Date of acceptance Grade

Instructor

Information Retrieval with Finnish Case Law Embeddings

Sami Sarsa

Helsinki August 17, 2019

MSc Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Sami Sarsa

Työn nimi — Arbetets titel — Title

Information Retrieval with Finnish Case Law Embeddings

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
MSc Thesis	August 17, 2019	61 sivua + 3 liitesivua

Tiivistelmä — Referat — Abstract

Information is nowadays abundantly available in human readable and easily accessible form in the Internet. However, the vast amount of text becomes a problem when searching for a specific piece of information and thus, the effectiveness of information retrieval methods carries great importance.

The most common method to retrieve information from a collection of documents is to input a set of keywords as a query [BYRN+99]. The query is then compared against the documents in the collection. An algorithm will rank documents in the collection based on relevance to the query and return the documents to the end user in the order of the ranking. This work studies an alternative to the traditional keyword query: the usage of a whole document as a query. A query consisting of a whole document has an advantage over the common keyword query, since there is no requirement for formulating the keywords.

In this work, document ranking for retrieval is based on the assumption of vector space models: The relevance of a set of retrieved documents to a query is approximately equal to the similarity between the query and documents in the retrieved set [GRG18]. Therefore, this work focuses on semantic textual similarity, a field of ongoing interest [BCA+17] in natural language processing, which has been mostly focused on paragraph, sentence or word similarity instead of full text documents.

To compute numerical inter-document similarities for ranking, texts are mapped into vector space with the use of document embedding models. A widely used word frequency based model TF-IDF [SJ72] and probabilistic model LDA [BNJ02] serve as baseline document embedding models for text similarity computation. More recent, neural network models, inspired by the popular Word2Vec [MCCD13], Doc2Vec [LM14] and Doc2VecC [Che17], are compared to and combined with TF-IDF and LDA in an attempt to improve on the standard methods. A comparison of the document embedding models' performance in capturing the semantics of Finnish case law documents for similarity comparison is provided. A gold standard set of human evaluated document pair similarities is created for evaluating the models. Also, topic classification with keywords is studied as an alternative evaluation method due to arduousness of human labelled similarity acquisition.

In addition, an effective full document query information retrieval system for Finnish case law is created and presented as a part of this work. The case law data set [OTM+17] used in the work consists of ca. 13 000 Finnish case law judgements from 1980 to 2019 and is provided by the Finnish Ministry of Justice.

CCS concepts:

- Information systems: Information retrieval
- Computing methodologies: Natural language processing
- Computing methodologies: Unsupervised learning

Avainsanat — Nyckelord — Keywords
Information Retrieval, Document Embedding, Natural Language Processing, Neural Networks

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Information is nowadays abundantly available in human readable and easily accessible form in the Internet. This excessive amount of information at hand has had a great impact on everyday knowledge acquisition for the people of today's world. However, there exists an inverse for the problem of getting information. It can be extremely difficult to find relevant pieces of information in the World Wide Web due to the ever increasing amount of available information. This phenomenon, known as information overload, infoxication or infobesity [Cer17], has given importance to information retrieval systems that aim to ease the search for relevant data.

Nowadays the most common way to retrieve information is using a Web search engine, such as Google [Chr18] or one of its competitors. The straightforward traditional querying model to retrieve information with search engines comprises of the following steps as described by Baeza-Yates et al [BYRN+99]:

1. Start with an information need.
2. Select a system and collections to search on.
3. Formulate a query.
4. Send the query to the system.
5. Receive the results in the form of information items.
6. Scan, evaluate and interpret the results.
7. Either stop, or reformulate the query and go to step 4.

Although this method is already simple and effective, it can be further simplified in certain cases. Let us suppose that a certain person has a piece of text in his or her hands and simply wants to retrieve similar pieces of text. Additionally, suppose we have an algorithm able to identify the important characteristics in a query document. With these assumptions, the aforementioned query model from Baeza-Yates et al can be simplified by eliminating the need to create and reformulate the query and will be reduced to:

1. Start with an information need.
2. Select a system and collections to search on.
3. Send the query document to the system.
4. Receive the results in the form of information items.
5. Scan, evaluate and interpret the results.

An example use case for the renewed query model is a citizen receiving a sentence and wanting to find similar ones to see whether his or her edict was just. This use case, along with the alleviating relevant judgement search for lawyers and law students, acts as a motivator for this work. We use Finnish case law judgements to create an information retrieval system for Finnish case law optimised for full document queries. Our case law data is provided by The Finnish Ministry of Justice [OTM+17] and consists of ca. 13000 Supreme Court's and Supreme Administrative Court's judgements.

An information retrieval system should be able to rank documents based on relevance to a query for retrieval so that the documents can be reviewed in a meaningful order. Document ranking for retrieval in this work is based on the assumption underlying vector space models: The relevance of a set retrieved documents to a query is approximately equal to similarity between the query and documents in retrieved set [GRG18]. That is to say that document retrieval is optimal when it is based on optimal query-document similarity. To enable direct computation of numerical document similarity values, documents are transformed into vectors using document embedding models. Consequently, the models' proficiency in embedding texts into vector space becomes the greatest factor that determines ranking effectiveness.

In this work, two traditional methods, Term Frequency - Inverse Document Frequency (TF-IDF) and Latent Dirichlet Allocation (LDA), are compared and combined with newer, neural networks based embedding models to improve upon the traditional methods performance in embedding texts for similarity based ranking. While the study of using text embeddings by neural networks has been popular recently in information retrieval [BGG18] and textual semantic similarity computation [MBB+14, CDA+17], the focus has been on short texts, such as keyword queries or sentences, rather than whole documents comprising of possibly numerous long paragraphs. In contrast, our case law data set consists of variously sized documents, the longer ones containing over 10 000 words.

Besides concentrating on finding optimal document embedding models (or a combination of such) for Finnish case law retrieval, we place some emphasis on model performance evaluation. Evaluation of information retrieval models is a difficult task due to challenges in measuring textual relatedness. Preferably, one would have a human assigned ground truth ranking with which to compare models' rankings. Unfortunately, good ground truths for information retrieval are difficult to obtain, wherefore an alternative approach is examined. This alternative approach consists

of classifying documents embeddings to their respective topics. We investigate the differences and challenges between evaluation by topic classification and evaluation using ground truth ranking.

Thus, we concentrate on the following research questions regarding Finnish case law retrieval:
RQ1: How do neural network generated document embeddings perform in ranking compared to traditional document embedding models, TF-IDF and LDA?
RQ2: How can multiple models' document embeddings be leveraged to improve ranking effectiveness from ranking with a single model's embeddings?
RQ3: How does topic classification compare to correlation with ground truth similarities as a method for evaluating document embeddings for ranking?

The majority of this work consists of describing, analysing and evaluating various text embedding models that are able to capture the texts' semantic information for similarity evaluation. This includes describing preprocessing the data for models and the theoretical grounds for evaluating the embedding models as well as evaluation results. In addition, we present a web application for Finnish case law retrieval.

This thesis is organised as follows. It starts with an introduction to theoretical perspective and gradually moves towards more concrete implementations. The background section introduces the theoretical bases for methods used in this work spanning the fields of information retrieval, natural language processing and machine learning. Background is followed by a section dedicated to document embeddings, which explains the models that are studied and used in our web application. After that, our case law data, data preprocessing, model evaluation and similarity computation from embeddings are addressed. Finally, we present evaluation results and conclude with discussion and future work. Our application that utilises the studied models is described in appendix A.

2 Background

The key concepts as well as the domain of this work belong to the fields of information retrieval (IR), natural language processing (NLP) and machine learning (ML). This section introduces key theoretical terms and concepts for this work within the three fields.

2.1 Information Retrieval

Information retrieval (IR) refers to acquiring knowledge from a collection of data and the study of methods that enable retrieval. Retrieved information can be in any format, e.g. text, image or sound, and it is stored in what is called an information retrieval system. An IR System is a system that maintains, stores and is able to retrieve stored data. William Frakes and Ricardo Baeza-Yates describe IR systems [FBY92] as systems that match user queries to data stored in a database. The data in an IR system is often surrogated with metadata to improve query time and storage space. Metadata may consist of keywords, such as the author, title and date, or data specific identifiers, e.g. European Case Law Identifier (ECLI) [EU17].

The creation of IR Systems, information acquisition, organisation, storage, as well as retrieval itself, all fall within the field of information retrieval [Bat12]. Within this work, we concentrate on retrieval, and only in the context of web document retrieval as opposed to e.g. desktop search or question answering. Document retrieval is described by Liu [L$^+$09] as follows: A retrieval system contains a collection of documents. A user asks for documents with a query consisting of words. The system retrieves documents containing the query words from the collection and ranks them chiefly by relevance with respect to the query. This process is illustrated in Figure 1.

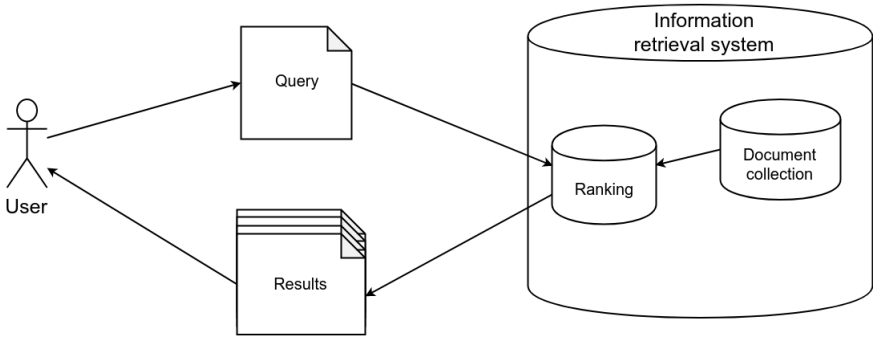

Figure 1: A depiction of document retrieval with an information retrieval system.

2.1.1 Ranking

A key IR concept in this work is *ranking*, which holds a central problem in multiple IR applications [QLXL10]. Ranking, or ranking creation, formally means using a function to create a ranked, i.e. ordered, list of objects. The ordering in the ranked list can be based on various things. For instance, in machine translation, the objects, or translations in this case, could be ranked not only by similarity to query, but also by how likely the translation is to be used in human written text.

In our case law document retrieval setting, we aim for optimal ordering of retrieval documents for a full document query. As explained in the introduction, we base our ranking on similarity of texts using the assumption that relevant documents to a query are somewhat equal to query-document similarity. It is not, however, simple to understand what is optimal ordering by textual similarity, as similarity itself is a complicated concept in natural languages.

2.1.2 Semantic similarity

Semantics, a branch of linguistics, is the study of meaning and relations of words and phrases. In other words, semantics seeks to explain what texts and their components stand for. When evaluating similarity between texts or words, often what is meant by similarity is semantic similarity or semantic relatedness. This is also the case in this work, since we are interested in finding out how close or far away the representations of texts' meanings are from each other by the means of document embedding.

For a more formal definition of semantic similarity, this work uses Harispe et al's definition [HRJM15] for semantic relatedness: "the strength of the semantic interactions between two elements with no restrictions on the types of the semantic links considered". Harispe et al make a distinction between semantic similarity and semantic relatedness, but here, the more broader definition of relatedness is used. The two semantic measures are considered synonyms for simplicity, since these terms are used somewhat interchangeably [MBB+14, ABC+14] in the scientific literature.

Truly understanding, or even defining semantic similarity is a demanding task. Whereas it is easy to say that the word *dog* is more similar to *hound* than *dog* is to *refrigerator*, it is far more difficult to tell whether *cat* is semantically further away from *kitten* than from *lynx*. One could argue that the semantic difference between *cat* - *kitten* is in a different dimension than *cat* - *lynx*. The former being related to age and the latter to biological taxonomy. However, regarding ranking,

the division to dimensions causes complications. The dimensions would have to be first defined and then weighted in order to create a numerical ordering of semantic items. An additional challenge arises from the human notion of similarity not being always symmetric as shown by Tversky [Tve77]. For instance, in one of Tversky's experiments, the similarity between a smaller country was deemed more similar to a large country than the other way around, with examples including *North Vietnam - Red China* and *Mexico - USA*.

Due to the complexity of similarity in natural languages, it would be excessive to dive deep into semantic similarity's cognitive or psychological definitions. Having some grasp on the matter will suffice. The goal in this work is to have automatically generated similarity values that correspond to what humans consider similar, not to fully analyse similarity as a concept.

2.1.3 Document embedding

In order to narrow the gap between machine and human performance in assessing semantic similarity, machines require sophisticated models to process texts. Simple string comparison is hardly likely to give meaningful values for similarity.

Vector Space Model (VSM) is a traditional [SWY75, GRG18] and widely used [BGLB16] family of IR models that provides a framework for computers to understand natural language texts by embedding documents into vector space as bag-of-words (BoW) [Har54] representations. BoW is a straightforward model to represent text in vector form, where a document's word token frequencies serve as embedding vector's elements. That is, a BoW consists of a fixed number of integer or floating point values, where each value represents a frequency of a vocabulary term.

When the documents are embedded into vectors, pairwise similarity between documents can be obtained from the correlation between their vector representations. We refer to these vector representations as *document embeddings*.

For information retrieval, as documents are embedded into vector space, user queries also undergo the same process to allow the usage of vector correlation for ranking. Given a user query, the query text is first converted to a vector. Then, similarities between the query and each document in the IR System are computed. When each query-document pair has been assigned a similarity score, the IR system's document collection can be ranked by the documents' respective similarity to the query. Then, documents in the collection can be delivered to the user who constructed the query to be viewed in a desirable order.

2.1.4 Vector similarity

In VSMs, the typical correlation measure for vectors is the angle between the vectors [S+01]. To compute vector correlation, the cosine of an angle θ between two non-zero vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ may be computed according to eq. (1) in order to obtain a numerical similarity value for the two vectors.

$$\cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||\ ||\boldsymbol{b}||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{1}$$

This method of computing vector correlation for document embeddings, according to Manning et al [MRS08], is the standard way to quantify the documents' similarity for information retrieval and is referred to as *cosine similarity* measure.

Multiple correlation measures for a pair of vectors' similarity exist, including Euclidean distance, Jaccard coefficient, Pearson correlation coefficient and Sørensen–Dice coefficient, among others. According to Huang's experiments [Hua08], the different metrics perform relatively similarly when used for clustering texts, apart from Euclidean distance being evidently the worst. Furthermore, Li and Han show that cosine similarity outperforms [LH13] Jaccard coefficient in several textual classification tasks.

Given a set of already available similarity labels of document pairs for a part, or all, of the text data, there exists another approach to derive similarity values for document pairs besides computing vector correlations. This other method consists of using a machine learning model to learn similarities with the pre-defined values as targets. The approach is called *learning to rank* [L+09], as the similarities are learned instead of computed directly from document representations.

Learning to rank has the potential of leading to better similarity rankings between document embeddings than the cosine similarity, since some vector dimensions might be more relevant than others, and this can be learnt from the vector embeddings. Learning to rank is, however, a double-edged blade. The learning improves as the training set for the learner model becomes larger, yet, if the set of gold standard labels is small, the model's performance is unlikely generalisable leaving a trained model ill suited for practical use.

2.2 Natural Language Processing

Natural language processing (NLP) is a subfield of computer science and artificial intelligence (AI) that can be considered to cover all kinds of natural language manipulation by computers [BKL09]. *Natural language* here denotes any language used by humans for ordinary communication. With this definition of NLP, it largely overlaps with IR. Converting texts into vector form for vector space models, or computing text similarities, can be considered NLP as much as they are a part of IR. These tasks, however, differ by their role in the two fields.

Computing text similarity is an NLP task, whereas in IR, such task is merely a tool for improving the task of document retrieval. In addition, multiple NLP tasks that consider minor manipulations of texts have been proven beneficial for IR. Brants [Bra03] shows that multiple such techniques can be used to improve document retrieval, but also that it is important to optimise NLP for IR.

For NLP methods considered in this work, besides text vectorisation and similarity computation, we study the most effective NLP tasks for IR in Brants's study: normalising word forms and filtering out words that may mislead algorithms.

2.2.1 Text normalisation

Document embedding models accept certain kinds of input which may differ depending on the model. In order to embed texts into vector space, the texts need to be converted into a format which is accepted by the embedding model. This falls within the NLP task of *text normalisation*, which according to Martin et al [MJ09] is defined broadly as converting text to a convenient standard form.

The aforementioned BoW model consists of word frequencies, but to create BoWs from text, one needs to separate *words* from text first. That is to say that it is required to define what counts as a word in a text. This task is called *tokenisation*, which can be described more formally as separating words in text into tokens. For languages where words are separated by whitespace, which includes Finnish, tokenisation can be performed relatively simply by extracting pieces of text between spaces, tabs and newline characters as separate tokens. This would mean that numbers, for instance “2”, “1998” or “3.14”, and also symbols and punctuation, e.g. “&” and “!” would be considered words if between whitespace characters. Punctuation, however, appears rarely on its own but rather connected to a word. Thus, it is often desirable to handle punctuation when tokenising to remove the distinction between

tokens such as “cat” and “cat.”. This can be done e.g. by removing punctuation marks entirely or including them as separate tokens. The usefulness of including punctuation is dependent on the task a text is used in.

While a set of tokenised words is enough to create a BoW, as the BoW is merely a set of token counts, *word normalisation*, a subtask of text normalisation that considers converting words into a standard form, can help make the BoWs more meaningful. For instance, words in text can be abbreviated, capitalised or inflected. Whereas capitalisation can be conveniently handled by converting a whole text into lower or upper case, and abbreviations unabbreviated via regular expressions or a list of known abbreviations, inflections are more difficult to normalise.

The study of words forms falls under morphology, which is a field in linguistics. Inflection, in morphology, stands for a word formation process, where a word is modified to express e.g. a different tense or plurality. Inflected word forms carry various additional meanings to a word’s base form, while the core meaning of the word remains the same as in its base form. Since our dataset is in Finnish, which is morphologically rich, that is, highly inflected, normalising different word forms affects the data considerably.

For similarity computation methods that consider any distinct combination of word characters a different one, such as any BoW based method, morphological variations of a word would be considered as completely different words from the original. Words such as “dog”, “dogs” and “dog’s” would be all counted as separate words and the only way to infer their similarity would be to consider the context for these words. Even an algorithm that has access to word context is incapable of inferring that obviously similar words convey the same meaning, if the words don’t appear enough in similar context.

The simplest way to normalise word forms, in order to make a machine understand that a set of morphologically different words carry the same meaning, is to use heuristics for cutting off the end or beginning of a word. In the optimal case, the resulting word will be reduced to its stem, i.e. the part of the word to which affixes can be added. Thus this word cutting process is, rather optimistically, called stemming. From the previous example, “dogs” and “dog’s” would be reduced to “dog”, and the three variants of the word “dog” would now be considered as the same word. However, stemming suffers from considering only the removal of prefixes and suffixes. It does not work well for irregular forms, since e.g. reducing the word “am” and “are” to “a” could be more harmful than leaving them be.

Another approach, which takes into account irregular forms, is to utilise dictionaries and morphological analysis to derive the base form of the word. In morphology, the base, or dictionary form of a word is called *lemma*, wherefore this approach is referred to as lemmatisation. With dictionary lookups, lemmatisation can overcome problems with irregular inflecting which stemming cannot. Additionally, lemmatisation is proven to work better than stemming for Finnish as Korenius et al [KLJJ04] finds lemmatisation superior to stemming for Finnish while evaluating the two word normalisation methods for document clustering. Lemmatisation, however, is difficult to perform perfectly as inflected forms can be ambiguous. For instance, the Finnish word "voi" (butter) has the same written form as the third person singular form of verb "voida" (to be able to).

Besides normalising word forms, the *vocabulary*, i.e. the set of used words, of an NLP task can be normalised by excluding some words from the vocabulary. A tempting group of words to exclude is stopwords, that is, the most common words in a language that do not carry any significance on their own [Ull11]. They include words such as "and", "for", "or" and "to". Unsurprisingly, information retrieval has a long history of removing stopwords from texts and queries before processing them for document retrieval [Bra03]. Without stopword removal, a model might apply excess weight on insignificant words, which would lead to suboptimal ranking. Removing stopwords, as well as punctuation, also reduces the size of an information retrieval system as less data has to be stored online.

Removing stopwords can be done using list of unwanted common words and filtering them out from tokenised text. The stopwords are often extracted from text by picking the most frequent words and are possibly then hand-picked to disallow removing semantically relevant frequent words [MRS08]. Another approach is to use a predetermined set of stopwords known to carry little meaning within a language. Stopword removal, however, carries its risks. There can be important meaning conveyed in stopwords when more than one word is analysed at once. If an algorithm has can leverage word context, removing stopwords or punctuation may result harmful. Ellen Riloff [Ril95] has shown that including prepositions with significant domain words can be critically important for classifying texts in their respective domain. Also, verb forms and differentiating between the singular and plural forms of nouns had a crucial effect in some cases in Riloff's experiment, which argues against removing certain morphological properties.

According to Manning et al [MRS08], the general trend in IR systems has been to reduce the number of stopwords removed with focus on finding other methods to cope with stop words. Nevertheless, stopword removal remains useful in improving vector space model effectiveness [ILS16, SAY+16] and is present in modern information retrieval [BL17, GTS+16].

2.3 Machine Learning

As NLP, machine learning (ML) also is a subfield of computer science and AI. What separates ML from other forms of AI is that it studies automated "learning" to perform well, where "learning" implies self-improvement upon making observations from data. This is opposed to e.g. AI models whose intelligent behaviour is hand-programmed or achieved by direct computation or manipulation of data.

Similarly to NLP providing tools for IR, one can say that ML provides tools for NLP. For instance, ML has been used to improve multiple NLP tasks, including machine translation [VBB+18] and Part-of-Speech (PoS) tagging [GZH+17], with PoS tagging being an NLP task that considers identifying words as nouns, verbs and other grammatical property categories. Within this work, ML is leveraged to generate optimally useful text embeddings and to learn ranking based on the generated embeddings.

2.3.1 Neural Networks

Neural networks, a subset of machine learning algorithms, has been a hot topic in recent Artificial intelligence (AI) discussion. Machine learning is a field within AI that studies algorithms, which are not explicitly programmed to behave in a certain way, but learn their desired behaviour from data. Neural Networks have gained popularity even in the non-academic world as the first AI models to win against the human current champion in complex games such as the board game Go [Gib17] and card game poker [Klö17]. Neural networks are also famous for being the state-of-the-art models in the field of image recognition [ZLK+18] and achieving human-like performance in multiple image classification tasks. Regarding NLP, a notable example is Google Translate's [WSC+16] highly increased performance after starting to use Neural Machine Translation (NMT) instead of previously used phrase-based machine translation [KOM03].

Importantly to us, the art of generating text embeddings has been affected by neural networks to large extent. Importantly, using neural network generated word and document embeddings has raised attention in the field of information retrieval and the legal setting. For instance, Ash et al [AC18] analyse judges' relations and judicial reasoning by examining spacial relationships between case law embeddings of different judges' verdicts. Moreover, closely related to our topic, Landthaler et al [LWHM16] use word embedding averages to improve a vector space model for retrieving related rights or obligations in EU data protection directives.

Neural networks as machine learning models are officially referred to as artificial neural networks (ANN). However, the "artificial" in "artificial neural network" is often dropped as artificiality is obvious from context, Thus, ANNs are mostly referred to as neural networks (NN). ANNs are algorithms whose study is motivated by the natural way of processing information in the organic brain [Smi97]. Brain, the organ that enables people and animals to control their body, think, reason and understand the world, is made of nerve cells, typically billions of them. These cells enable thought processes and are called neurons. Neurons communicate with each other by sending signals that, if powerful enough, activate recipient neurons, which in turn send signals to other neurons.

While an organic neural network is formed by physical connections between physical neurons, an ANN consists of inter-connected activation functions. As can be seen in Figure 2, the biological neuron has dendrites for receiving, a cell body for processing and an axon for sending signals [BH00], whereas an ANN's neuron is purely mathematically defined as having inputs, activation function and output as depicted in Figure 3.

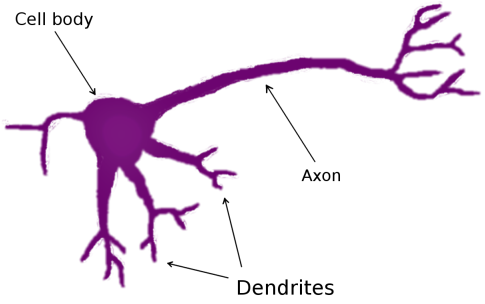

Figure 2: A simplified biological neuron. Signals received through synapses between neurons are picked up by dendrites, processed by cell body and sent to other neurons via the axon.

Each neuron in an ANN produces a real valued activation according to its activation function, which is in contrast to their natural counterparts whose neurons' activations are always binary. Every neuron in has at least one input and a weight for every input of the neuron. The activation, i.e. the output of the neuron, is the weighted sum of the inputs that is passed through the neurons activation function as illustrated in Figure 3.

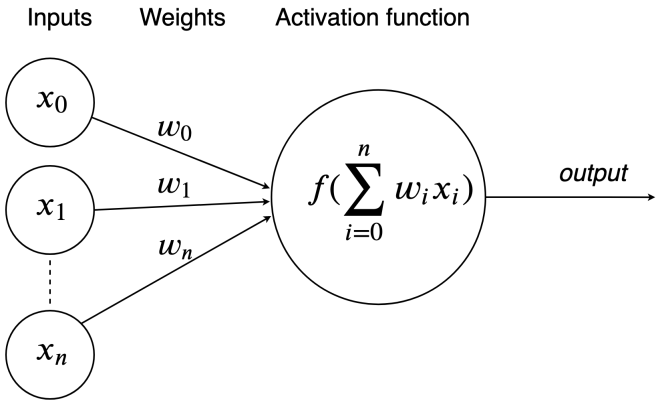

Figure 3: A single neuron, where x_i is an input, w_i its weight and f is the activation function of the neuron.

Depending on the problem at hand, a machine learning model should have complexity that matches the problem to perform optimally. Optimising complexity in machine learning is crucial, since an overly complex model is likely to learn wrong things during training and not generalise well when applied to new data, whereas a simple model might not be able to separate even its training data properly [GBC16]. In NNs, neurons are arranged in layers and the complexity of a neural net is managed by altering the number of layers, the number of neurons within layers and how the neurons in the layers are connected to each other.

At a minimum, a NN contains two layers to allow the net to interact with data from outside the network, one layer for input data and another for output data. There can, however, be arbitrarily many if needed. The first layer of the network is called the input layer since it depicts the network's input data. The final layer is referred to as the output layer for a similarly obvious reason, its outputs serve as output for the whole network. Layers in between input and output layers are referred to as hidden layers due to them being in contact with nothing outside the network.

A common neural network model is a dense feed-forward network, whose structure is visualised in Figure 4. Being dense, within the context of neural networks, indicates

that every input node of a layer is connected to every neuron of the layer. For a dense network, this applies for all layers. Feed-forwardness indicates that connections between the neurons do not form loops, but the arithmetic operations within the network flow directly from one end to the other. In other words, for a one forward or backward pass through an ANN, a neuron is visited at most once. Descriptions for other forms of neural networks, such as recurrent neural network, which includes loops, are omitted, since only dense feed-forward networks are used in this work.

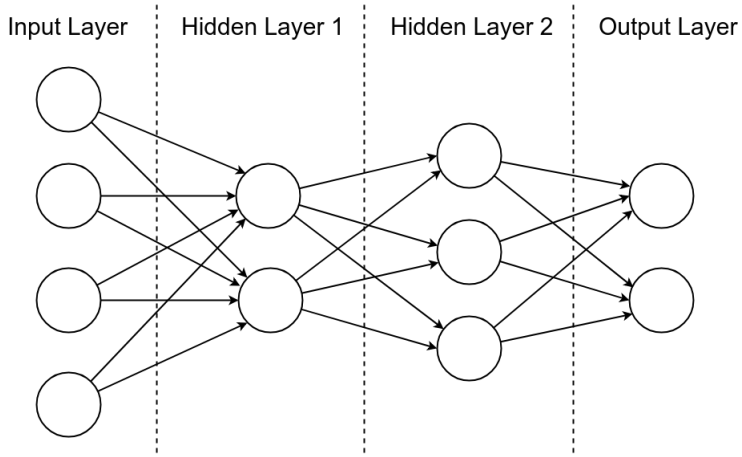

Figure 4: An illustration of a simple feed-forward four layer neural network, where each circle represents a single neuron. In terms of biological neural networks, the arrows represent synapses, which connect the neurons.

As computation for a single neuron's output is the sum of its weights,

$$\hat{y} = f(\sum_i w_i x_i)), \tag{2}$$

passed through an activation function, a dense layer's computation can be conveniently described as a matrix product between the matrix of layer weights and layer inputs. For instance, with a layer of 5 neurons, and 3 inputs, the layer's outputs $\hat{\boldsymbol{y}} \in \mathbb{R}^5$ can be expressed using weight matrix $\boldsymbol{A} \in \mathbb{R}^{3\times5}$ and layer's inputs $\boldsymbol{x} \in \mathbb{R}^3$

$$\hat{\boldsymbol{y}} = f(\boldsymbol{A}^T \boldsymbol{x}). \tag{3}$$

In order to train an ANN, the model needs ground truth targets with which to compare its outputs. Another requirement is that of a loss function to assign error values for its outputs, e.g. squared error

$$\sum_i (y_i - \hat{y}_i)^2, \tag{4}$$

where y is a target value and $\hat{y}$ is a machine learning model's predicted approximation of y.

When the error function is differentiable, which squared error is, gradient descent can be used to minimise the error. Gradient descent is an iterative method that updates model parameters based on the model's output errors' gradients gradually decreasing the errors towards the error function's minimum, thus, enabling a machine learning algorithm to "learn". Essentially, training by gradient descent is a reversed hill-climbing problem, as the error function, when flipped around, can be visualised as a mound on top of which the model attempts to climb step by step.

3 Text embedding models

Document embedding models, i.e. computation procedures that convert text documents into vectors, hold the greatest responsibility regarding vector space model effectiveness since ranking in VSMs is based on text embedding models' outputs. The capability of text embedding models in mapping a text documents' information content to vectors is essential in determining how effectively the documents can be ranked for information retrieval.

With *text embedding* we refer to all forms of text vectorisation, including word embedding as well as document embedding. This section introduces text embedding models that are evaluated for Finnish case law retrieval. The models are presented with some mathematical detail in chronological order by the time of their creation. Besides the individual models, two ensembles of the embedding models are presented.

The embedding algorithms used in this work are the statistical model TF-IDF, probabilistic LDA, neural network based word embedding model Word2Vec and Word2Vec's two variants designed for document embedding, Doc2Vec and Doc2VecC. We use readily available implementations of the embedding models with Doc2VecC as exception. For Doc2VecC, there was available only an inconvenient implementation in C that lacked the ability to transform documents into embeddings after saving a trained model. Thus, we created a python module[1] for Doc2VecC from its original C-implementation to simplify its use for information retrieval and performance evaluation.

3.1 Bag-of-Words Models

3.1.1 Term Frequency - Inverse Document Frequency

Building on the idea of word frequency embeddings, i.e. BoWs consisting of word counts, a widely used [BGLB16] document embedding model in IR, Term frequency - Inverse Document Frequency (TF-IDF), adds regularisation for term, i.e. word, frequencies to give more weight to uncommon words among the whole dataset. In short, TF-IDF consists of BoWs, whose elements are the product of two numeric statistics, term frequency and inverse document frequency.

[1] https://github.com/taikamurmeli/Doc2VecC_python

Term Frequency considers a single document and accounts for the proportional number of occurrences for a term in that document. It can be expressed as a function $tf(t, d)$ of a term t and a document d, which gives the occurrence frequency of t in d normalised by the document's length, i.e.

$$tf(t, d) = \frac{\text{word count for word } t \text{ in document } d}{\text{total word count in document } d}. \tag{5}$$

Inverse Document Frequency [SJ72] (IDF) was introduced by Karen Spärck Jones in 1972 to improve on Term Frequency, as she found out that matches on less frequent terms should carry more value than commonly occurring terms.

As an example to demonstrate Spärck's observation, let us have a corpus which has the common English word *and* appearing multiple times in every document. Querying the corpus by the phrase *coffee and sugar* with mere term frequency gives seemingly random results if the word *and* is not frequent in documents containing the words *coffee* or *sugar* but appeared more often in other documents.

Thus, to weigh a document's term counts appropriately, Spärck presented the inverse document frequency function (IDF)

$$f(N) - f(n) + 1 : \quad log_2(x) - 1 < f(x) \leq log_2(x) \tag{6}$$

where N is the number of documents in a collection of texts D and n is the number of documents containing t, i.e. $n = count(\{d \in D | t \in d\})$.

Nowadays, however, the most cited version of the term weighting function is

$$idf(t, D) = log\frac{N}{n} \tag{7}$$

[Rob04], which is more precise than the original IDF-function eq. (6). In this thesis, we use the common versions of TF in eq. (5) and IDF in eq. (7) for the TF-IDF algorithm

$$tfidf(t, d, D) = tf(t, d) * idf(t, D). \tag{8}$$

While TF-IDF is a powerful algorithm, it has no built-in way to account for word semantics. For TF-IDF, terms like "milk", "coffee" and "alien" are all equally distant from each other. Consequently, TF-IDF gives a similarity score of 0 for sentences "Dogs consume meat" and "Wolves eat other animals" as they have no words in common. Furthermore, being based on BoW, it is oblivious to word orders, causing e.g. "river bank" and "bank river" to be equal. Although there exist pre-processing solutions to add contextual information to TF-IDF, its shortcomings have led to more complex text embedding algorithms being developed for describing documents and the relationships between words.

3.1.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [BNJ02, BNJ03], introduced by Blei et al in 2002, is another relatively old model used in this work. Similarly to TF-IDF, LDA is oblivious to word ordering within documents and treats them as BoWs. However, as opposed to TF-IDF, LDA is not a simple statistical model but rather a machine learning model. Whereas TF-IDF generates embeddings by directly computing word counts in a corpus, LDA has parameters that are trained, i.e. optimised, using the corpus's words as model inputs.

LDA is a generative probabilistic model that assumes a document is created by first sampling topic distributions and then the document's words from selected topic distributions. Using this assumption, LDA reverse engineers topic probabilities for each document in a corpus. The reverse engineering is done by optimising parameters for the topic and word probability distributions to the extent that picked words for a document correspond to the original words in the document.

In essence, for texts, LDA produces a vector embedding for a document, where each dimension in the vector corresponds to some topic determined by the LDA model. Thus, likewise TF-IDF, LDA can be used to embed texts for vector space model. The *latent* part of LDA refers to the topics being latent, i.e. hidden variables, variables that are not directly observed but inferred from observed variables. LDA's latent variables are called *topics* merely to provide an intuitive meaning in the context of natural language processing. The remainder of LDA's name, Dirichlet Allocation refers to the model drawing topic probabilities from a Dirichlet distribution.

In order to depict the model more precisely we use the following variables:

$N \in \mathbb{N}$	corpus document count
$d \in 1..N$	integer identifier of a document
$K \in \mathbb{N}$	number of topics
$k \in 1..K$	integer identifier of a topic
$V \in \mathbb{N}$	number of words in vocabulary
$W_d \in \mathbb{N}$	number of words in document d
$\boldsymbol{w}_d \in 1..V^{W_d}$	vector of word identities for document d
$\hat{\boldsymbol{w}}_d \in 1..V^{W_d}$	vector of word identities inferred by LDA for document d
$\boldsymbol{\alpha} \in \mathbb{R}^K_{\geq 0}, \boldsymbol{\beta} \in \mathbb{R}^V_{\geq 0}$	non-negative vectors, used as distribution prior
$\boldsymbol{\varphi}_k \in [0,1]^V$	probability distribution of words in topic k
$\boldsymbol{\theta}_d \in [0,1]^K$	vector of topic probabilities for document d
$\boldsymbol{z}_d \in \mathbb{N}^K$	vector of topic identities for document d

To infer topic probabilities, first LDA generates the corpus documents as BoWs by the process described in algorithm (1).

foreach topic identity k in *1..K* **do**
  Using a vector $\boldsymbol{\beta}$ as a prior, sample word probabilities per topic k from Dirichlet distribution:
    $\boldsymbol{\varphi}_k \sim \text{Dir}(\boldsymbol{\beta})$
end
foreach document d in *1..N* **do**
  Sample probabilities for topics in document d from Dirichlet distribution:
    $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$
  foreach word index i in $1..W_d$ **do**
    Sample a topic: $z_{d,k} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
    Sample a word $\hat{w}_{d,i}$ based on the sampled topic: $\hat{w}_{d,i} \sim \text{Multinomial}(\boldsymbol{\varphi}_{z_{d,k}})$
  end
end

Algorithm 1: LDA generative process

As any machine learning algorithm, LDA needs to be trained for it to become useful. In the natural language context, it needs training to infer the topic and word probabilities so that it produces BoWs similar to the actual BoWs in a given corpus. The goal is to infer the hidden variables based on the observed ones. Formally, that is to compute the posterior distribution

$$p(\boldsymbol{\theta}_d, \boldsymbol{z}_d | \boldsymbol{w}_d, \boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{9}$$

of the latent variables $\boldsymbol{\theta}_d$ and $\boldsymbol{z}_d$ given priors $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and the document's words $\boldsymbol{w}_d$ for each document d.

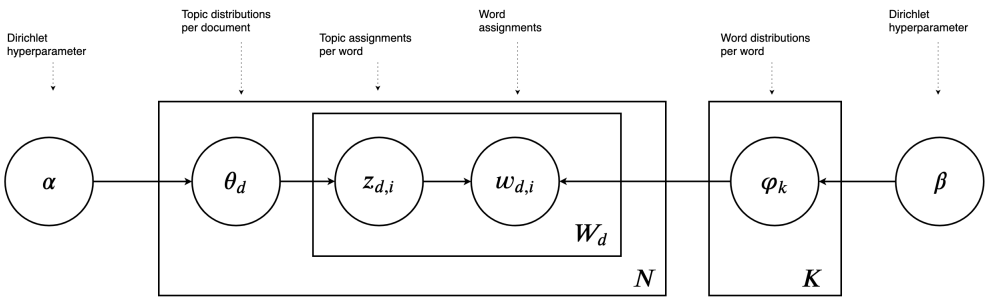

Figure 5: LDA in plate notation. The circles refer to LDA's probability variables and arrows to dependencies. The squares indicate repeated entities, with the exterior corresponding to documents and the interior to words within documents.

However, as Blei et al show in their original paper for LDA, solving the problem directly is computationally intractable due to dependency connections between $\boldsymbol{\theta}_d$ and $\boldsymbol{\beta}$, wherefore they propose using a variational Bayes approximation to compute the topic probabilities.

The variational inference method for LDA introduces two new variables, a dirichlet parameter $\boldsymbol{\gamma}$ and a Multinomial parameter $\boldsymbol{\phi}$. This allows replacing the computationally problematic dependencies in LDA with simpler ones using the variational distribution

$$q(\boldsymbol{\theta}_d, \boldsymbol{z}_d|\boldsymbol{\gamma}_d, \boldsymbol{\phi}_d) = q(\boldsymbol{\theta}_d|\boldsymbol{\gamma}_d) \prod_{i=1}^{W_d} q(\boldsymbol{z}_{d,i}|\boldsymbol{\phi}_{d,i}). \tag{10}$$

The variational distribution is optimised to resemble the original posterior distribution in eq. (9) by minimising the Kullback-Leibler divergence, a measure of probability distributions' differences, between the two distributions. As a reference to training a neural network, the Kullback-Leibler divergence between the two distributions serves a similar purpose as the loss function in neural networks. While neural nets loss is optimised via gradient descent, LDA's Kullback-Leibler divergence is minimised by variation expectation maximisation, as explained by Blei et al, or alternatively, via Gibb's sampling, which was proposed later by Porteous et al [PNI+08] to speed up LDA training time.

Contrary to TF-IDF, for training the model, LDA has manually selectable parameters, which are referred to as hyperparameters. Model parameters, on the other hand, are parameters that are learned by a machine learning model. The most notable hyperparameter in LDA is the number of topics for the documents, which we denote by K. Since K cannot be known beforehand, different values of K are tested to see which provide optimal results. Besides affecting model performance, the value K also determines the dimensionality of LDA embeddings, making small values of K preferable in terms of space and time complexity of the model.

3.2 Neural Network Models

3.2.1 Word2Vec

Word2Vec [MCCD13, MSC+13] is a neural network model introduced by Mikolov et al's research team in 2013, with the chief purpose of creating dense vector representations for words. Similarly to LDA, Word2Vec is an unsupervised learning method, as it does not require labelled text for its training. It has become a popular word

embedding method, as it is effective in mapping words' semantic meanings close to each other using mere text data as its input. Word2Vec's competence is shown in the original paper where Mikolov et al show the generated vectors' performance in phrase analogy task. In the task, word relationships of form "What is the word a that is similar to b in the same sense as c is similar to d?" are tested with simple algebraic vector operations. Famous examples of Word2Vec's capability in this include computing the vector of word *queen* from *king, man* and *woman* vectors,

$$v(queen) \approx v(king) - v(man) + v(woman)$$

and deriving capital vectors using countries e.g.

$$v(Rome) \approx v(Paris) - v(France) + (Italy),$$

which tell that "queen" is similar to "woman" as "king" is to "man", and that "Rome" is to "Italy" as "Paris" is to "France".

As a consequence of Word2Vec's success, it has received multiple competitors including GloVe [PSM14], SVD [LGD15] among others. In Word2Vec's defence Vylomova et al [VRCB15] show that Word2Vec is superior to other later methods in clustering vectors to lexical relations. However, pre-trained Glove vectors have been slightly more effective than Google's Word2Vec embeddings[2] trained on ca.100 billion words, e.g. when used with Named Entity Recognition [CN15]. Thus, as the differences in word embedding model performances are small and inconsistent, we consider only the original model in this work.

Word2vec is a simple shallow neural network that consists of input layer, one hidden layer and an output layer. Although Word2Vec is considered an unsupervised learner, it is trained in a supervised fashion by predicting documents' textual content from a sliding window of text within the documents. There are two versions of the model, Continuous Bag of Words (C-BoW) and Skip-gram, which differ by their inputs and outputs. The differences between the versions are illustrated in Figure 6.

[2]https://code.google.com/archive/p/word2vec/

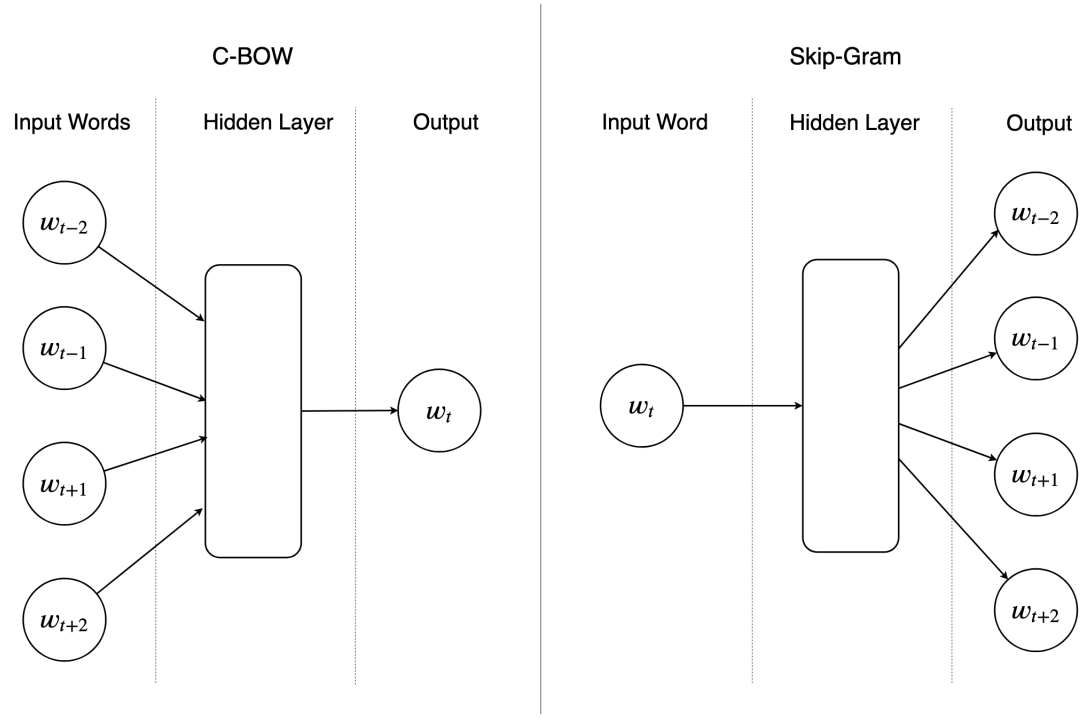

Figure 6: Visualisation of Word2Vec C-BoW and Skip-gram model differences with window of size four, where $w(t \pm n)$ indicates a context word's position in the window.

Rong [Ron14] explains the mathematical details of the Word2Vec model and how it is optimised for fast training with large data. In this work, the two variations of the model are depicted more briefly. For mathematical description of the model, the network's components are denoted as:

w_t	t:th word in the vocabulary
V	size of the vocabulary
H	size of the hidden layer and thus, also size of resulting vector embeddings
C	maximum distance between context word and predicted word
$\boldsymbol{c}_t \in \mathbb{R}^{1 \times V}$	BoW of the training window, i.e. the context words $w_{t-C}, ..., w_{t-1}, w_{t+1}, ..., w_{t+C}$
$\boldsymbol{A} \in \mathbb{R}^{V \times H}$	matrix representation of weights between the input and hidden layer of the model
$\boldsymbol{a}_t$,$\boldsymbol{b}_t \in \mathbb{R}^{H \times 1}$	respective t:th column vectors of weight matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

The C-BoW version of Word2Vec is trained via gradient descent to maximise the probability of predicting a word that is taken out of its context, i.e. the surrounding words of the predicted term. In C-BoW, the hidden layer's output is the averaged context words' weights $\boldsymbol{A}^T \frac{1}{C} \boldsymbol{c}_t$ and softmax function [Bis06], i.e. normalised exponential function

$$x_j = \frac{\exp(\boldsymbol{y}_j)}{\sum_{k=1}^{K} \exp(\boldsymbol{y}_k)}, \tag{11}$$

is used on the final layer to force the model's outputs into probabilities. This means that, given a context window $\boldsymbol{c}_t$ of a document d, the probability of observing a word w_t according to the model is

$$p(w_t|\boldsymbol{c}_t) = \frac{\exp(\overbrace{\boldsymbol{b}_t^T}^{\text{output layer weights}} \overbrace{\boldsymbol{A}^T \frac{1}{C} \boldsymbol{c}_t}^{\text{averaged hidden layer outputs for context words}})}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T \boldsymbol{A}^T \frac{1}{C} \boldsymbol{c}_t)}. \tag{12}$$

The network's loss is defined as the logarithmic loss for the above probability

$$L = -log(p(w_t|\boldsymbol{c}_t)), \tag{13}$$

thus C-BoW maximises the probability of predicting a word missing from its context.

C-BoW's, alternative, Skip-Gram, can be considered a mirrored version of C-BoW, as it attempts to maximise the probabilities of context-words for a given term.

The probability for a context word w_{t+i} in Skip-Gram is defined as

$$p(w_{t+i}|w_t) = \frac{\exp(\boldsymbol{b}_{t+i}^T \boldsymbol{a}_t)}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T \boldsymbol{a}_t)}, \tag{14}$$

and its loss

$$L = \sum_{-C \leq i \leq C, i \neq 0} - \log p(w_{t+i}|w_t) \tag{15}$$

is the sum of context word probabilities' logarithmic losses.

The word vectors for each word w_t are obtained from a trained model for both C-BoW and Skip-Gram by using the neural net weights corresponding to word w_t as the vector dimensions. Similarly to LDA, Word2Vec has its vector size provided as a hyperparameter. In Word2Vec and its dervivatives, the resulting vector size

is the same as the number of neurons in the neural net's hidden layer. Word2Vec incorporates additional hyperparameters that affect model performance, which include context word window size C and the number of times the model goes through its training data to learn weights for word representation. We refer to these data run-throughs as *epochs*.

Goldberg and Levy [GL14] analyse the model details further and discuss the reasons behind Word2Vec being such a powerful model. Interestingly, Goldberg and Levy rest their case at not knowing why the model works as well as it does. They argue that, although Word2Vec's performance is related to the well studied distributional hypothesis [RG65, Sah08], which states that words often appearing in the same context have similar meanings, the fact that Word2Vec's goal is to distinguish between good and bad context-pairs does not count as a formal explanation for the model's effectiveness, but is rather "hand-wavy". Regardless of the lack in understanding the linguistic reason for word embeddings' excellence, using them where possible often provides state-of-the-art results. Thus the models' vectors have been used for a wide variety of tasks such as query expansion [KSK16], text classification [ZSLL15], sentiment analysis, question classification and answering [MBXS17].

We, however, are interested mostly in document representation. For this, we can leverage word embeddings to construct embeddings for whole documents by simple mathematical operations. The sum or average of a document's word embedding vectors of can be thought of as a vector representing the document. However, this method is likely to no longer provide state-of-the-art results even for short texts [BS16]. The word embedding averages should rather be used as baseline for more intelligent document embedding techniques alongside the widely popular TF-IDF. To support this, in a recent article [SWW$^+$18], Wang et al. analyse averaged word embeddings that they call simple word-embedding-based models (SWEM). They show that using SWEMs should still be considered as a noteworthy baseline, since in some tasks SWEMs produce better results than more complex models.

Another approach besides averaging, the concatenation of word vectors, may seem intuitive, but the approach holds a lot of problems. Similarity metrics that require same dimensionality for different sized texts, such as cosine similarity in eq. (1), are bound to fail. To overcome this, the long documents could be cut or the short ones padded with zero-dimensions, but this would make the metric sensitive to document length. Discriminating documents by their word length is something that should be avoided, since it does not account for semantic similarity; a brief overview of a story

should be considered semantically similar to a long detailed version of the same story. Additionally for long texts, computing similarity for vectors with extreme common non-zero element counts can result in unacceptable computation times and space requirement for information retrieval.

3.2.2 Doc2Vec

In 2014 the creators of Word2Vec, Le and Mikolov, presented a modified version of their Word2Vec model, *Paragraph vector* [LM14], to make the word embedding model embed whole texts rather than words. In the original paper, Mikolov et al show that paragraph vector is better suited for embedding documents to fixed-length vectors than averaging word vectors or LDA when the embeddings are used for measuring similarity of Wikipedia articles. The paragraph vector model has since then received the more popular name Doc2Vec, which is how we will refer to the model.

Like Word2Vec, Doc2Vec has also received enough attraction to have papers dedicated to thorough analysis of the model. Lau et al provide [LB16] recommended hyperparameters as well as comparison to state-of-the-art document embedding methods and Lai et al show [DOL15] that Doc2Vec soundly beats LDA and averaged word vectors in classifying Wikipedia articles to their respective categories.

Doc2Vec's increased performance compared to Word2Vec is obtained by using a document identifier in each context window when training the model. In the model's perspective, the document identifier acts as memory of the document's topics during training and is able represent missing context information the window words are lacking.

The document identifier is applied to the model as one-hot encoded input in the same manner as context words in a training window. Once Doc2Vec is trained, the weights of its document ids are taken from the model and used to represent the documents as their vector representations. Supposedly the weights capture the semantic representation of a document the same way individual words' weights represent the meanings of words.

Like Word2Vec, Doc2Vec is a shallow neural network that includes two versions of the algorithm. "Distributed memory" (DM) model resembles the C-BoW as it outputs a single word from context, whereas the other model, "Distributed Bag of Words" (D-BoW), outputs words using the document's id as its only input and is

more similar to Skip-Gram with its single input and multiple outputs. DM and D-BoW differences are visualised in Figure 7.

Denoting the vector of model's weights for a document d as $\boldsymbol{d} = \boldsymbol{a}_d^T$, the Doc2Vec DM model is described mathematically as

$$p(w_t|\boldsymbol{c}_t, \boldsymbol{d}) = \frac{\exp(\boldsymbol{b}_t^T(\boldsymbol{A}^T \frac{1}{C}\boldsymbol{c}_t + \boldsymbol{d}))}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T(\boldsymbol{A}^T \frac{1}{C}\boldsymbol{c}_t + \boldsymbol{d}))}, \tag{16}$$

which is identical to Word2Vec's C-BoW except for the added document weights and the weight matrix dimension $\boldsymbol{A} \in \mathbb{R}^{(V+N)\times H}$.

Accordingly, the probability for a context word w_{t+i} in D-BoW is defined as

$$p(w_{t+i}|w_t, \boldsymbol{d}) = \frac{\exp(\boldsymbol{b}_{t+i}^T(\boldsymbol{a}_t + \boldsymbol{d}))}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T(\boldsymbol{a}_t + \boldsymbol{d}))}. \tag{17}$$

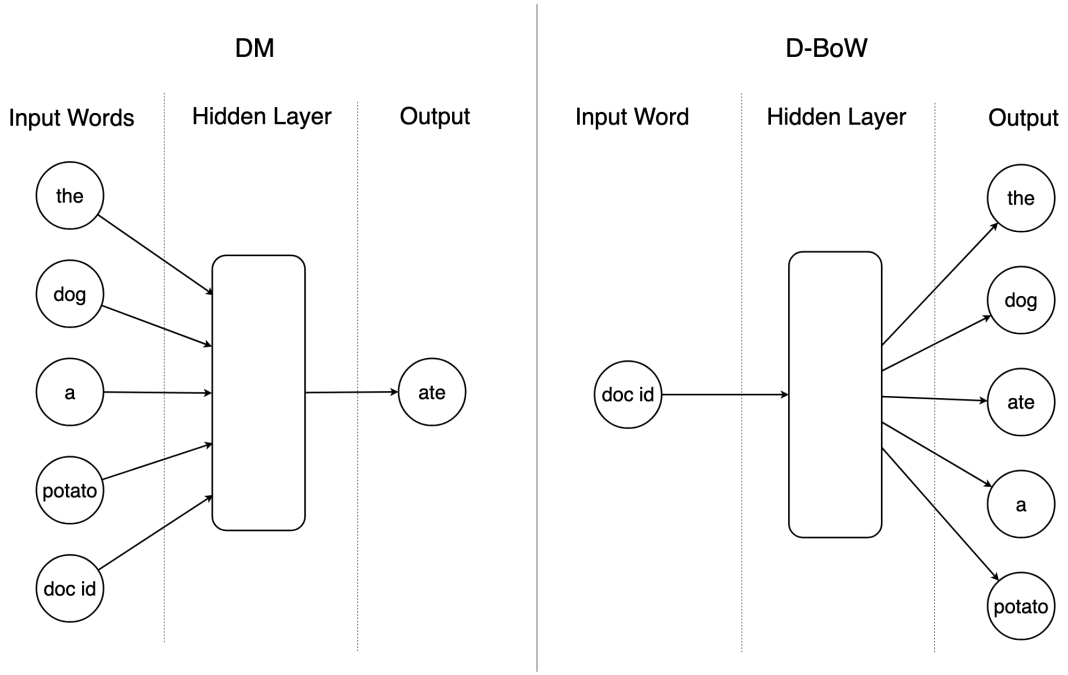

Figure 7: Doc2Vec DM and D-BoW models illustrated with an example window.

3.2.3 Doc2VecC

Doc2VecC [Che17] by Minmin Chen is another extension of Mikolov et al's Word2Vec algorithm that provides effective document representations for semantic relatedness and document classification tasks. Interestingly, Doc2VecC is not an algorithm that directly creates document embeddings, it rather attempts to modify word vectors

in a way that the average of a document's word vectors meaningfully represent the semantics of said document.

Doc2VecC is motivated by Mikolov et al's observation that averages or sums of Word2Vec embeddings can contain a lot of semantic meaning from the whole set of words. On grounds of this observation, Chen proposes that meaningful document representations can be learnt by averaging word vectors during training. As a comparison to Doc2Vec, which has a separate vector for representing a document embedding during training, Doc2VecC uses the average of randomly sampled words from the document as document's context. This document context is then added to the local context, i.e. context given by words in the training window, to aid in inferring word probabilities during model training.

Chen refers to the random sampling of document context words as *corruption* and states it is critical to both training speed and model performance. The corruption is applied to document context, denoted as $\tilde{\boldsymbol{x}}$, by setting the context dimensions $\tilde{x}_t$ to 0 randomly with a probability q, given as a hyperparameter for the model. The remaining dimensions are normalised to $1/W_d(1-q)$ times the dimension's original value $\tilde{x}_t$, where W_d is the number of words in a document d. This gives a formal definition for the corruption:

$$\tilde{x}_t = \begin{cases} 0, \text{with probability } q \text{ or word } w_t \text{ not in document } d \\ \frac{x_t}{W_d(1-q)}, \text{otherwise} \end{cases} . \tag{18}$$

The probability for word w_t in Doc2VecC is the same as in Word2Vec except for added document context. The context enhanced word probabilities are

$$p(w_t|\boldsymbol{c}_t, \tilde{\boldsymbol{x}}) = \frac{\exp(\boldsymbol{b}_t^T(\overbrace{\boldsymbol{A}^T \frac{1}{C}\boldsymbol{c}_t}^{\text{window context}} + \overbrace{\frac{1}{D}\boldsymbol{A}^T\tilde{\boldsymbol{x}}}^{\text{document context}}))}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T(\boldsymbol{A}^T \frac{1}{C}\boldsymbol{c}_t + \frac{1}{D}\boldsymbol{A}^T\tilde{\boldsymbol{x}}))} \tag{19}$$

for C-BoW and

$$p(w_{t+i}|w_t, \tilde{\boldsymbol{x}}) = \frac{\exp(\boldsymbol{b}_{t+i}^T(\boldsymbol{a}_t + \frac{1}{D}\boldsymbol{A}^T\tilde{\boldsymbol{x}}))}{\sum_{t'=1}^{V} \exp(\boldsymbol{b}_{t'}^T(\boldsymbol{a}_t + \frac{1}{D}\boldsymbol{A}^T\tilde{\boldsymbol{x}}))} \tag{20}$$

for Skip-Gram.

Although the modification to Word2Vec is rather minuscule, Chen shows that Doc2VecC is able to surpass Doc2Vec's performance in a sentiment analysis classification task.

Additionally, the word vectors generated by Doc2VecC outperform Word2Vec in the phrase analogy task devised by Mikolov et al in syntactic questions and most semantic questions.

3.3 Ensembles

In addition to studying the presented models, ensembles of the models are evaluated to investigate whether a combination of the algorithms outperforms any single model. Kim et al [KSCK19] already showed recently that multi-co-training the models TF-IDF, LDA and Doc2Vec for topic classification yielded promising results.

Compared to Kim et al, we use a simple method to create ensembles of the models. Our goal is to infer similarity values corresponding to human evaluation, and those are computed from the embeddings. Thus, we can take averages over the models' real valued predictions to create an ensemble model.

We test two ensembles, a Mean Ensemble, which weighs all the model predictions equal, and a Linear Regression [SL12] ensemble, a model with optimised weights based on ground truth similarity values.

Regression is a task for computer program to predict a numerical value based on some input [GBC16]. Linear regression is regression model that assumes its inputs $\boldsymbol{x} \in \mathbb{R}^n$ are linearly related to an observed variable $y \in \mathbb{R}$. In our case, $\boldsymbol{x}$ contains similarity values for a document pair computed from embeddings given by the individual models and y is a ground truth human assigned similarity value for the pair of case law documents. As the relation between predicted similarities and human similarities is assumed linear by the regression model, for each target y, it optimises weights $\boldsymbol{\beta} \in \mathbb{R}^n$ in the equation

$$y = x_1\beta_1 + .. + x_n\beta_n + \epsilon = \boldsymbol{x}^T\boldsymbol{\beta} + \epsilon, \tag{21}$$

where $\epsilon \in \mathbb{R}$ is an error term denoting disturbance in the linear relation.

3.4 Text Embedding Similarity

As stated in the background section, ranking for information retrieval requires ordered values for document pairs. We use embedding models to represent case law documents in vector space to enable ranking based on vector correlation. We use two methods to extract a numerical similarity value from a pair of embeddings: direct computation using cosine similarity in eq. (1) and a machine learning method by Kiros et al [KZS+15] to learn a ranking from document embeddings and their respective gold standard similarities.

In the learning to rank method, similarity values for pairs of document vectors are learned by logistic regression using gold standard values as learning targets. Logistic regression, unlike its name suggests, is a classification model and not a regression model. The model predicts a probability $Pr(Y = 1)$ based on a linear combination of the logistic regression's vector of inputs[3] $\boldsymbol{x}$, weight vector $\boldsymbol{\beta}$ and logarithmic base b:

$$Pr(Y = 1) = \frac{1}{1 + b^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. + \beta_n x_n)}} = \frac{1}{1 + b^{-(\boldsymbol{\beta}^T \boldsymbol{x})}} = \frac{b^{\boldsymbol{\beta}^T \boldsymbol{x}}}{1 + b^{\boldsymbol{\beta}^T \boldsymbol{x}}}, \tag{22}$$

thus resembling linear regression in eq. (21). With more than two classes to predict and using the differentiationally convenient natural base e, logistic regression assigns a probability $Pr(Y = c)$ according to eq. (23) where c denotes the class in question, and in our case, $c \in \{0, 1, .., 5\}$.

$$Pr(Y = c) = \frac{e^{\boldsymbol{\beta}_c^T \boldsymbol{x}}}{\sum_i e^{\boldsymbol{\beta}_i^T \boldsymbol{x}}} \tag{23}$$

Logistic regression inputs are represented as a single vector. Thus, to map a document embedding pair of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ into a single vector, a concatenation of two features of the vectors are selected to represent the pair, namely dot product $\boldsymbol{u} \cdot \boldsymbol{v}$ and absolute difference $|\boldsymbol{u} - \boldsymbol{v}|$.

Given a gold standard similarity set that consists of similarity values $y \in [0, 5]$, in order to be effective, the learning to rank classifier should predict similarity scores similar to gold standard values. However, the logistic regression classifier predicts probabilities for classes instead of numerical similarity values. Thus, similarly to Kiros et al, we use a mapping described in eq. (24) from target similarities y to logistic regression's target vectors $\boldsymbol{p} \in [0, 1]^6$ using a vector of the possible integer similarity ratings $\boldsymbol{s} = [0, .., 5]$ that satisfies $y = \boldsymbol{s}^T \boldsymbol{p}$. The mapping was introduced by Tai et al [TSM15] and is equivalent to setting the $\lfloor y \rfloor$th value of $\boldsymbol{p}$ as one and the rest of the vector values as zero.

$$p_i = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

Since $y = \boldsymbol{s}^T \boldsymbol{p}$, predicted similarity values $\hat{y}$ are obtained by multiplying the logistic regression's outputs $\hat{\boldsymbol{p}}$ with the vector of distinct similarities $\boldsymbol{s}$, i.e. $\hat{y} = \boldsymbol{s}^T \hat{\boldsymbol{p}}$.

[3] A fixed variable $x_0 = 1$ is prepended to inputs $\boldsymbol{x}$ in order to use the more concise matrix representation in eqs. (22) and (23).

4 Data

Algorithms and models can be considered tools, and there hardly exists a tool that works well for every task. It is crucial to understand the data one is working with, as well as the models that process the data.

This section describes the data used in this work and how the data is preprocessed for the embedding models.

4.1 Finnish Case Law

The data used in this work is a corpus, i.e a collection of structured texts, of Finnish case law documents. At the time of writing this section and model evaluation, the Finnish case law corpus consists of 13053 judgements from 1980 to 2019. The texts are in XML-format and are provided by the Finnish Ministry of Justice. The language of the texts is Finnish, which is an agglutinative language. This provides challenge for text embedding models, since words often appear in multiple forms in agglutinative languages.

The effects of the Finnish language having a high quantity of inflected words, compared to e.g. English, can be seen from Table 1. The count of distinct terms in the corpus totals up to more than 5% of the total word count and the distinct term count with inflections as separate terms is over double the term count when inflections are disregarded. As a reference, the King James Version of the Bible[4], including the book titles, consists of 821 130 terms with only 12 798 of them distinct, or 17 330 when verse identifiers such as “22:20” are included.

Number of documents	13 053
Total term count of corpus	17 154 739
Distinct term count of corpus	899 488 (342 668)
Average term count in a document	1314
Median term count in a document	659
Maximum term count in a document	84224
Minimum term count in a document	25

Table 1: Finnish case law corpus statistics. The distinct count in parenthesis indicates the count when words have been reduced to their dictionary form i.e. lemmatised.

[4]http://www.gutenberg.org/cache/epub/10/pg10.txt

Even with the different passages considered as distinct terms, King James Bible's distinct term count amounts only to 2.1% of its total term count. The term counts for FinLex data and the reference text are computed from tokenised texts in lower case with punctuation removed. All numerical data, such as Bible passages or ECLI-identifiers are counted as ordinary terms.

A redeeming quality within our case law data, in terms of expected model performance for similarity computation, is that the case law contains several aspects alleviating the models' task. Foremostly, each text contains the laws that are applied in giving a legal decision. Judgements having one or more laws in common inherently convey that they are in a certain way similar to each other, and these laws are depicted in only two possible forms, either abbreviated or in full form. Another aspect is that the case law terminology is quite narrow, as is it written by juridical personnel, and chiefly for juridical personnel. This makes the case law corpus somewhat simpler dataset compared to e.g. news or scientific articles, where there exist far more varying terminologies spanning a vast amount of different topics and writers with distinct styles of writing.

4.2 Preprocessing

As stated in the introduction, texts need to be normalised to some standard form before applying algorithms to process them. Our models use vectors of words as inputs, wherefore we tokenise the case law documents into word tokens by splitting the text by whitespace characters. Regarding model input, all our text embedding models consider distinct character combinations in a token completely different from each other. Thus, inflected word forms are likely to provide challenges for the models. In addition, natural language texts contain punctuation and stopwords which carry little or no semantic meaning by themselves. Thus, they may be nothing but a hindrance for algorithms oblivious to word ordering.

In order to diminish word inflections' and hindering terms' effects on embedding models, we normalise words by stripping punctuation, removing stopwords, lemmatising words and expanding abbreviations. Additionally, automatic query expansion is performed on BoW models to reduce problems caused by synonymous words.

4.2.1 Punctuation and stopwords

Punctuation is removed from the tokenised words by stripping punctuation marks from the beginning and the end of a word. Therefore, "cat." would become "cat", "'knowledge'" becomes "knowledge" and other tokens with punctuation in the middle, such "0.123" or "linja-auto" (a bus, as in public transport vehicle) would be left as is.

For stopword removal, we filter out stopwords using a list of predefined stopwords of the Finnish language. We use the Natural Language Toolkit's [LB02] Finnish stopword set as our predefined stopwords for preprocessing the Finnish case law data.

4.2.2 Abbreviations

The law texts contain abbreviations at some times for certain key terms, while not always. In most cases, the key terms are abbreviated rather simply, e.g. "momentti" (section) has a shorthand "mom", and "laki" (law), is reduced to its initial "l". In order to make our embedding models consider full versions of the law terms the same as abbreviated, the abbreviations are expanded by simple regular expressions targeting the word "mom" and anything ending in "l".

Word "mom" does not exist in Finnish, and words ending in "l" are rare, with expanded versions of them not confusable with laws. Thus, the simple law term disabbreviation can be deemed a safe operation. Other abbreviation expansions are not considered due to them being rather rigorous to implement without possibly doing more harm than good. As an example "Asetus" (*decree*) is abbreviated as "A" and "A" is also used for anonymised names, which implies that information about the word context is required to assign the expansion to correct terms.

In addition to expanding abbreviations, the laws and their subsections are combined to enable easy differentiation between law sections and subsections for the models. For instance, "law x section y subsection" is converted to "law-x law-section-x-subsection-y

4.2.3 Lemmatisation

Two applications that support Finnish have been available online since year 2016: LAS [Mäk16] and FinnPos [SRLK16]. Both of the lemmatisers leverage OmorFi, a freely available open source lexical database for the Finnish language [Pir15]. From the two, LAS was chosen to be used to lemmatise case law for this project.

These lemmatisation applications are evidently not perfect, which is likely to difficulty in automatically distinguish between two identical words in terms of spelling. As an example, "voi" is both an third-person present form of "voida" (Engl. to be able) and also the base-form of the noun "voi" (Engl. butter). When glancing at LAS lemmatiser results, it can be seen that it leaves the word "voi" as is, even when it is used as a verb. Another clearly erroneous example is the word "niistä", an inflection of the pronoun "ne" (Engl. they) being interpreted as the imperative form of "niistää" (Engl. to blow ones nose).

The most obvious problems in the lemmatiser performance seems to be solvable through context based Part of Speech (POS) tagging, which would allow, e.g., verbs to be distinguished from pronouns more precisely. However, this would require an efficient automatic POS-tagger and improving existing lemmatisers is out of scope for this work.

4.2.4 Automatic query expansion

Automatic query expansion (AQE) is a common method used to improve information retrieval by adding related words to queries [Eft96, AD17]. Word frequency based methods on their own would have no way of understanding that two texts are similar in semantic sense if they have no words in common. This is problematic since the texts can nevertheless be semantically nearly identical when containing synonyms. The sentence "A chicken crossed the road" has little difference in semantic meaning with "The hen walked across the street". But regarding term frequency, there is nothing in common apart from a single stopword in the sentences.

While query expansion's potential benefits are rather easy to comprehend as it allows querying by synonyms, query expansion may be useless or even harmful if words are carelessly mapped together. Specifying an optimal level of similarity that accounts for synonymity is a challenge in itself. It is widely agreed among linguists that true synonyms do not exist [Urd93], there can only be near synonyms. Words appear in different contexts with varying frequency and appearing in a certain context alone

can slightly change the meaning of a word in the sense of connotation, familiarity and appropriateness. Moreover, as Carpineto et al. describe in their survey [CR12], automatic query expansion is required foremostly in short queries where there might not be many words to describe the information need. In our case, the queries are long on average and case law has a certain terminology. This minimises what Carpineto et al call "the most critical language issue", i.e. a term mismatch problem, where users and IR systems use different vocabularies.

We test query expansion using three Finnish ontologies containing semantically linked words to add synonyms, hypernyms and hyponyms, i.e. near equivalent, broader and narrower terms, to our case law documents. Query expansion is performed as weighted expansion with weights being one for original words, half for synonyms and one fourth for hypernyms. The ontologies from which term relations were retrieved are OIKO[5], an ontology of Finnish legal terms, and a Finnish ontology collection KOKO[6]. KOKO includes Yleinen Suomalainen Ontologia[7] [HVK+05] (general Finnish ontology), which is an ontology of the Finnish language based on the Finnish thesaurus Yleinen Suomalainen Asiasanasto[8].

Query expansion is applied by manipulating BoW frequencies, which is not plausible for the neural net models. Therefore, query expansion is tested only on the BoW models, TF-IDF and LDA, and not on Word2Vec nor its extensions. Terms in documents are lemmatised for query expansion, since the ontology word labels used for searching related terms are in their base form.

[5]https://finto.fi/oiko/en/
[6]https://finto.fi/koko/en/
[7]https://finto.fi/yso/en/
[8]http://finto.fi/ysa/fi/

5 Evaluation methods

IR system performance evaluation considers two aspects, efficiency and effectiveness [BCC]. Efficiency is measured by query execution time or space requirements, which are both straightforward to assess. Effectiveness, i.e. the quality or relevance of query results, on the other hand, is a far more complex matter to evaluate, as it requires human judgement. Regardless of the difficulty in assessing effectiveness, it is of utmost importance, as it does not make sense to use a model for ranking query results if there are no means to tell whether the model performs well or not.

A rather obvious method for effectiveness evaluation is to take a glance at the outputs of a model and decide how good the results are using one's own intuition. For instance, one might take the top ten similar results to a query and examine the order and relevance to the query document or term. This procedure could then be repeated for all models that require assessment. However, in order to have a numerical score for the models, even with only 5 different queries and examining the top ten results, there would have to be $5*10*modelcount$ comparisons. As similarity assessment by human examination is rather tedious work, especially when dealing with large documents, automation is needed to properly assess similarity-computing algorithms with reasonable resources.

In order to perform accurate automatic evaluation for ranking, a ground truth with which to compare a model's results is required. Models for retrieval system evaluation without a ground truth have been proposed [SNC01, WC03, Spo07], but empirical studies [SL10] show that they are unreliable when evaluating a single IR System.

A ground truth set consisting of human assigned document similarity labels is referred to as a gold standard set. A gold standard similarity value is a ground truth semantic similarity score assigned by humans. With semantic similarity being difficult for humans, and keeping in mind that humans make plenty of mistakes, similarity gold standards are far from perfect. However, as its name "gold standard" suggests, it is the best thing available.

While being optimal ground truth data, gold standard similarity scores are strenuous to obtain as they need to be manually assigned. Shi et al [SLW10] state that the creation of relevance judgements, i.e. similarity scores, consumes the most time and human resources in the creation of an information retrieval system. This section describes our document embedding model evaluation for IR ranking with two meth-

ods, topic classification and correlation with a gold standard label set. In addition, we present our gold standard set and explain how it was acquired.

5.1 Topic classification

Due to the workload required to generate a gold standard set, we investigate an alternative approach which does not require manual labelling of document similarities for our Finnish case law corpus. In order to skip using hand-labelled similarity values for evaluation, we use substitutes for the similarity values as ground truth values. To be of use, the substitute values require at least two properties. Firstly, they need to be something that can be reasonably assumed to have a correlating model performance with the similarities. Secondly, the substitute ground truth should be automatically extractable to get rid of the tedious manual labelling.

We select topics, i.e. semantic spaces, which are included in the case law as keywords as our suitable substitute. Both topic classification and semantic similarity computation require understanding of text semantics. If the case document embeddings contain information that allow them to be classified into topics effectively, they can be assumed to also contain information that enables computing reasonable similarities between documents. On the other hand, if the embeddings are poorly designed for topic classification, the model that produced the embeddings is unlikely to perform well in estimating document similarity.

The number of distinct keywords in the case law data is ca. 16 000. Keeping all of the keywords would make the topic classification label set extremely large considering that a document may have multiple keywords. Additionally, a significant proportion of the keywords appear only in one or a few documents, which makes the rare keywords unreasonably difficult to be learned from the data. Thus, a minimum keyword appearance count of 15 is chosen to reduce the label space to a more convenient size. Consequently, we remove all documents from the evaluation data set which do not contain any accepted keywords. With the minimum keyword count restriction, the label space is reduced to one fiftieth of the original. This speeds up the training, makes it less memory intensive and increases the chances for a classifier to learn all the correct topics for embeddings.

To evaluate that a trained document embedding model's performance is generalisable, we split the remaining documents alongside their labels to training set and test set at ratio 80:20. Our document embedding models, excluding TF-IDF, contain

hyperparameters, i.e. user defined parameters used for learning model parameters, which need to be tuned for optimal performance. Thus, we select a range for each tunable hyperparameter and train the embedding models with different hyperparameter combinations in order to find out the best combinations for topic classification. We obtain a range for a selected hyperparameter by fixing other hyperparameter values and evaluating the model with increasing and or decreasing values of the selected hyperparameter until there is no improvement. Starting hyperparameters for range selection were picked from the readily available implementations of the document embedding models. In addition to hyperparameter tuning, we similarly tune the preprocessing steps stopword removal, lemmatisation and query expansion alongside the hyperparameters by separate training for each preprocessing combination.

The embedding models are trained using the training set and then the test documents are embedded by the trained model. With all documents embedded, the embeddings can be classified into topics. As our goal is to distinguish which document embedding model is the best, it is not critical to optimise the performance of the topic classifier model. Thus, we chose perceptron, a neural network consisting of a single neuron, to classify embedding vectors into topics. Perceptron was chosen based on training time and performance from several models with readily available implementation. The other tested models include support vector machines, logistic regression and gradient boosted decision trees.

Perceptron, having only one output node, is a binary classification model and documents may belong to more than one topic, wherefore a binary classifier cannot be directly applied for a multi-class prediction problem. Therefore, we apply one-vs-rest classification [BWG10], to use perceptron as a multi-label classifier. In one-vs-rest, a single binary classifier is assigned for each target class so that the binary classifier generates a class probability for all the classes separately. We use the same train-test split for the topic classifier as with the document embedding models.

For algorithm performance measures, we opt for relevance measures computed from true positive (TP), false negative (FN) and false positive (FP) prediction counts. We use three common measures that provide information on model accuracy: precision and recall in eq. (25) and their harmonic mean known as F1-score in eq. (26), which have been previously used by e.g. Quercia et al [QAC12] for topic classification evaluation.

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN} \tag{25}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{26}$$

To note a possible problem for our topic classification method, the keyword label set contains some inconsistencies. As an example, cases ECLI:FI:KKO:2012:3[9], ECLI:FI:KKO:1989:15[10] and
ECLI:FI:KKO:2004:120[11] are all three about crimes involving driving a car, yet they have no keywords in common. This can affect negatively the usefulness of topic classification for IR ranking evaluation.

5.2 Gold standard correlation

Topic classification as a substitute for similarity ranking is slightly far-fetched and the two have a risk of not being sufficiently correlated. Additionally, the topic classification with our corpus has its own flaws, wherefore spending time to create a ground truth similarity set with human evaluation may be deemed well spent.

Assuming there is a gold standard set of similarity scores available to use for evaluation, one can use correlation measures to assess automatically derived similarities. As an example, the SemEval International Workshop[12] has used linear correlation measures, namely Spearman rank order correlation and Pearson correlation to evaluate the relatedness between gold standard similarity annotations and algorithmically acquired text similarities.

The Pearson correlation [Sta13], also known as linear correlation and product moment correlation, is a rather straightforward correlation measure. Given two vectors or populations, e.g. gold standard and predicted similarities, it is the quotient of their covariance and standard deviation's product. The Pearson correlation is often denoted with the symbol r and formally written as

$$r_{\boldsymbol{x},\boldsymbol{y}} = \frac{cov(\boldsymbol{x}, \boldsymbol{y})}{\sigma_{\boldsymbol{x}} \sigma_{\boldsymbol{y}}} = \frac{\sum_i (x_i - \overline{\boldsymbol{x}})(y_i - \overline{\boldsymbol{y}})}{\sqrt{\sum_i (x_i - \overline{\boldsymbol{x}})^2 (y_i - \overline{\boldsymbol{y}})^2}}, \tag{27}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ stand for the populations that are compared and $\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}$ their respective means.

The Spearman rank order correlation is defined as Pearson correlation for ranks, with ranks being the numerical ordering for the values of a population. The Spearman

[9] https://data.finlex.fi/ecli/kko/2012/3.html
[10] https://data.finlex.fi/ecli/kko/1989/15.html
[11] https://data.finlex.fi/ecli/kko/2004/120.html
[12] http://alt.qcri.org/semeval2019/

correlation is thus

$$r_{rank(\boldsymbol{x}),rank(\boldsymbol{y})} = \frac{cov(rank(\boldsymbol{x}), rank(\boldsymbol{y}))}{\sigma_{rank(\boldsymbol{x})}, \sigma_{rank(\boldsymbol{y})}}, \tag{28}$$

where $rank()$ is a ranking function that reduces a vector's values to their ordinal numbers. For instance, $rank([0.2, 0.21, 0.1, 0.99]) \mapsto [2, 3, 1, 4]$.

Spearman correlation is intuitively suited to the ranking task, as it measures rank correlations, but it does not consider the distance between two differently ranked values as Pearson correlation does. For the SemEval workshop for similarity evaluation in year 2014 [ABC+14], Agirre et al argue that Pearson has more informative value than Spearman, since Pearson takes into account both value and rank differences, whereas Spearman rank correlation does not consider the value differences. Agirre et al also performed analysis indicating that Pearson is better suited for gold standard comparison than Spearman.

Additionally, Hauke et al state that one must be careful not to overinterpret Spearman rank order correlation values [HK11] by showing that positive Spearman and negative Pearson correlation can co-occur. Due to the differences between the two popular correlation measures we use both metrics to provide insight on ranking correlation.

With the correlation measures being possibly inconsistent with each other, we include a simple non-correlation measure, mean squared error (MSE) in eq. (29) between target values $\boldsymbol{y} \in \mathbb{R}^m$ and predicted values $\hat{\boldsymbol{y}} \in \mathbb{R}^m$ to give additional insight on model performances.

$$MSE = \frac{1}{m}\sum_{i}^{m}(y_i - \hat{y_i})^2 \tag{29}$$

Before computing MSE, the scores are standardised to unit variance and zero mean for MSE to give intuitive values. Otherwise MSE would vary depending on differences in the models' embedding spaces.

As with topic classification, we train embedding models individually for each hyperparameter-preprocessing combination. However, for comparison against gold standard, we use all documents in the Finnish case law data set to train our embedding models in order for the embeddings to be as good as possible. We then compute similarity for document embedding pairs with two methods that are described in section 3.4. We evaluate these methods separately for each document embedding

model by computing Pearson correlation, Spearman rank correlation and MSE using gold standard labels. Additionally, for our linear regression ensemble, which learns weights for each model based on gold standard similarities, we apply 5-fold cross-validation to ensure that the ensemble's weights are generalisable beyond the gold standard set. 5-fold cross-validation, i.e. splitting data into 5 sets and using each in turn as a test set and the rest as training set, is also performed for the mean ensemble and individual models for comparison.

5.3 Hand-labelling similarity values

In order to evaluate computed similarity values with correlation to gold standard similarities and to use our learning to rank with logistic regression, we need gold standard ground truth similarities. Hang Li [L+09], in his book "Learning to Rank for Information Retrieval and Natural Language Processing", states that there are two common ways of creating this kind of ground truth data. One is to extract preferences from click-through data of people's Internet usage. This method, however, is out of question, since there is not click-through data available for our specific case of finding relevant Finnish case law documents with full document queries. The remaining method is to use humans to explicitly define similarities, i.e. labelling similarity values manually. This, as previously discussed, can be very time-consuming.

Besides being time-intensive, acquiring reliable similarity annotations is not always straightforward. When judging document similarity, it might not be obvious for an annotator how to label the similarities for documents well, no matter how refined their expertise. Additionally, people can have different opinions on similarity. Therefore, it is reasonable to devise an intelligent system that is easy to use, which leverages preferably more than one annotator to acquire similarity labels for a gold standard set.

In SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity [ADCGA12], Agirre et al used a scalar scale from 0 to 5 with explanations for each value, e.g. "(2) Not equivalent, but share some things" and "(1) Not equivalent, but are on the same topic". They annotated the data themselves and state that the pairwise Pearson correlation among the annotators was between 84% to 87% and that each annotator agreed with less than 90% of the other annotators' average scores.

For a subsequent year's task [ACD+13], Agirre et al used crowd sourcing to obtain the annotations in addition to the task organisers' own gold standards. They used

a similar 6 value scale for the semantic similarity and provided the annotators a detailed instruction on how to score the texts. The crowd sourced annotators were tested by comparing their results to the organisers' gold standard and the worst performers' annotations were dropped. Regardless of the crowd source pruning, the Pearson correlation between human assigned similarities ranged from 62% to 87%. This shows that crowd sourcing, while it is efficient, will make it difficult to achieve good ground truths for machine learning models. Agirre et al's experiments show that there is room for improvement in making gold standards for textual similarity, but more importantly, they emphasise the importance of having more than one annotator. If the annotators do not agree with each others' semantic evaluation, the average of many annotators' scores is likely less biased than having merely one annotator's opinion on the similarities.

Nevertheless, in order to obtain a gold standard semantic similarity evaluation set, an annotator is required to compare pairs of documents and assess their similarity. To alleviate this process, we incorporate comparing and evaluating case law similarity within our self-built case law finder web application. The evaluation within the application is made possible by having an optional sign up and login, which allows a user to evaluate the similarity of any result for a query document. Having a login required for evaluation allows selecting verified evaluators from others, as well as excluding completely incoherent evaluations, i.e. random or seemingly dishonest submissions. The application is described in more detail in appendix A.

Inspired by the SemEval's system, we use a 0 to 5 scale for similarity, minimally tailored for the Finlex case law corpus:

- 5 - Almost identical.
- 4 - Similar topics and content.
- 3 - Multiple shared topics.
- 2 - At least one common topic.
- 1 - Some common elements.
- 0 - Completely different.

While this similarity system does not deal with the aforementioned multiple forms of similarity problem, being based chiefly on topic similarity, its simplicity is beneficial in order to make submitting similarity ratings an easy task.

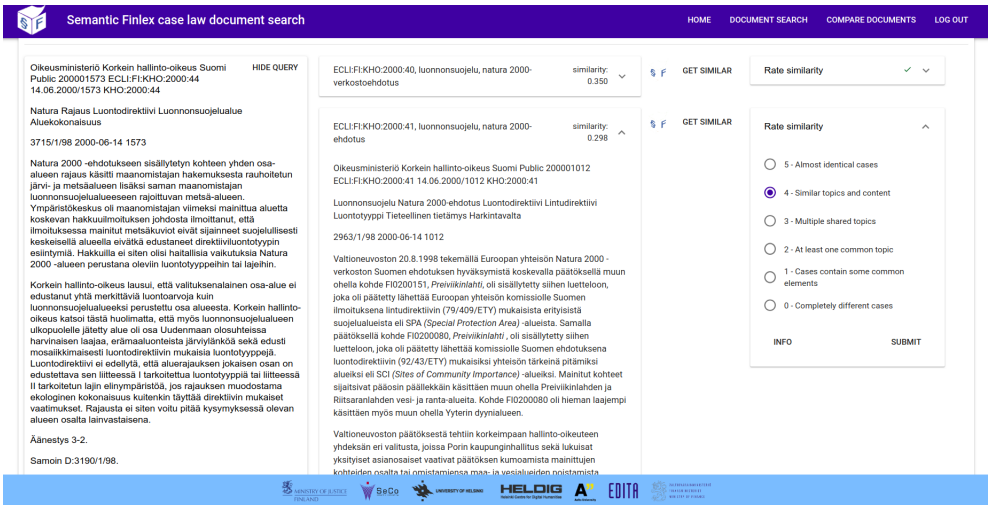

Figure 8: Rating similarity in the Finnish case law finder application. Green tick indicates a query-result pair similarity has been rated.

Submitted ratings contain a timestamp, user id and document ids, to enable the usage of average user-given similarities for document pairs as a gold standard for evaluation.

The final similarity rating set obtained using the application's submission include 138 distinct ratings in total with pairs from various topics. Due to difficulty in acquiring expert labels (likely a result of lack of advertising or not providing a materialistic reward for rating), 129 of the labels are assigned by myself, while 9 are given by a volunteering law student.

With the gold standard being constructed mostly by a single person and not someone more familiar with case law, or better yet, multiple such people, our ground truth for evaluating Finnish case law ranking leaves plenty of room for improvement. However, in our defence, 2017 SemEval task [CDA$^+$17] in multi-lingual text similarity, uses 250 pairs for each language, which are either constructed by a single expert or non-experts via crowd-sourcing. Additionally, regarding our gold standard set size, Campr et al [CJ15] had three annotators manually label only 150 pairs of summaries in order to compare the accuracy of various similarity computation models.

6 Results

This section describes the results of our document embedding model evaluation. As explained in more detail in section 5, we evaluate ranking for information retrieval by two methods, topic classification and similarity correlation with gold standard set.

We evaluate document embedding models separately for different hyperparameter-preprocessing combinations for both our evaluation methods. This enables us to see which hyperparameters are optimal for our task and whether the same model settings should be used for the two evaluation methods. We use accuracy measures, namely precision, recall and F1-score in eqs. (25) and (26) as our measures for topic classification. Whereas for similarity correlation with gold standards, we use Pearson correlation, Spearman rank order correlation and mean squared error (MSE) in eqs. (27), (28) and (29).

6.1 Topic classification using keywords

The best topic classification results per model can be seen in Table 2. The table shows that the traditional TF-IDF embeddings performed the best, closely followed by neural network model embeddings. LDA, while being a topic model by definition, performed significantly worse than the other models in providing embeddings for classifying keywords to topics.

Model	Precision	Recall	F1-score
TF-IDF	0.63	0.53	0.54
LDA	0.41	0.23	0.27
Word2Vec	0.55	0.46	0.49
Doc2Vec	0.54	0.56	0.53
Doc2VecC	0.60	0.48	0.51

Table 2: Best topic classification results for each tested model. Note that the values for precision, recall and F1-score are averages weighed by true instance count for each label. This can cause F1-score to be lower than both precision or recall as harmonic mean of two values weighs the lower value more heavily.

For TF-IDF, there was no hyperparameters to tune and for LDA we tuned only the

number of latent topics, which determines the size of LDA's embeddings. On the other hand, neural network methods had multiple hyperparameters to tune. Investigating around the default hyperparameters for the embedding model implementations, the best F1-scores for models were produced by hyperparameters displayed in Table 3.

Model	embedding size	model variant	window size	RSR
LDA	500	-	-	-
Word2Vec	900	C-BoW	10	-
Doc2Vec	100 / 300	D-BoW	5 / 10	-
Doc2VecC	500 / 700	C-BoW	10	0.0125

Table 3: Optimal hyperparameters for machine learning models in topic classification. The slash symbol "/" between two options denotes the options resulting in equal performance for model evaluation. RSR is the random sampling rate for context words in Doc2VecC model.

The best scores for preprocessing steps we tested are presented in Table 4. The hyperparameters for the best score for a given preprocessing step varied, but only slightly. We find that each preprocessing method is either of little use or ineffective for the embedding models with LDA as exception. LDA benefited from each preprocessing step and greatly from lemmatisation and query expansion.

Model	b	b, ns	b, lem	b, lem, ns	b, lem, qe
TF-IDF	0.54	0.54	0.53	0.53	0.53
LDA	0.15	0.17	0.22	0.23	0.27
Word2Vec	0.46	0.48	0.49	0.49	-
Doc2Vec	0.52	0.53	0.52	0.53	-
Doc2VecC	0.48	0.51	0.51	0.51	-

Table 4: F1-scores of test set evaluation for models and preprocessing methods in topic classification. Title shorthands are: b=basic preprocessing, ns=no stopwords, lem=lemmatised, qe=query expansion. The F1-scores are obtained using best model hyperparameters for each preprocessing method. Basic preprocessing includes tokenisation, punctuation removal and abbreviation expansion.

6.2 Gold standard correlation

Contrary to results in topic classification, we found texts should not be lemmatised for training with LDA as an exception. However, lemmatising texts before embedding them improved model performance significantly for all models apart from the worst performing model, Word2Vec. Additionally, unlike in topic classification, Doc2Vec and LDA perform the best from individual models. With the gold standard correlation method, we also evaluated two ensemble models. These outperform all of the individual models as can be seen in Table 5, which includes measurements for lemmatised and unlemmatised texts. As with topic classification, LDA benefited from query expansion for training data while TF-IDF was unaffected.

Regarding stopword removal, we found initially mixing results for its usefulness. However, after keeping the stopwords "yli" (over) and "ei" (no) in the texts while removing other stopwords, we found that stopword removal was beneficial for all models.

Model	Pearson	Spearman	MSE
TF-IDF	0.57 (0.42)	0.56 (0.50)	0.85 (1.17)
LDA	0.62 (0.46)	0.60 (0.48)	0.76 (1.07)
Word2Vec	0.42 (0.54)	0.41 (0.52)	1.16 (0.93)
Doc2Vec	0.64 (0.48)	0.64 (0.46)	0.71 (1.03)
Doc2VecC	0.56 (0.53)	0.53 (0.52)	0.89 (0.94)
Mean ensemble	0.70 (0.62)	0.69 (0.62)	0.61 (0.76)
LinReg ensemble	0.75 (0.70)	0.74 (0.70)	0.5 (0.6)

Table 5: Best correlations and mean squared error between gold standard similarities and embeddings' cosine similarity values for tested hyperparameters and preprocessing steps. Results for embeddings from lemmatised texts are without brackets and ones without lemmatisation in round brackets.

The best hyperparameters per model are depicted in Table 6. Comparing these to the best hyperparameters for topic classification in Table 3, we see that there is correlation with the results, but also differences in embedding sizes. LDA performs best in similarity correlation with less latent topics than in topic classification, whereas Doc2Vec benefits from a bigger vector size.

TF-IDF	LDA	Doc2Vec	Doc2VecC	Word2Vec
0.16	0.12	0.66	0.06	0.00

Table 7: Model weights for best performing ensemble

Model	embedding size	model variant	window size	RSR
LDA	300	-	-	-
Word2Vec	900	C-BoW	10	-
Doc2Vec	500	D-BoW	5 / 10	-
Doc2VecC	700	C-BoW	10	0.0125

Table 6: Optimal hyperparameters for machine learning models. The slash symbol "/" between two options denotes the options resulting in equal performance for model evaluation. RSR denotes the random sampling rate for context words in Doc2VecC model.

As our best performing model is linear regression ensemble, which computes weights for each individual model, we further analysed the weights, to see which model's are deemed the most important by the regression. By examining the regression weights in Table 7, we find that Doc2Vec is by far the most weighed model contributing to over half of the ensemble's predicted similarity. On the other hand, the other neural net models seem less important with Word2Vec contributing nothing to the ensemble and Doc2Vec less than TF-IDF or LDA.

Besides using cosine similarity to extract similarity from document embeddings, we tested using logistic regression to learn similarities using ground truth values. The results can be seen in Table 8, which shows that the correlations for test sets are remarkably lower than the ones computed using cosine similarity. Most training set results improve upon cosine similarity and especially the ones for ensembles indicating that our learning to rank model has potential with the Finnish case law corpus if given more similarity targets to train with.

Model	Pearson	Spearman	MSE
TF-IDF	0.24 [0.62] (0.47)	0.27 [0.67] (0.37)	1.52 [0.76] (1.06)
LDA	0.17 [0.54] (0.58)	0.14 [0.56] (0.51)	1.66 [0.92] (0.83)
Word2Vec	0.25 [0.62] (0.44)	0.13 [0.67] (0.40)	1.50 [0.72] (1.13)
Doc2Vec	0.31 [0.69] (0.58)	0.36 [0.74] (0.58)	1.38 [0.62] (0.84)
Doc2VecC	0.40 [0.50] (0.59)	0.25 [0.47] (0.51)	1.20 [1.01] (0.81)
Mean Ensemble	0.41 [0.78] (0.68)	0.39 [0.81] (0.65)	1.18 [0.44] (0.64)
LinReg Ensemble	0.44 [0.81] (0.68)	0.36 [0.83] (0.65)	1.13 [0.38] (0.64)

Table 8: Average correlations and mean squared error between gold standard similarities and embeddings' learned similarity values for 5-fold cross validation. Cross-validation training average is shown in square brackets and cosine similarity average for test sets is shown in round brackets for comparison.

7 Conclusions and Future Work

We have evaluated five models' performance in ranking for Finnish case law retrieval, namely, the performance of TF-IDF, LDA, Word2Vec, Doc2Vec and Doc2VecC. Additionally two ensembles of the models were evaluated. Two different evaluation methods were reviewed, topic classification and similarity correlation against a self-provided gold standard similarity set.

Returning to our research questions, we found that the neural network Doc2Vec's embeddings outperform those of LDA and TF-IDF for use in Finnish case law ranking. However, Doc2VecC's performance is worse than LDA and also slightly worse than TF-IDF. Averaged Word2Vec's embeddings was the most useless embedding model of the tested models. For model combinations, the evaluation results show that a linear regression weighted combination of presented models fairs best for ranking the case law documents. Additionally, we find that a mean ensemble of the models beats all the individual models and that the weighted ensemble outperforms the individuals models significantly. As per inspecting the regression model's weights, Doc2Vec gave the most weighted similarities, which is not unexpected as it also is the best individual model. However, Word2Vec carried no weight and is thus redundant for our case law retrieval. Regardless of Word2Vec's performance, our results show that neural networks can perform better than TF-IDF and LDA for Finnish case law retrieval. Moreover, we found that a combination of all models except Word2Vec is the most useful for our retrieval setting.

Regarding our tests with preprocessing methods, we see that the case law texts should be lemmatised only for trained models (apart from LDA) with stopwords removed during model training. Rather interestingly, only LDA benefits from lemmatisation for model training data. The probabilistic model's performance is also boosted with synonym and hypernym query expansion making it the second best individual model behind the neural network Doc2Vec. Additionally, we found that removing the stopwords "yli" (over) and "ei" (no) was harmful for ranking with embeddings. This confirms that stopword removal should be performed carefully as discussed in section 2.2.1.

The correlation between our human-assigned similarities and those from our best model is highly promising. We reach 0.75 for both Pearson and Spearman rank order correlations, which is more than might be for human evaluator's similarity assignments amongst themselves [ABC+14]. Our gold standard similarity set would,

however, require more annotators and annotated document pairs to give a robust indication of human-like similarity assignment performance for the tested models. We also tested a learning to rank method for computing similarities from document embeddings, however, the results are not in its favour. It seems that our gold standard similarity set is not large enough for logistic regression to give generalisable results that could compete against cosine similarity in extracting similarity values between document embeddings.

Considering our final research question about topic classification evaluation for IR ranking, we find that it is rather unreliable based on the differences in evaluation results compared to the more authoritative evaluation measure for ranking, i.e., correlation with gold standard similarities. However, it should be noted, that neither our gold standard set nor the keywords for topic classification targets are optimal, which leaves room for errors in the evaluation. Thus, while the differences between the two methods are too notable to ignore, further investigation using more refined evaluation targets as well as using other datasets would provide more definitive results.

In addition to model comparison for ranking and assessing topic classification as an evaluation method for information retrieval, we present an information retrieval application for Finnish case law. The application enables users to query the Finnish case law efficiently with full texts by allowing direct text input and uploading from a file. As a consequence of query text preprocessing and the embedding based result ranking, free natural texts queries work rather well in the application, although this performance is not tested by any numerical measure but only by reviewing such queries' results manually. The created application is briefly described in appendix A.

As for future work concerning ranking improvement with neural network models, Che et al introduce ELMo [CLW+18], a new state-of-the-art word embedding model that, contrary to our models, as well as other popular methods, encodes words with a sentence as an input instead of just the word. In other words, the model's word representations are a function of a whole sentence. This has the benefit of having different embeddings for homonyms, i.e. words with the same form but distinct meanings, so that the model can disambiguate between such terms.

In their paper, Che et al test the performance of their ELMo vectors in six different NLP tasks and improve the results in each task by adding ELMo to models previously used in the tasks. This shows that the ELMo generated embeddings capture

semantic information better than the other word representations that were used in the tested NLP task models. Besides simply testing averaged ELMo embedding's capabilities for ranking, Doc2VecC's idea of adding random context words during training can be tested with ELMo to see whether it could help provide better document averages as happened with Word2Vec. While ELMo is a model much more complicated than Word2Vec, adding context words can be done by manipulating the model inputs. However, ELMo is sensitive to word ordering in sentences, which might provide challenges in adding context words.

Another task for the future is to evaluate for ranking the Word Mover's Distance (WMD) [KSKW15] model, which provides a similarity metric for documents by leveraging predefined word vectors. True to its name, WMD computes a similarity value for a document pair from distances between vector's that correspond to the documents' words with the following assumption: The less the two documents' vectors have to be manipulated to become the same, the higher the document pair's similarity value.

A freely available WMD model was tested, but it resulted being far too slow for information retrieval purposes. In preliminary testing, computing distance, i.e. similarity, between two documents took up to several minutes for small word vectors of 100 elements. However, in 2017 Atasu et al presented a Linear-Complexity Relaxed Word's Mover's Distance (LC-RWDM) [APD+17, AM19], an efficient version of Kusner et al's WMD. LC-RWDM improves the original WMD's quadratic time-complexity $O(W_d^2)$ by applying several heuristic optimisation techniques resulting in a linear time complexity model. With an implementation of LC-RWMD, computing document distances could be feasible for an IR application. This would also provide an interesting way to evaluate different word embedding models, including the aforementioned ELMo, for ranking.

References

ABC+14 Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G. and Wiebe, J., Semeval-2014 task 10: Multilingual semantic textual similarity. *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pages 81–91.

AC18 Ash, E. and Chen, D. L., Case vectors: Spatial representations of the law using document embeddings. *Social Science Research Network*.

ACD+13 Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A. and Guo, W., * sem 2013 shared task: Semantic textual similarity. *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, 2013, pages 32–43.

AD17 Azad, H. and Deepak, A., Query expansion techniques for information retrieval: a survey.

ADCGA12 Agirre, E., Diab, M., Cer, D. and Gonzalez-Agirre, A., Semeval-2012 task 6: A pilot on semantic textual similarity. *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2012, pages 385–393.

AM19 Atasu, K. and Mittelholzer, T., Linear-complexity data-parallel earth mover's distance approximations. *International Conference on Machine Learning*, 2019, pages 364–373.

APD+17 Atasu, K., Parnell, T., Dünner, C., Sifalakis, M., Pozidis, H., Vasileiadis, V., Vlachos, M., Berrospi, C. and Labbi, A., Linear-complexity relaxed word mover's distance with gpu acceleration. *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pages 889–896.

Bat12 Bates, M. J. *Understanding Information Retrieval Systems*, chapter 2. Taylor & Francis Group, Boca Raton: USA, 2012.

BCA+17 Bethard, S., Carpuat, M., Apidianaki, M., Mohammad, S. M., Cer, D. and Jurgens, D., Proceedings of the 11th international workshop on semantic evaluation (semeval-2017). *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017, URL `http://www.aclweb.org/anthology/S17-2000`.

BCC Büttcher, S., Clarke, C. L. and Cormack, G. V. *Information retrieval: Implementing and evaluating search engines*.

BGG18 Basu, M., Ghosh, S. and Ghosh, K., Overview of the fire 2018 track: Information retrieval from microblogs during disasters (irmidis). *Proceedings of the 10th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE'18, New York, NY, USA, 2018, ACM, pages 1–5, URL `http://doi.acm.org/10.1145/3293339.3293340`.

BGLB16 Beel, J., Gipp, B., Langer, S. and Breitinger, C., Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17,4(2016), pages 305–338. URL `https://doi.org/10.1007/s00799-015-0156-0`.

BH00 Basheer, I. A. and Hajmeer, M., Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43,1(2000), pages 3–31.

Bis06 Bishop, C. M. *Pattern recognition and machine learning*, chapter 2.4. springer, 2006.

BKL09 Bird, S., Klein, E. and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*, chapter preface. " O'Reilly Media, Inc.", 2009.

BL17 Berger, A. and Lafferty, J., Information retrieval as statistical translation. *SIGIR Forum*, 51,2(2017), pages 219–226. URL `http://doi.acm.org/10.1145/3130348.3130371`.

BNJ02 Blei, D. M., Ng, A. Y. and Jordan, M. I., Latent dirichlet allocation. *Advances in neural information processing systems*, 2002, pages 601–608.

BNJ03 Blei, D. M., Ng, A. Y. and Jordan, M. I., Latent dirichlet allocation. *Journal of machine Learning research*, 3,Jan(2003), pages 993–1022.

Bra03 Brants, T., Natural language processing in information retrieval. *CLIN*, 2003.

BS16 Brychcín, T. and Svoboda, L., Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pages 588–594.

BWG10 Bengio, S., Weston, J. and Grangier, D., Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems 23*, Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S. and Culotta, A., editors, Curran Associates, Inc., 2010, pages 163–171, URL http://papers.nips.cc/paper/4027-label-embedding-trees-for-large-multi-class-tasks.pdf.

BYRN+99 Baeza-Yates, R., Ribeiro-Neto, B. et al., volume 463. ACM press New York, 1999.

CDA+17 Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I. and Specia, L., Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Cer17 Cerezo, M., Infoxication and information overload: In due measure without overdoing it, 2017. URL https://www.gmv.com/blog_gmv/language/en/infoxication-and-information-overload-in-due-measure-without-overdoing-it/.

Che17 Chen, M., Efficient vector representation for documents through corruption. *5th International Conference on Learning Representations*.

Chr18 Chris, A., Top 10 search engines in the world, 2018. URL https://www.reliablesoft.net/top-10-search-engines-in-the-world/.

CJ15 Campr, M. and Ježek, K., Comparing semantic models for evaluating automatic document summarization. *International Conference on Text, Speech, and Dialogue*. Springer, 2015, pages 252–260.

CLW+18 Che, W., Liu, Y., Wang, Y., Zheng, B. and Liu, T., Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium, October 2018, Association for Computational Linguistics, pages 55–64, URL http://www.aclweb.org/anthology/K18-2005.

CN15 Chiu, J. P. C. and Nichols, E., Named entity recognition with bidirectional lstm-cnns. *CoRR*, abs/1511.08308. URL http://arxiv.org/abs/1511.08308.

CR12 Carpineto, C. and Romano, G., A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44,1(2012), page 1.

DOL15 Dai, A. M., Olah, C. and Le, Q. V., Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*.

Eft96 Efthimiadis, E. N., Query expansion. *Annual review of information science and technology (ARIST)*, 31, pages 121–87.

EU17 EU, Ecli-eu, 2017. URL `https://e-justice.europa.eu/content_european_case_law_identifier_ecli-175-en.do`.

FBY92 Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

GBC16 Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Gib17 Gibney, E., Google reveals secret test of ai bot to beat top go players. *Nature*, 541,7636(2017), page 142. URL `http://dblp.uni-trier.de/db/journals/nature/nature541.html#Gibney17`.

GL14 Goldberg, Y. and Levy, O., word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

GRG18 Gudivada, V. N., Rao, D. L. and Gudivada, A. R., Chapter 11 - information retrieval: Concepts, models, and systems. In *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, Gudivada, V. N. and Rao, C., editors, volume 38 of *Handbook of Statistics*, Elsevier, 2018, pages 331 – 401, URL `http://www.sciencedirect.com/science/article/pii/S0169716118300245`.

GTS+16 Gurulingappa, H., Toldo, L., Schepers, C., Bauer, A. and Megaro, G., Semi-supervised information retrieval system for clinical decision support. *TREC*, 2016.

GZH+17 Gui, T., Zhang, Q., Huang, H., Peng, M. and Huang, X., Part-of-speech tagging for twitter with adversarial neural networks. *Proceedings of the*

2017 Conference on Empirical Methods in Natural Language Processing, 2017, pages 2411–2420.

Har54 Harris, Z. S., Distributional structure. *Word*, 10,2-3(1954), pages 146–162.

HK11 Hauke, J. and Kossowski, T., Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30,2(2011), pages 87–93.

HRJM15 Harispe, S., Ranwez, S., Janaqi, S. and Montmain, J., Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8,1(2015), pages 1–254.

Hua08 Huang, A., Similarity measures for text document clustering. *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, 2008, pages 49–56.

HVK+05 Hyvönen, E., Valo, A., Komulainen, V., Seppälä, K., Kauppinen, T., Ruotsalo, T., Salminen, M. and Ylisalmi, A., Finnish national ontologies for the semantic web - towards a content and service infrastructure. 11 2005.

ILS16 Ibrahim, O. A. S. and Landa-Silva, D., Term frequency with average term occurrences for textual information retrieval. *Soft Computing*, 20,8(2016), pages 3045–3061.

KLJJ04 Korenius, T., Laurikkala, J., Järvelin, K. and Juhola, M., Stemming and lemmatization in the clustering of finnish text documents. *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 2004, pages 625–633.

Klö17 Klöhn, A., Libratus Poker AI Beats Humans for $1.76m, Is End Near? *PokerListings*. URL `http://www.pokerlistings.com/libratus-poker-ai-smokes-humans-for-1-76m-is-this-the-end-42839`.

KOM03 Koehn, P., Och, F. J. and Marcu, D., Statistical phrase-based translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pages 48–54.

KSCK19 Kim, D., Seo, D., Cho, S. and Kang, P., Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec. *Information Sciences*, 477, pages 15–29.

KSK16 Kuzi, S., Shtok, A. and Kurland, O., Query expansion using word embeddings. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, New York, NY, USA, 2016, ACM, pages 1929–1932, URL `http://doi.acm.org/10.1145/2983323.2983876`.

KSKW15 Kusner, M., Sun, Y., Kolkin, N. and Weinberger, K., From word embeddings to document distances. *International Conference on Machine Learning*, 2015, pages 957–966.

KZS+15 Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R. and Fidler, S., Skip-thought vectors. *CoRR*, abs/1506.06726. URL `http://arxiv.org/abs/1506.06726`.

L+09 Liu, T.-Y. et al. *Learning to rank for information retrieval*, volume 3, pages 225–331. Now Publishers, Inc., 2009.

LB02 Loper, E. and Bird, S., Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

LB16 Lau, J. H. and Baldwin, T., An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

LGD15 Levy, O., Goldberg, Y. and Dagan, I., Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3, pages 211–225.

LH13 Li, B. and Han, L., Distance weighted cosine similarity measure for text classification. *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2013, pages 611–618.

LM14 Le, Q. and Mikolov, T., Distributed representations of sentences and documents. *International Conference on Machine Learning*, 2014, pages 1188–1196.

LWHM16 Landthaler, J., Waltl, B., Holl, P. and Matthes, F., Extending full text search for legal document collections using word embeddings. *JURIX*, 2016, pages 73–82.

Mäk16 Mäkelä, E., LAS: an integrated language analysis tool for multiple languages. *The Journal of Open Source Software*, 1,6(2016). URL `http://dx.doi.org/10.21105/joss.00035`.

MBB+14 Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S. and Zamparelli, R., Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pages 1–8.

MBXS17 McCann, B., Bradbury, J., Xiong, C. and Socher, R., Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107. URL `http://arxiv.org/abs/1708.00107`.

MCCD13 Mikolov, T., Chen, K., Corrado, G. and Dean, J., Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

MJ09 Martin, J. H. and Jurafsky, D. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, chapter 2. Pearson/Prentice Hall Upper Saddle River, 2009.

MRS08 Manning, C. D., Raghavan, P. and Schütze, H. *Introduction to Information Retrieval*, chapter 2. Cambridge University Press, New York, NY, USA, 2008.

MSC+13 Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J., Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546. URL `http://arxiv.org/abs/1310.4546`.

OTM+17 Oksanen, A., Tuominen, J., Mäkelä, E., Tamper, M., Hietanen, A. and Hyvönen, E., Law and justice as a linked open data service. 2017. Submitted.

Pir15 Pirinen, T. A., Omorfi—free and open source morphological lexical database for finnish. *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, 2015, pages 313–315.

PNI+08 Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P. and Welling, M., Fast collapsed gibbs sampling for latent dirichlet allocation. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pages 569–577.

PSM14 Pennington, J., Socher, R. and Manning, C., Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pages 1532–1543.

QAC12 Quercia, D., Askham, H. and Crowcroft, J., Tweetlda: Supervised topic classification and link prediction in twitter. *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, New York, NY, USA, 2012, ACM, pages 247–250, URL `http://doi.acm.org/10.1145/2380718.2380750`.

QLXL10 Qin, T., Liu, T.-Y., Xu, J. and Li, H., Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13,4(2010), pages 346–374.

RG65 Rubenstein, H. and Goodenough, J. B., Contextual correlates of synonymy. *Communications of the ACM*, 8,10(1965), pages 627–633.

Ril95 Riloff, E., Little words can make a big difference for text classification. *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1995, pages 130–136.

Rob04 Robertson, S., Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60,5(2004), pages 503–520.

Ron14 Rong, X., word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

S+01 Singhal, A. et al., Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24,4(2001), pages 35–43.

Sah08 Sahlgren, M., The distributional hypothesis. *Italian Journal of Disability Studies*, 20, pages 33–53.

SAY+16 Shin, J.-H., Abebe, M., Yoo, C. J., Kim, S., Lee, J. H. and Yoo, H.-K., Evaluating the effectiveness of the vector space retrieval model indexing. In *Advances in Computer Science and Ubiquitous Computing*, Springer, 2016, pages 680–685.

SJ72 Sparck Jones, K., A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28,1(1972), pages 11–21.

SL10 Sakai, T. and Lin, C.-Y., Ranking retrieval systems without relevance assessments: Revisited. *EVIA@ NTCIR*, 2010, pages 25–33.

SL12 Seber, G. A. and Lee, A. J. *Linear regression analysis*, volume 329, chapter 1.3. 2012.

SLW10 Shi, Z., Li, P. and Wang, B., Using clustering to improve retrieval evaluation without relevance judgments. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 2010, pages 1131–1139.

Smi97 Smith, S. W., *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997. URL `http://www.dspguide.com`. Available at www.dspguide.com.

Smi07 Smith, R., An overview of the tesseract ocr engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, Washington, DC, USA, 2007, IEEE Computer Society, pages 629–633, URL `http://dl.acm.org/citation.cfm?id=1304596.1304846`.

SNC01 Soboroff, I., Nicholas, C. and Cahan, P., Ranking retrieval systems without relevance judgments. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pages 66–73.

Spo07 Spoerri, A., Using the structure of overlap between search results to rank retrieval systems without relevance judgments. *Information Processing & Management*, 43,4(2007), pages 1059–1070.

SRLK16 Silfverberg, M., Ruokolainen, T., Lindén, K. and Kurimo, M., Finnpos: an open-source morphological tagging and lemmatization toolkit for

finnish. *Language Resources and Evaluation*, 50,4(2016), pages 863–878.

Sta13 StatSoft, I., *Electronic statistics textbook*. 2013. URL `http://www.statsoft.com/textbook/`.

SWW+18 Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R. and Carin, L., Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *CoRR*, abs/1805.09843. URL `http://arxiv.org/abs/1805.09843`.

SWY75 Salton, G., Wong, A. and Yang, C. S., A vector space model for automatic indexing. *Commun. ACM*, 18,11(1975), pages 613–620. URL `http://doi.acm.org/10.1145/361219.361220`.

TSM15 Tai, K. S., Socher, R. and Manning, C. D., Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075. URL `http://arxiv.org/abs/1503.00075`.

Tve77 Tversky, A., Features of similarity. *Psychological review*, 84,4(1977), page 327.

Ull11 Ullman, J. D., *Mining of massive datasets*. Cambridge University Press, 2011.

Urd93 Urdang, L., The oxford thesaurus: an az dictionary of synonyms.

VBB+18 Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, L., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N. and Uszkoreit, J., Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416. URL `http://arxiv.org/abs/1803.07416`.

VRCB15 Vylomova, E., Rimell, L., Cohn, T. and Baldwin, T., Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. *arXiv preprint arXiv:1509.01692*.

WC03 Wu, S. and Crestani, F., Methods for ranking information retrieval systems without relevance judgments. *Proceedings of the 2003 ACM symposium on Applied computing*. ACM, 2003, pages 811–816.

WSC+16 Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. et al., Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

ZLK+18 Zhou, B., Lapedriza, A., Khosla, A., Oliva, A. and Torralba, A., Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40,6(2018), pages 1452–1464.

ZSLL15 Zhou, C., Sun, C., Liu, Z. and Lau, F. C. M., A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630. URL `http://arxiv.org/abs/1511.08630`.

A Case Law Finder Application

We present a web application leveraging the text embedding models studied in this work to allow effective full-document search of Finnish case law. The application is made available online[13] for free use. We briefly describe the created application's main components, architecture and user interface.

The application's data is stored in a relational database with a table for documents to include document texts, metadata and an integer document id, which corresponds to a document's index in training data for embedding models. The database also includes tables for users and similarity ratings to enable users to rate document pair similarities within the application. The chief goal of the application is to enable efficient and precise search with full texts. However, text documents come in various formats. For instance, a user might have a case law text in paper or PDF. Since papers can be photographed and PDFs may contain text as image, we include object character recognition (OCR) in the application to extract text from image files.

Tesseract OCR [Smi07], an open source OCR system that has a well performing pre-trained model for Finnish text, good documentation and the possibility of retraining the model further, was chosen for the task. Although having a model for Finnish out-of-the-box, Tesseract OCR was not directly implemented into the software. Instead, it was first retrained to include letters "Å", "å" and the section sign "§", which were not included in the Tesseract OCR's readily available Finnish text model.

Within the web application, a user is able to input a text document with which to query FinLex case law. This text document may be uploaded to the application's search form from a file, or it can be directly written to the form as text. Supported file formats for uploading documents are plain text, XML, PDF, and with Tesseract OCR, image formats, e.g. JPEG or PNG. The search form also allows a user to choose the algorithm that ranks the documents, since ranking with some algorithms may work better than others for certain topics, or depending on what kind of relatedness is preferred.

[13] https://data.finlex.fi/document-search

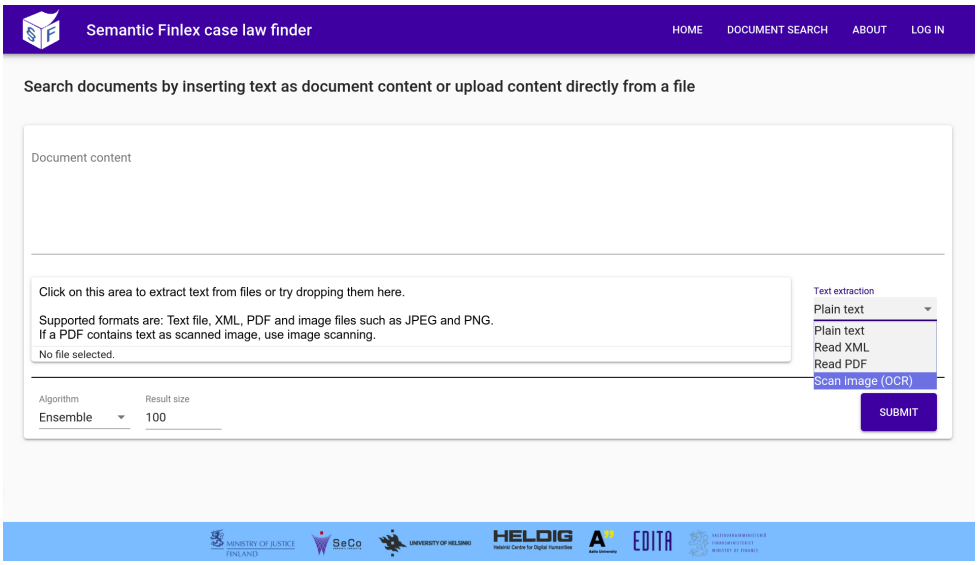

Figure 9: Semantic FinLex case law finder application document search.

Once a document is submitted in document search form, it is sent to the application back-end where the magic happens. The backend provides a simple API for retrieving case law documents. A query document is sent to the API via a HTTP/POST request where an embedding model is specified in the requested URI. An optional parameter n is provided to limit the number of retrieved documents, since sending the results via HTTP causes a bottleneck in retrieval time.

A received query document is preprocessed to the same format as models' training data and the formatted query text is given to the model as input. The model transforms its input into a vector, and cosine similarity values are computed between the query's vector representation and all database's case law documents' vector representations. Then, all document ids are sorted by the computed similarities and the top n ranked documents are retrieved from the database and returned in JSON-format. The query retrieval and processing is illustrated as a graph in Figure 10.

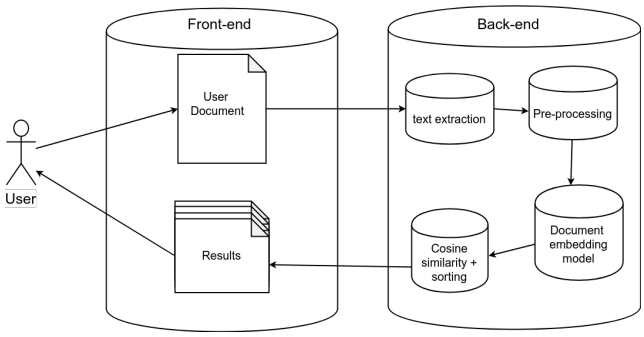

Figure 10: Semantic FinLex case law finder application architecture.

The ranked case law is shown to the user as a list of expandable panels with the case law identifier and keywords as a panel's title. Similarity rating is shown by default to give insight on how the similarities change. This allows the user to see from the values whether there are likely relevant results left to view. Besides the actual document text, the result items contain a button for quickly querying case law related to a result document. This is intended as a helpful measure when a user does not have a certain query document, but rather wants to search the case law corpus exploratively.

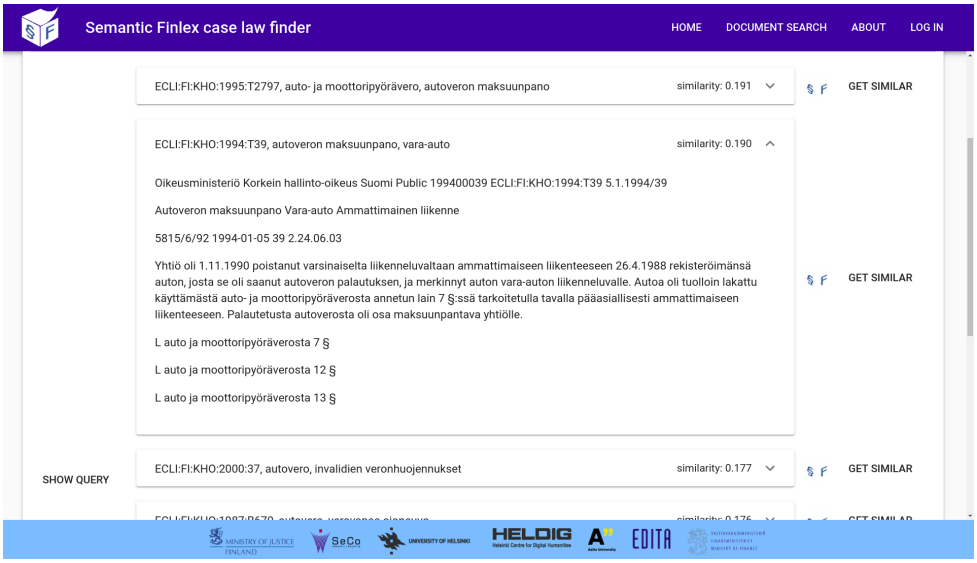

Figure 11: Semantic FinLex case law finder application document search results.

The created application and its models are intended to work as generally as possible. While the application is created for the specific domain of Finnish case law, there is little besides the trained models that restricts the application to be used with other corpora. Only regex abbreviation expansion, ties the textual context to the

Finnish language or juridical terminology. This suggests that the performance is generalisable to texts from other linguistic domains as well as other languages.

The models in the application leverage lemmatisation as it was deemed beneficial in model evaluation. Since lemmatisation normalises effectively word inflections, it can make queries of natural sentences less challenging for machines. Thus, we performed an additional test on our working application with short natural language queries to see how it would manage the task although the models are optimised for full document search. A working example is the query "törmäsin autoon" (I collided into a car). Without lemmatisation, the inflected word "törmäsin" (I collided) would be non-existent in the training data. Thus, even the neural net models would not be able to infer that the user is inquiring about collisions. As the result of our additional experiment we find that using the example query returns cases concerning cars and traffic accidents. However, we did not perform any extensive testing for short natural language queries as this was not our primary objective.