
Radboud Universiteit

---

School of Psychology and Artificial Intelligence

Bachelor Thesis

# An Application Of Word Embeddings In Recommending Alternative Query Terms In Domain-Specific Search

*author:* Sander Moonemans, s4608364
*supervisor:* dr. F.A. Grootjen

June 2019

**Abstract**

**An Application Of Word Embeddings In Recommending Alternative Query Terms In Domain-Specific Search**

by
Sander Moonemans

Radboud University
School of Psychology and Artificial Intelligence
BSc in Artificial Intelligence
Supervisor: dr. F.A. Grootjen
June 2019

A Query is a statement of a requester specifying their information need. A poorly formed query can be ambiguous, which can lead to poor performance of an information retrieval machine. Aiding the user by suggesting different query terms could be of use in avoiding this problem. A common way of finding query recommendations is by using query logs (Baeza-Yates et al.). However, smaller companies and institutions that operate in a specific domain rarely possess such query logs as they require large user-bases. Instead of using logs, one could use a language model to find semantically similar terms to the input. A popular example of such a model is a word embeddings (Word2Vec, Mikolov et al.) model. This technique uses a neural network to encode word features to real vectors based on neighboring words in texts of a corpus. These vectors can be compared, so similar words to an input can be extracted. This research proposes a system that can recommend words based on single query terms provided by a user. This system functions as an add-on to an existing domain-specific search engine. A model was trained as part of this thesis and its quality was evaluated. Furthermore, the model was used in a recommendation system and subsequently experimented with. No significant evidence was found regarding a performance gain in this thesis. Improvements are proposed that could potentially lead to a significant result in the future.

# Contents

# 1 Introduction

## 1.1 Background

A Query is a statement of a requester specifying their information need. Querying is an essential part in the exchange of information between humans and machines, but translating the information need of a user to a machine can be a complicated task. The field of information retrieval (IR) is concerned with tackling this exact problem. The information that needs to be retrieved can take many forms, among which text, images or sounds.

Vast advances have been made in information retrieval in the last decades. Modern IR techniques are very efficient in storing data en retrieving information based on queries. However, the first step in information retrieval is constructing the query. Retrieving a large amount of information rapidly is not useful if the information need of a user does not reside in the information that is retrieved by the machine. Often too many results are retrieved which can lead to overloading the user with information, which in turn lead to decreased information processing (Hwang and Lin (1999)). This means the performance of the machine is dependant on the user's ability to specify their information needs.

A common way of studying the performance of an algorithm or machine in IR is by measuring the precision and recall of a set of retrieved information. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over the total amount of existing relevant instances. A poorly formed query can be ambiguous, which can lead to poor performance of the IR machine. For example, an ambiguous query can be interpreted in different ways. The machine can decide to retrieve information of all these interpretations. This increases the chance that the retrieved information contains the user's information need which increases the recall. However, this also results in a larger amount of retrieved information which can lead to lower precision. In modern search engines, thousands or millions results are displayed, with only a small portion being relevant. The user has to look through this heap of information themselves which is very time consuming. Ideally, an IR system should have a high recall and a high precision, but in practice this seems hard to achieve. A balance must be found between recall and precision in order to get optimal results.

There are many ways to facilitate these trade-offs. In modern search engines search results are ranked based on some relevancy metric which helps the user with finding the most relevant documents, thus reducing the need for high precision. A classic, but outdated, example is Google's PageRank system introduced by Page et al. (1999). Expanding queries with synonyms can lead to a higher recall but a lower average precision. However, combining query expansion with relevancy based ranking reduces this downside.

Another way of tackling ambiguous queries is by recommending alternative input. Recommending queries can help the user define their information needs by displaying related, yet different, ideas. Currently there are several ways of recommending alternative queries. A commonly used technique of finding alternative queries is by the use of a query log. That is, a history of similar searches to the input query performed by the same user or other users. Currently much literature is available on the topic of query suggestion using query logs. Baeza-Yates et al. (2004) presented an algorithm that recommends queries based on the clustering of similar queries and their associated information from query logs. This clustering process is based on term-weight vector representations of queries. These vectors were obtained from the term-weight vectors of the URLs clicked by users in the output of the queries. Baeza-Yates et al. propose that even though semantically similar queries do not always share the same words, they do share the same words that are contained in the documents that are selected by the user.

The disadvantage of using query logs for the purpose of recommending queries is the size of the logs that is required. In domain-specific or enterprise search engines, these logs are not available or lack the necessary information to learn appropriate models (Bhatia et al. (2011)). Instead of looking at the implicit semantic relation of queries based on the documents selected by the user, a recommendation model can be built purely by analyzing the language in a corpus.

Some research has been done on query suggestion in the absence of query logs. Bhatia et al. (2011) have proposed a probabilistic mechanism for generating query suggestions from a corpus without using query logs. They utilized a corpus by extracting candidate sentences. This same corpus was being searched in. Sentences are extracted by taking all N-grams of order 1, 2 and 3. In order to avoid sentences that start or end with a stop-words, if such a sentence is made, the next word will be taken into the sentence as well and the stop-word is skipped. This technique is similar to a skip-gram. When the user starts typing, some words are recommended to finish the query. This framework of query suggestion has shown a statistically significant improvement upon two state-of-the-art baselines. Besides giving recommendations for words coming after a given query, not much research has been done on finding words that are different, yet related to the input query. This could help the user specify their information need as it gives a user different ideas that could fit the information need even better.

## 1.2 Query recommendations and language models

In order to find similar words based on a single query term, the computer needs some understanding of the meaning of the input. This requires a more in-depth look of the language and semantics involved. A field that is concerned with

these kind of problems is the field of Natural Language Processing. Certain NLP techniques allows the computer to learn models of language. An example of such a model is the n-gram model. An n-gram model is a probabilistic language model that is able to predict a word given a history; N previous words. Although n-gram models could be used to finish queries, they are not that useful for recommending different query terms.

In order to obtain meaningful different query terms, the machine needs to have some semantic representation of the words. An influential field of research that tries to quantify and categorize semantic similarity between linguistic items is the field of distributional semantics. In a sense, words can be summarized as a sum of its context. This is know as the 'Distributional Hypothesis' first coined by Harris (1954). The distribution of an element is defined as the sum of all its environments. In a sentence, the distribution of a word is defined as the sum of all its neighboring words. The Distributional Hypothesis states that words with similar distributions tend to have similar meaning. There is a correlation between distributional similarity and meaning similarity, which allows us to use distributional representations to estimate meaning similarity (Sahlgren, 2008). Distributional representations of words are usually in the form of a vector. In recent years machine learning techniques have been used to encode various features of words into these vectors, the most influential architectures established by Bengio et al. (2003) and subsequently Mikolov et al. (2013a).

Neural network based language models (NNLMs) have been proven to outperform N-gram models significantly in predicting words in a sequence (Bengio et al., 2003). The main problem with complex language models is dealing with the curse of dimensionality. Modelling the joint probability distributions between many discrete random variables yields a very large amount of free parameters. For example, modelling the joint probability distribution of N consecutive words in a language with vocabulary size $V$ yields $V^{\mathrm{N}}$ - 1 free parameters. The size of V can grow up over a million in some languages like Korean. This would yield $110000^{10} - 1$ free parameters. In their paper, Bengio et al. propose a way to tackle the problem of dimensionality by using distributed representations of discrete variables. In short, each word in a vocabulary is assigned a distributed word feature vector. This vector is valued in the real domain $\mathbb{R}$. The joint probability function of word sequences are expressed in terms of the feature vectors of the words in the sequence. The last step is simultaneously learning the word feature vectors and the parameters of that probability function. A word feature vector represents the different features of a word and can be projected as a point in a vector space. The number of features of the feature vector is set arbitrarily. Bengio et al. experimented with 30, 60 and 100 features per vector. These numbers are significantly smaller than the vocabulary size that would be used in a naive modelling approach. When the modelling is done the feature vectors can be compared to each other and the similarity between them can be measured. A frequently used method of measuring similarity between vectors is calculating the cosine similarity. The cosine similairty measures the

angle between two vectors. If the angle is small, it means that the vectors are alligned in a similar orientation which means they are similar; the vectors have similar features.

A team of researchers at Google recently introduced a method of achieving this in a more efficient way; Word2Vec. The Google researchers, Mikolov et al. (2013a) expanded upon the idea of neural network based language model by introducing the skip-gram model and the continuous bag of words model (CBOW). The most computational expensive part of the feed-forward NNLM resides in the hidden layer. This layer calculates probability distributions. In these new architectures the hidden layer is removed and the projection matrix is shared between all words. This would mean that the precision of the model would suffer per training sample but it also means that it is possible to use much more training data in the same amount of time. The second major difference between these architectures and the neural network proposed by Bengio et al. is that both past and future words are used for the prediction of words and learning the vectors. The CBOW model tries to predict a current word based on the context, whereas the skip-gram model predicts the context based on the current word. These new architectures allow for reasonable training times on large amounts of data. Because of the ability in training on much more data, the vectors learned by the CBOW en skip-gram architectures actually outperform the original neural network language models in semantic and syntactic language tasks significantly.

Together with their new architectures, Mikolov et al. (2013b) introduced a way to extract common n-grams in texts, which in turn can be used as individual feature vectors. The n-grams, or phrases, are formed based on unigram and bigram counts:

$$score(w_{\mathrm{i}}, w_{\mathrm{j}}) = \frac{count(w_{\mathrm{i}} w_{\mathrm{j}}) - \delta}{count(w_{\mathrm{i}} \times count(w_{\mathrm{j}})}$$

Scores above a certain threshold are kept while other the other bigrams are discarded. This method is usually run multiple times in order to get n-grams, where $n > 2$. In this thesis, this technique is used in order to extract bigrams from a large corpus.

The vectors constructed by these neural networks are called word embeddings. Two embeddings trained on different corpora could have different representation as the embeddings are calculated from the context the words are in. This means an word embeddings could have an added benefit in domain-specific search. If an embedding is trained on a domain-specific corpus, logically the embeddings should have a better understanding of the language used in that domain.

## 1.3 Scope of the thesis

The research question of this thesis is the following: How can word embeddings be used in a recommendation system to increase domain-specific search engine performance? In order to obtain relevant query term recommendations, it is important to first investigate the quality of the word embeddings in relation to the texts that are searched for by users. If the texts are domain-specific it would seem logical to train embeddings on these texts as they would best reflect the semantics that reside in the corpus. This brings us at the first hypothesis: Word embeddings trained on a domain-specific corpus outperform word embeddings trained on a cross-domain corpus in language tasks containing domain-specific language. In order to measure the quality of these word represenations we can use the word embeddings in some language tasks that are quite intuitive to us. In the 'experiment' section these tasks will be outlined, together with an explanation about what the results would tell us about the quality. Once we've established the quality of these representation we can investigate their use and potential influence in the act of querying.

The second hypothesis would be: Using a query recommendation system based on word embeddings significantly improves search engine performance. The most straight-forward way of investigating information retrieval paired with a recommendation system is by performing two sets of tasks; a querying task with the help of a recommendation system and one without help. Now the results can be analyzed, marking them either as a relevant or irrelevant. This allows the calculation of precision and recall, and gives us a good measure of performance. One problem arises: the number of results that are returned is too large. Using a domain-specific search engine provided by the publisher Wolters Kluwer, a simple query can return over ten-thousand results. It is out of the scope of this thesis to have domain-experts label all of these results multiplied by multiple trials. It also means that all documents, around half a million, in the database have to be marked for ALL queries performed in order to calculate recall. What we can do however, is take a look at the first set of results returned and calculate the precision of this set.

## 1.4 Outline

In the following section, the materials, experimental design and the results are discussed. In the 'Materials' section the different models that were used are discussed as well as the user interface that was constructed for the query construction process. The different models include a custom trained domain-specific word embeddings model as well as a pre-trained model trained on a large cross-domain corpus. For the domain-specific model, the data, pre-processing and training process are discussed in detail. Following this section, it is discussed how the materials were used for the experiments that investigate the hypotheses and research question. The results are displayed afterwards and subsequently

discussed. Lastly, conclusions are drawn from the research presented in this thesis.

# 2 Materials

## 2.1 Data

In this thesis two corpora were used for learning the word feature vectors. Both corpora contain a similar number of words, four hundred million versus five hundred million words. The key difference between these two data sets is the contents of the corpus. One data set consists of domain specific texts while the other corpus consists of cross domain texts retrieved from sources like newspapers and books.

### 2.1.1 SoNaR Corpus

The SoNaR: STEVIN Dutch Reference corpus is collected as part of a project of the Centre for Language and Speech Technology at the Radboud University (Oostdijk et al., 2013). This corpus contains standard Dutch and Flemish texts published after 1954. The corpus includes original Dutch texts as well as texts that are translated to Dutch by a professional translator. The contents of the texts cover a wide spectrum of domains and genres. The corpus serves as a general reference for various research in language and language use. The corpus itself was not processed as part of this thesis but a Word2Vec model trained on the corpus was already published by other researchers of the University of Antwerp, Tulkens et al. (2016).

### 2.1.2 Wolters Kluwer Dataset

Wolters Kluwer is the biggest publisher and information service provider of The Netherlands. Wolters Kluwer Legal  Regulatory provides up-to-date information and software tools for legal and fiscal professionals. One of the services they provide is the navigation through a large database of judicial documents. The contents of these documents cover a vast array of topics including court rulings, jurisdiction, legislation and regulations. Wolters Kluwers lend us access to this database and search engine allowing the processing of these documents. The corpus contains roughly five hundred thousand texts, which totals to four hundred million words.

### 2.1.3 Pre-processing

In order to obtain the tokenized input for training a Word2Vec model, some pre-processing was done on the documents. The database consisted mostly of

XML documents which are tagged by hand by the company. The tags were consistent throughout the data set. Two tags containing the most usable text were extracted for further processing. Some of the text snippets are interrupted by 'references' which reference other documents or legislative articles. These references did not contain useful information and were removed from the texts. The most common abbreviations were restored to their full form in order to make tokenization simpler and increase their usability when recommending them using the model. Moreover, digits, special characters and multiple whitespace were removed in order to obtain clean text. Hereafter the text was tokenized. Tokens of length 1 (single letters) were removed. If the text contained less than 5 tokens, the text was excluded as it contains too few tokens for the training process. In total 7 million training texts were collected consisting of four hundred million words. Furthermore, common bigrams were extracted and tokenized.

## 2.2 Modeling procedure

The first modeling choice is whether to use the skip-gram model architecture or the CBOW model. Both models have some upsides and downsides. The CBOW model's training time is very fast. This is due to the amount of computations needed for every input token at the cost of the quality of the vectors of infrequent words. The skip-gram model has to perform a number of calculations equal to the context window, the CBOW model averages the vectors of the context window, which is much faster. In this setting, we have extracted bigrams from the corpus which increases the vocabulary size. This also means some of these bigrams have a relatively low frequency. In order to get the best vector representations it is best to use the skip-gram model. An added benefit of using skip-gram model is that you can apply so-called negative sampling, which speeds up the training process (Mikolov et al., 2013b).

In their paper, Tulkens et al. evaluated unsupervised Dutch word embeddings as a linguistic resource. The quality of the word embedding model depends on the parameters used in the modelling process. The most important parameters are the dimension of the vectors D, the word window size, the minimum word count and the amount of training epochs. The dimension of the vector represent the number of features that will be encoded for every word. This means the dimensions of the vector space model is equal to $D \times V$ where $V$ is the vocabulary size. The window size is the maximum distance between the target word and the predicted word. Tulkens et al. found that optimal results where found with a vector size of 320 and a window size of 5 across all corpora. Tulkens et al. released the word embeddings of the SoNaR database which is used further in this research as the cross-domain embedding.

In order to compare the the SoNaR embeddings and embeddings learned from the Wolters Kluwers dataset, the latter was learned with the same parameters as the former. This allows for good comparisons while controlling important

variables. Both corpora have similar sizes, 400 million vs 500 million, and both embeddings are trained with the same parameters. For the Wolters Kluwer corpus the model was trained using the Word2Vec implementation (Mikolov et al.) from gensim (Řehůřek and Sojka, 2010). The original Word2Vec implementation was written in C by Mikolov et al..The gensim implementation ported the algorithms of the C implementation to Python and extended them with additional functionality and optimization.

## 2.3 Recommendation System Interface

The figure below displays the recommendation system interface as it may appear during use. The user can enter one or multiple query terms. 15 recommendations will be given based on the last word typed. The user may click on one of the recommendations to replace the last query term with the recommended one. When the user is finished constructing the query, the user may click on search, which will open a search engine and copies the query to clipboard.

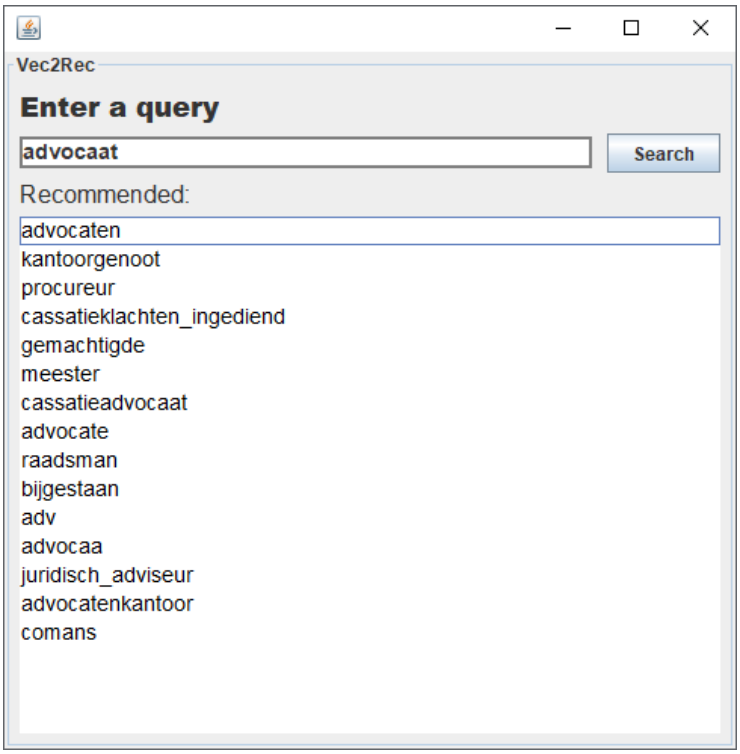



Figure 1: An example of the interface during use, the word 'advocaat' has been entered

# 3 Experimental Design

## 3.1 Inspecting the Quality of Word Embeddings

The first step in answering the research question is by investigating the first hypothesis: Word embeddings trained on a domain-specific corpus outperform word embeddings trained on a cross-domain corpus in language tasks containing domain-specific language. Mikolov et al. themselves proposed a way to measure the quality of a word embeddings model trained with their architectures. They noticed that simple algebraic manipulations could be performed on the vectors to find meaningful syntactic and semantic relationships between words. Multiple degrees of similarity were expressed through the vectors. For example, the model has an 'understanding' of analogies. A basic example would be the analogy: A king is to a man what a queen is to a woman, denoted as King : Man :: Queen : Woman. The model can accurately predict one of the words given the other 3. This is done by performing the following operations: $v_{(\text{King})} - v_{(\text{Man})} + v_{(\text{Woman})}$ this results in a new vector that is closest to $v_{(\text{Queen})}$. This sort of test can give us an intuitive sense of the quality of the semantic representations inside the vectors.

Besides semantic similarity, syntactic similarity can also be measured by applying the same algebraic operations as above on a different setting. Take a verb for example, we can find the conjugations of a verb given another verb with its correct conjugation. This problem can be represented the same way as the analogies: Walking : Walked :: Swimming : Swam. In turn the conjugation of swimming can be calculated as: $v_{(\text{Walking})} - v_{(\text{Walk})} + v_{(\text{Swimming})} \approx v_{(\text{Swam})}$. This principle can be applied to a number of different syntactic settings like opposites, comparatives, adjective to adverb conversion, superlatives and plural nouns or verbs.

Another way of testing the semantic quality of the embeddings is by looking at different kind of language tasks. One of those tasks is the 'odd one out' task, where a subject has to pick a word out of a sequence that is conceptually not related to the other words. For example, given the sequence (dog, cat, bird, giraffe), the odd one out is 'giraffe' because it is not a pet. A word embedding model can be used to solve this by averaging the vectors $v_{(\text{Dog})}, v_{(\text{Cat})}, v_{(\text{Giraffe})}, v_{(\text{bird})}$ which results in a new vector : $v_{(\text{X})}$. We can calculate the cosine similarity of the words in the sequence to $v_{(\text{X})}$ and measure which one is most dissimilar. The order of the words was randomized for all tests.

Of all these tests we can calculate the accuracy of the model over a number of trials which in turns gives an indication of the quality of the semantic and syntactic representations in the word vectors.The second part of the hypothesis has to do with domain-specificity. The language tasks mentioned before can be used in two different settings; a cross-domain setting and a domain-specific

setting. The former are tasks that include very broad and general language like cities, animals or persons. The latter contains language that is less likely to be used outside of a domain. The domain in this case is general and fiscal law. The tests are set-up in such a way that the answers are not obvious and require some expertise in the domain. An example used in the 'odd one out' set (for dutch readers): (staatsrecht, bestuursrecht, fiscaalrecht, burgerrecht), where the first three are part of public law but the last one is not. In this case the 'odd one out' and various syntactic tests were performed using both cross-domain and domain-specific language.

A total of 20 analogy tests, 20 cross-domain and 20 domain-specific 'odd one out' and syntactic tests were performed, for a total of 100 tests, whereof 40 containing domain-specific language. The answer to all test cases were predetermined and only that answer is seen as a correct answer. In order to measure the quality of the embeddings, the accuracy was computed as a performance measure. The accuracy is calculated as the fraction of correct answers over all test cases.

## 3.2 Using the Recommendation System

the second hypothesis was defined as follows: Using a query recommendation system based on word embeddings significantly improves search engine performance. In order to investigate this hypothesis, tests need to be performed in two settings: one where the recommendation system is used and one where it is not used. the best way to test if this system has any added value to domain-specific search is by having a domain-expert naturally query for some information and see if the performance is better while using the recommendation system. In this case 'better performance' is defined as the higher precision in the results of the first page. When the domain-expert chooses to use a recommended word in his/her query, we can compare those results to the results from the original query and measure which of the results has a better precision.The recommendation system simply returns the ten most similar words in the word embeddings when the user is typing the query. The recommendations for a query term pop up after the term is entered. In total 15 words were recommended to the user per query term.

The second thing measured is the number of useful recommendations per query term entered. Every individual query term that was used before was put in the recommendation system again and the useful recommendations were counted.

# 4 Results

## 4.1 Word embeddings quality results

The following table displays the accuracy that was obtained from the tests that were performed with the domain-specific word embeddings and the cross-domain word embeddings. Highlighted are the results from the domain-specific tests and the average over all tests.

| Task | Wolters Kluwer Dataset | SoNaR |
|---|---|---|
| Analogy task | 0.25 | 0.45 |
| odd-one-out(cross-domain) | 0.85 | 0.75 |
| odd-one-out(domain-specific) | **0.85** | **0.80** |
| syntax test(cross-domain) | 0.40 | 0.60 |
| syntax test(domain-specific) | **0.30** | **0.45** |
| Average | **0.53** | **0.61** |

Table 1: Accuracy of semantic and syntactic word embeddings tasks

## 4.2 Search engine performance test results

The following tables outline the precision that was achieved using the domain-specific search engine in the two settings. The first one without using a recommendation system, the second including the use of the recommendation system.

| Query | #Relevant items | #Irrelevant items | Total results | Precision |
|---|---|---|---|---|
| 1 | 9 | 8 | 17 | 0.529 |
| 2 | 6 | 12 | 18 | 0.333 |
| 3 | 8 | 10 | 18 | 0.444 |
| 4 | 12 | 6 | 18 | 0.666 |
| 5 | 2 | 7 | 9 | 0.222 |
| 6 | 0 | 7 | 7 | 0 |
| 7 | 4 | 10 | 14 | 0.286 |
| 8 | 4 | 9 | 13 | 0.308 |
| 9 | 3 | 12 | 15 | 0.200 |
| 10 | 1 | 11 | 12 | 0.083 |
| | | | **Average** | **0.307** |

Table 2: Precision of search engine without using the recommendation system

In 4 situations the domain-expert chose to change their query based on the recommendations that were provided. These queries are highlighted.

| Query | #Relevant items | #Irrelevant items | Total results | Precision |
|---|---|---|---|---|
| 1 | 9 | 8 | 17 | 0.529 |
| **2** | **5** | **11** | **16** | **0.313** |
| 3 | 8 | 10 | 18 | 0.444 |
| 4 | 12 | 6 | 18 | 0.666 |
| **5** | **6** | **6** | **12** | **0.500** |
| **6** | **5** | **7** | **12** | **0.417** |
| **7** | **6** | **8** | **14** | **0.429** |
| 8 | 4 | 9 | 13 | 0.308 |
| 9 | 3 | 12 | 15 | 0.200 |
| 10 | 1 | 11 | 12 | 0.083 |
| | | | **Average** | **0.389** |

Table 3: Precision of search engine in combination with the recommendation system

In order to test for siginificant differences between the two means a one-tailed T-Test was performed. No significant differences in mean were found ($p = 0.606 > 0.05$)

In total the domain-expert found an average of 6,875 query recommendations useful out of fifteen.

# 5 Discussion

## 5.1 Interpretation of results

From the results of the first experiment we can conclude that word embeddings trained on domain-specific data do not yield significant better results in semantic and syntactic language task. The accuracy of the domain-specific embedding is just a little bit better in the odd one out tests than the cross-domain embedding. In this case, the domain-specific word embedding perform poorly on tests that use vector manipulation, the analogy tasks and the syntax tasks, in relation to the cross-domain embeddings. What the results from the odd one out tests mean is that in 85% of the cases, the odd one is indeed selected. With a sequence of length 4, we can assume that baseline performance is equal to 25%. Both embeddings have a much higher accuracy than chance level, but both perform similarly despite of the difference in training data. In the syntax test the cross-domain embedding performs even higher than the domain-specific embedding in domain-specific tasks. Both embeddings perform on a similar level as the embeddings trained originally by Mikolov et al. (2013a) which was around

40%. Because the there is no quality gain in training an embedding on domain-specific data, we can safely conclude that using a cross-domain embedding in domain-specific query recommendation should have no adverse consequences. Rather, Mikolov et al. (2013b) showed that training on more data, billions to tens of billions of words instead of hundreds of millions, significantly improves word embedding quality.

The results from the second experiment are clear, in this experiment we cannot conclude that querying with the recommendation system significantly outperforms querying without such a system. This means the hypothesis is discarded. However, we can see that there is a small performance increase and some improvements to the system will be discussed in the sections below. We can observe however, that a large portion of the recommendations were thought as useful to the user.

## 5.2 Research design

The design of the first experiment was appropriate to investigate the first hypothesis. The experimental setting was controlled, both embeddings had comparable training set sizes and were trained with the same parameters. The most important difference between the two embeddings were the vocabulary sizes. The biggest flaw of this experiment was that the sample size per test was rather low (20 tests per set). However, this still gives a good indication of the quality of both embeddings. Both word embeddings underwent two conditions; cross-domain and domain-specific with varying results.

The second experiment was partially appropriate to investigate the second hypothesis. The first problem with this design was the dependence on an external search engine. The domain-expert noticed that the results on the first page retrieved by the search engine were irrelevant most of the times (precision $\approx 0.307$) without digging deeper into every individual article. Because the results were so unremarkable, investigating the influence of the recommendation system was obstructed. However, the design of the experiment itself could have been effective. A simple comparison of the precision in two settings would give us a good measure of performance difference. In the experiment 10 trials were performed for each setting. Admittedly the amount of trials per setting were low (N=10). Performing more trials would yield a better representation of the performance of the search engine in both settings.

## 5.3 Limitations

An important limitation posed by word embeddings are single word vectors. Often a user queries more than one query term, but as of now, the program can only return recommendations based on the last keyword entered. The semantics of the combination of different query terms, if they are not bigrams, is lost. There is a solution available however, although not perfect, that can

take multiple queries terms into account. It is possible to sum or average two or more word vectors. We can compute the cosine similarity of this vector to other vectors in the model which will return new results. This is a simple way of creating 'sentence embeddings'. The problem with this method however, is that semantics of the individual words get diluted. In practice this means that the recommendations that are given by this new word vector are a mix of the separate word recommendations with barely any new recommendations.

Another limitation of word embeddings are the recommendations given for broad terms and verbs. Terms that are ambiguous or too general return results that are also very general. This result is actually the opposite what this thesis is trying to achieve. A recommendation system that takes a vague query and returns another vague query is not that helpful, as stated by the domain-expert during the experiment. Finding recommendations for verbs usually results in different conjugations of that same verb. For example, for the word 'publiceren', translated as 'to publish', the first six recommendations are: 'publicatie', 'publikatie', 'publiceert', 'gepubliceerd', 'gepubliceerde', 'publicaties'. Only below these words some different words are recommended like 'krant' and 'dagblad', translated as 'newspaper' and 'daily newspaper'. One way of tackling this problem is by lemmatizing these words. However, this means that any syntactic value of the embedding is lost.

## 5.4 Future research

Considering the results from the first experiment, it can be concluded that using a domain-specific source as training data does not yield significant improvements to embedding quality. Rather, Mikolov et al. showed that training on billions of words significantly increases the quality of the embeddings. Future research could include a larger training data set if available. Combining different corpora could pose an easy solution to increase dataset size. In the case outlined in this thesis, it might have been beneficial to train only one embeddings model on both the Wolter Kluwer dataset and the SoNaR corpus, which would double the amount of training data.

Like described previously, there are also some improvements possible to the recommendation system itself. Future research could apply sentence embeddings or even document embeddings, which are extensions of word embeddings, in order to improve search en query recommendations. Queries can be represented as individual vectors based on the individual terms by averaging or summing the corresponding vectors. However, recommending complete sentences based on word embeddings might be a difficult task to achieve on itself.

The biggest problem with the second experiment was the dependence on a pre-built search engine. The domain-expert noticed that the results retrieved by the search-engine were often irrelevant or it was not clear at all what those results were about. In order to fully control the variables concerning search, it

would be recommended to build a stripped-down version of a domain-specific search engine by using open-source search engine platforms like Apache Solr$^{TM}$.

Future research could include a recommendation system that uses word embeddings trained on lemmatized texts in order to get more useful recommendations. Researchers could investigate the importance of syntactic word features of the embeddings in relation to search. If the syntactic value of the embeddings are small in relation to search, this opens a relatively easy way to improve recommendation quality.

# 6 Conclusion

The research proposed in this thesis was about building a recommendation system that could increase domain-specific search engine performance. The research question is: How can a word embeddings model be used as a query recommendation system to increase domain-specific search engine performance?

From this research we can conclude the following: training a word embeddings model on domain-specific texts yield no better quality embeddings in that domain. The second thing we can conclude is that using a word embeddings model in a recommendation system does not significantly increase search engine performance in the way that was tested in the second experiment. Both hypotheses stated in this thesis are rejected. From the second experiment we can conclude that a large portion of the recommendations were useful to the user.

Even though not directly proven in this thesis, the author is confident that with a some mild tweaks to the recommendation system, more training data for the embeddings and a different search engine, performance increases can be expected. Building a recommendation system could pose a useful and inexpensive way in creating a query recommendation system in domain-specific or enterprise search engines. The non-significant outcomes are layed out in the discussion section.

The research question is answered as follows. Word embeddings could be used in a query recommendation system, but not necessarily in the same way as proposed in this thesis. With some adjustments to the recommendation system the author is optimistic about significant improvements to search engine performance.

# References


Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology*, EDBT'04, pages 588–596, Berlin, Heidelberg. Springer-Verlag.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bhatia, S., Majumdar, D., and Mitra, P. (2011). Query suggestions in the absence of query logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 795–804, New York, NY, USA. ACM.

Harris, Z. S. (1954). Distributional structure. *WORD*, 10(2-3):146–162.

Hwang, M. I. and Lin, J. W. (1999). Information dimension, information overload and decision quality. *Journal of Information Science*, 25(3):213–218.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119.

Oostdijk, N., Reynaert, M., Hoste, V., and Schuurman, I. (2013). *The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch*, pages 219–247. Springer Berlin Heidelberg, Berlin, Heidelberg.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54.

Tulkens, S., Emmery, C., and Daelemans, W. (2016). Evaluating unsupervised dutch word embeddings as a linguistic resource. *CoRR*, abs/1607.00225.

# 7 Appendix

## 7.1 Appendix 1: Language Test Sets

### 7.1.1 Analogy Tests

reads as: hij : zoon :: zij dochter Algabraic operation performed:
$v_{(\text{zoon})} - v_{(\text{hij})} + v_{(\text{zij})} \approx v_{(\text{dochter})}$
1. zoon hij zij dochter
2. vader hij zij moeder
3. christendom jezus mohammed islam
4. duitsland berlijn parijs frankrijk
5. nederland amsterdam warschau polen
6. spanje madrid rome italië
7. frankrijk wijn bier duitsland
8. facebook posts tweet twitter
9. hamer spijker haar kam
10. wit zwart boven onder
11. nat droog lang kort
12. nacht maan zon dag
13. potlood schrijven knippen schaar
14. wolven roedel school vissen
15. eten honger moe slapen
16. auto straat zee boot
17. voetbal veld baan tennis
18. benzine auto fornuis gas
19. zoogdier koe slang reptiel
20. mens huid vacht hond

### 7.1.2 Cross-Domain Odd One Out

The last word is the odd one. The order was randomized while testing:
1. aardbei banaan peer hamburger
2. hamburger kip biefstuk broccoli
3. frankrijk duitsland nederland warschau
4. finland noorwegen denemarken spanje
5. rome madrid parijs rotterdam
6. facebook twitter linkedin internet
7. sneeuw regen hagel zon
8. cd floppy dvd laptop
9. varken koe schaap olifant
10. boom struik plant stoepv
11. takken stam wortels gras
12. bier wijn champagne water
13. dokter ziekenhuis verpleegster brandweer
14. jodendom christendom islam monarchie

15. monarchie democratie dictatuur gemeente
16. voetbal hockey tennis knuppel
17. toernooi clubs wedstrijden bal
18. vuurwapen kogel revolver steekwapen
19. paus aartsbisschop priester imam
20. mes keuken koken misdaad

#### 7.1.3 Domain-Specific Odd One Out

The last word is the odd one. The order was randomized while testing:
1. advocaat rechter verdachte kantoor
2. jaarrekening balans resultatenrekening accountant
3. egks eeg eu navo
4. salarisadministratie voorraadadministratie debiteurenadministratie boekhouder
5. statuten bv mededinging secretariaat
6. vennootschapsbelasting inkomstenbelasting btw for
7. wajong ziektewet bijstand salaris
8. omgevingsvergunning wabo bestemmingsplan wethouder
9. adolescentenstrafrecht jeugdstrafrecht volwassenenstrafrecht eigendomsrecht
10. repliek pleidooi bezwaar jurisprudentie
11. ondertoezichtstelling voogdij jeugdzorg strafrecht
12. onderbewindstelling kantonrechter bewindvoerder ondertoezichtstelling
13. cumulatie procesrecht rechtsvordering risico
14. curator failliet schuldeiser jeugdrecht
15. derdenverzet vonnis rechtsmiddel bev
16. gevangenis bewaring gevangenhouding politiebureau
17. personenrecht privaatrecht handelingsbekwaamheid strafrecht
18. publiekrecht wetten vergunningen burgerrecht
19. tekortkoming schuldeiser schuldenaar donatie
20. testament erflater akte schuldenaar

#### 7.1.4 Cross-Domain Syntactic Tests

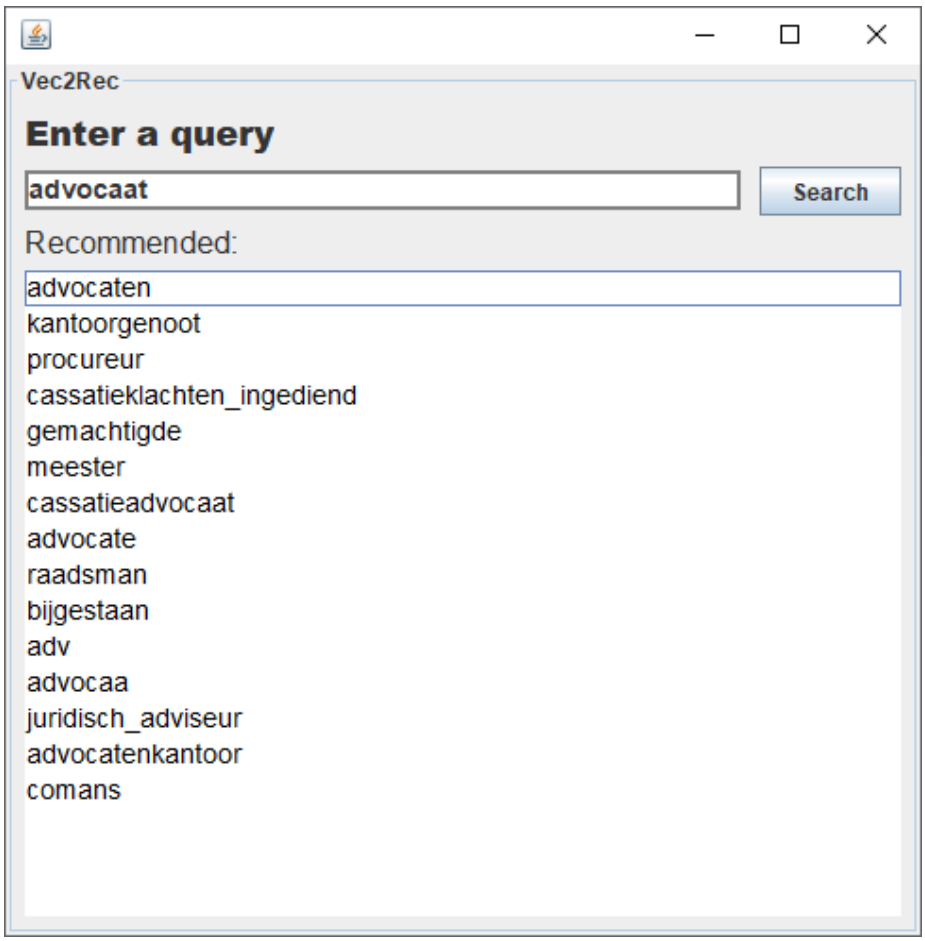
Algabraic operation performed:
$v_{(\text{groene})} - v_{(\text{groen})} + v_{(\text{blauw})} \approx v_{(\text{blauwe})}$
1. groene groen blauw blauwe
2. mooie mooi interessant interessante
3. ronde rond vierkant vierkante
4. grootste groot klein kleinste
5. beste goed veel meeste
6. gespeeld spelen bouwen gebouwd
7. gedacht denken leren geleerd
8. geschreven schrijven typen getypt
9. spreken spreek speel spelen

10. lopen loop spring springen
11. drink drinken eten eet
12. vaar varen rijden rijd
13. nederlands nederland frankrijk frans
14. spaans spanje italië italiaans
15. kippen kip varken varkens
16. tassen tas beker bekers
17. muis muizen tafels tafel
18. bellende bellen schieten schietende
19. invoegende invoegen kruipen kruipende
20. brabant brabantse amsterdamse amsterdam

#### 7.1.5 Domain-Specific Syntactic Tests

1. civiele civiel burgerlijk burgerlijke
2. bewijzen bewezen opgespoord opsporen
3. geschillen geschil behandeling behandelingen
4. meervoudige meervoudig enkelvoudig enkelvoudige
5. overtreden overtredende verzoeken verzoekende
6. verdagen verdaagd uitgesteld uitstellen
7. verleen verlenen plegen pleeg
8. strafzaak strafzaken aanklachten aanklacht
9. gesubsidieerd gesubsidieerde gedaagde gedaagd
10. adviseren geadviseerd vervolgd vervolgen
11. draagkrachtige draagkrachtig financieel financiële
12. voorgeleid voorgeleiding oproep opgeroepen
13. wetsontwerp wetsontwerpen maatregelen maatregel
14. terechtzitting terechtzittingen verslagen verslag
15. verkrijg verkregen begrepen begrijp
16. preventief preventieve mondelinge mondeling
17. gerechtsgebouwen gerechtsgebouw rechtsgeleerde rechtsgeleerden
18. behandelen behandelt veroordeelt veroordelen
19. exploiteren exploitatie administratie administreren
20. executoriaal executoriale conservatoire conservatoir ('executoriaal' was not in the SoNaR embeddings vocabulary.

## 7.2 Appendix 2: Recommendation system GUI




Code can be accessed on: **https://github.com/SanderMoon/QueryRecommendations**

## 7.3 Appendix 3: Instruction Manual For Experiment 2

### Instruction manual

Thank you for agreeing to participate in this study about information retrieval. Your full attention is required for this experiment and an instructor will be present at all times. All results gathered from this experiment will be made anonymous. Read the following instructions carefully. This experiment will require you to use two programs; one is a web-based search engine, the other is program provides a comparable interface as the search engine but with some recommendations provided. The experiment proceeds in three stages:

- An instruction round, in which you will get familiar with the programs. Any questions related to one of the programs can be directed at the instructor.
- The first trial consists of finding information through the use of a provided search engine. You can type one or multiple words in the search bar that is provided. It is not allowed to use the aids provided by the web page. Once the words are typed in you can press the 'zoek' button or the 'ENTER' key on the keyboard. The instructor will ask you to briefly take a look at all results on the first page. Subsequently, the instructor will ask you whether a result is relevant or irrelevant to the information that you were looking for, it is allowed to inspect the links by clicking on them. Pressing the browser return button will bring you back to the results. This task is repeated a number of times.
- The second trial is comparable to the first trial, however, the query is not typed in the same search bar as before. Another program is provided in which you first enter a query. When you are content with the words that you typed in, you can press 'search'. The use of the list of recommended words is permitted by clicking on a word, this will replace the last word you typed in. Pressing Search will open a web page where you now can paste the query by pressing 'CTRL+V' or by clicking the right mouse button on the search bar and press 'Paste'. The instructor will again ask you which of the results are relevant or irrelevant to the information you were looking for. This trial will again be performed multiple times.

Again, thank you for your participation and good luck! Any questions during the experiment can be directed to the instructor. If there are any questions left after the experiment is over, they can be directed to the following e-mail address: **s.moonemans@student.ru.nl**