critique of estimates, trends, and regulations," *Patterns*, vol. 2, no. 9, p. 100340, 2021. doi: https://doi.org/10.1016/j.patter.2021.100340. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389921001884 [Page 33.]

[47] M. Nadeem, A. Bethke, and S. Reddy, "Stereoset: Measuring stereotypical bias in pretrained language models," 2020. [Page 33.]

[48] S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, N. Goharian, and O. Frieder, "Expansion via prediction of importance with contextualization," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, jul 2020. doi: 10.1145/3397271.3401262. [Online]. Available: https://doi.org/10.1145%2F3397271.3401262 [Page 36.]

TRITA-EECS-EX-2023:479