Master's thesis
Master's Programme in Data Science

Document-level embeddings and graph-augmented learning for Finnish articles

Olli Jokinen

June 2, 2024

Supervisor(s): Dr Lidia Pivovarova

Examiner(s): Associate Professor Arto Klami

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Olli Jokinen

Työn nimi — Arbetets titel — Title

Document-level embeddings and graph-augmented learning for Finnish articles

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 2, 2024	56

Tiivistelmä — Referat — Abstract

The rise of large language models (LLMs) has revolutionized natural language processing, particularly through transfer learning and fine-tuning paradigms that enhance the understanding of complex textual data. This thesis builds upon the concept of fine-tuning to improve the understanding of Finnish Wikipedia articles. Specifically, a BERT-based language model is fine-tuned to create high-quality document representations from Finnish texts. The learned representations are applied to downstream tasks, where the model's performance is evaluated against baseline models.

This thesis draws on the SPECTER paper, published in 2020, which introduced a training framework for fine-tuning a general-purpose document embedder. SPECTER was trained using a document-level training objective that leveraged document link information. Originally, SPECTER was designed for scientific articles, utilizing citations between articles. The training instances consisted of triplets of query, positive, and negative papers, with the aim of capturing the semantic similarity of the documents.

This work extends the SPECTER framework to Finnish Wikipedia data. While scientific articles have citations, Wikipedia's cross-references are used to build a document graph that captures the relatedness between articles. Additionally, Wikipedia data is publicly available as a full data dump, making it an attractive choice for the dataset in this thesis.

One of the objectives is to demonstrate the flexibility of the SPECTER framework on a new dataset that has a similar networked structure to that of scientific articles. The fine-tuned model can be used as a general-purpose tool for various tasks and applications; however, in this thesis, its performance is measured in topic classification and cross-reference ranking. The Transformer-based language model produces fixed-length embeddings, which are used as features in the topic classification task and as vectors to measure the L2 distance of article vectors in the cross-reference prediction task. This thesis shows that the proposed model, WikiSpecter, optimized with a document-level objective, outperformed baseline models in both tasks. The performance indicates that Finnish Wikipedia provides relevant cross-references that help the model capture relationships across a range of topics.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Natural language processing
→ Information extraction

Avainsanat — Nyckelord — Keywords

natural language processing, word embeddings, fine-tuning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**

2 Background **5**

2.1 A Brief Introduction to Neural Networks 5
2.2 Non-Contextual Word Embeddings 7
2.3 Contextual Word Embeddings 7
2.4 Transformers and Self-Attention 8
2.5 BERT 11
2.6 SPECTER 13
2.6.1 Model input 14
2.6.2 Initialized parameters 15
2.6.3 Embeddings and Inference 15
2.6.4 Loss and Training Objective 16
2.7 Related work 17

3 Data **19**

3.1 Data Selection and Comparison 19
3.2 Document Graph and Intuition on Data and Cross-References 22
3.2.1 Intuition 22
3.2.2 Triplet Sampling Process 25
3.3 Data Collection and Preprocessing 27
3.4 Training Data - Fine-Tuning 28
3.5 Evaluation Data - Topic Classification and Cross-Reference Prediction . 30

4 Experiments and Evaluation Framework **35**

4.1 Training and Implementation 35
4.2 Models 36
4.3 Evaluation Framework 37
4.4 Article Topic Classification 38
4.5 Cross-Reference Prediction 39

5 Results and Discussion **41**
5.1 Error Analysis of Misclassifications 41
5.2 Cross-Reference Prediction 44
5.3 Further Research and Potential Applications 45

6 Conclusions **47**

Bibliography **49**

Appendix A Relationship Between Top-Level to Second-Level Topics **53**

1. Introduction

Large language models (LLM) have emerged from the field of natural language processing (NLP). Recent development and research has been rapid and accelerating, especially after 2018 when NLP was empowered with frameworks and pretrained models [1],[2] that can be fine-tuned for task specific problems. This development enabled similar progress to NLP what ImageNet [3],[4] did to image processing almost 10 years before. Transfer learning and fine-tuning are the key paradigms enabling the evolution of performance in various tasks. This thesis contributes to the concept of using pretrained models in task-specific model training.

After BERT [1] was released, multiple benchmark results were exceeded using the same paradigm: a model is first pretrained on a massively large dataset and then used to initialize a new model which is fine-tuned for a specific problem with a smaller task-specific dataset. BERT was pretrained with a masked language modeling (MLM) objective, but it can also be fine-tuned for alternative objectives. One of the research papers that took an alternative approach was SPECTER (**S**cientific **P**aper **E**mbeddings using **C**itation informed Transform**ER**s) [5]. At that time, the authors came up with a new approach to train a model to describe documents as fixed length vectors. Their idea was to improve the quality of document embeddings by incorporating external information from other documents through a document graph, whereas pretrained models trained with language modeling objectives use only the in-document context to predict masked tokens. This thesis is inspired by that setting and uses many ideas and principles presented in SPECTER.

The objective of this thesis is to apply the SPECTER framework to train a new language model with a graph learning training objective in a different domain and linguistic context. The aim is to evaluate the applicability of SPECTER beyond the original language and domain for which it was designed. To achieve this, a new dataset from outside the scientific domain, but with similar textual components for building a document graph, is selected. Additionally, the training framework is adjusted to accommodate the Finnish language and general domain Wikipedia data. Next, I will list research questions of the thesis and then outline SPECTER and the document graph.

- RQ1: Is it possible to utilize the structure of Finnish Wikipedia data for training large language models (LLMs) in a similar fashion to SPECTER?
- RQ2: Given that the document graph and SPECTER's training framework are generic and data-agnostic, what can be said about SPECTER's applicability for different domains and linguistic contexts beyond the scientific literature?

SPECTER, published in 2020, is designed for encoding documents and creating document-level representations from scientific texts. This is obtained by fine-tuning a new model from pretrained SciBERT [6] (BERT for scientific texts). SciBERT was a natural choice for an initial checkpoint model as it already leverages scientific publications in pretraining. Although both models focus on the scientific domain, SPECTER differs from the previous models in the BERT family by its document-level training objective. The authors showed that this objective enables producing higher quality representations of documents compared to the models that use only token- or sentence-level objectives in training and generate document vectors by averaging concatenated vectors from smaller pieces of text.

The key component of the framework is a document graph. The graph is built by creating a map of related and unrelated documents in the dataset. The relatedness is defined by citations between articles where a citation suggests that the source document is related to the cited document. The graph-based approach for modeling the similarity of elements has been employed prior to SPECTER, as demonstrated in works like [7] and [8] by Dor et al. and Michihiro Yasunaga et al. Both papers utilized graphs to model sentence similarity for different tasks. In the first paper [7], the authors constructed a graph where sentences within the same section of Wikipedia are likely to be semantically similar, while sentences in different sections are considered more distantly related. In SPECTER, the training instances are constructed from triplets containing positive and negative paper for each query paper. The instances are sampled directly from the document graph.

In this thesis, the document graph is based on articles on Finnish Wikipedia and hyperlinks between articles. While references from one scientific paper to another are the basis for building a citation graph in SPECTER, analogously, this work uses ***cross references*** between Wikipedia articles to build a document graph. Intuition on building the document graph is demonstrated in Section 3 with examples. The experimentation part of the thesis proposes a new model called *WikiSpecter* which is fine-tuned with a triplets which are sampled from the document-graph.

The SPECTER model is fundamentally a general-purpose document encoder. For encoding new unseen documents, the document graph is not needed at inference time. The same applies to WikiSpecter. The fine-tuned model can be used as a tool

for various tasks and applications such as topic classification and cross-reference recommendation by applying the learned vector representations from one task to another.

During the evaluation, unseen documents are embedded as such and no citation data is required. The resulting embeddings are then used as model features in the topic classification task and as vectors to measure the *L2-distance* of document vectors in cross-reference prediction task. Topic classification and cross-reference prediction tasks are typical evaluation tasks in the context of evaluating machine learning (ML) models. The tasks for this thesis are selected from the SciDocs framework [5] which was developed to evaluate SPECTER. By applying the same evaluation framework to assess WikiSpecter performance, evidence is obtained supporting the generic nature of the SPECTER framework. The fine-tuned model is evaluated against two different baseline models, Finnish BERT [9] and Finnish Sentence BERT [10], which are trained with token- and sentence-level language modeling tasks. The expectation is that the model optimized for document-level tasks outperforms the baseline models in the evaluation tasks. This seems intuitive since the evaluation criteria measure the performance on *document-level* in particular. However, it is unclear whether other data sources can be applied in parallel.

Despite rapidly advancing NLP research, fine-tuning BERT-based pretrained models remains a relevant and interesting area of research. It still has a good trade-off in costs and efficiency. Fine-tuning is an affordable way to train a model with a smaller data and computational requirements and also to solve specific problems effectively. Also, it builds understanding from the fundamentals of transformer architecture [11] and fine-tuning training paradigm which is the base of many other language models. In the industry, BERT applications can create significant business value and have been widely integrated to empower popular applications like Google Search [12]. In general, in 2023 the language model applications started to gain mass adaptation and naturally it keeps fueling the research field and keeping it a current topic.

The rest of the thesis is organized as follows. Section 2 gives background to the essential research papers and inventions in NLP in years 2010s. This includes Word2Vec, Seq2Seq models, the attention mechanism, the exploration of the transformer architecture, and its impact on subsequent papers and frameworks, including BERT and SPECTER. In Section 3, the dataset, data collection, and preprocessing and document graph are covered. In Section 4 experiments and evaluation framework are introduced. In Section 5, the results are presented and analyzed. Also further research ideas and potential application ideas are covered. Finally, in Section 6, I will conclude the thesis and summarize the findings. As this thesis replicates SPECTER in many ways, the comparison between the two is made along the thesis.

2. Background

In the background section the key milestones of NLP research in 2010s are covered. The purpose is to highlight essential papers and inventions, which paved the way for transfer learning and fine-tuning paradigms and models like BERT [1] and SPECTER [5]. These are also the focal point of the thesis. Since almost all concepts presented in this thesis are based on neural networks which have been successful in 2010s, a brief introduction to neural networks is provided first.

2.1 A Brief Introduction to Neural Networks

Neural networks are built from layers of connected processing units, where each unit receives input from previous layers. Each unit processes the input and passes the result to subsequent layers [13]. One of the most common type of neural networks are deep feedforward networks [13]. The goal of the network is to find a function $f(x)$ that maps input x to target class y. The verb *find* is used, since the function is approximated from some data. Mathematically, the processing in the unit is simple mathematical operations and can be represented as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

where y is the output, x_i are the inputs, w_i are the weights, b is the bias, and f is the activation function. Each layer consist of weights w_i which are adjusted to minimize the error in predictions. The predictions are the final output of the network, for example a probability distribution over the classes in a classification task. Adjusting of weights is guided by a loss function which measures the error of predictions during training. Two common loss functions, Mean Squared Error (MSE) and Cross-Entropy Loss [14], are presented to provide a basic understanding of loss functions to the readers. MSE is typically used in regression tasks where the output of the network is a continuous value. It can be expressed as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.2}$$

where L is the loss, y_i is the actual value, $\hat{y}_i$ is the prediction of the network and N is the number of elements for which the loss is computed. The actual value is based on the collected dataset, while the prediction is provided by the network. The difference of actual value and prediction is squared which ensures that the loss is always a positive value. The sum of losses is averaged resulting the final loss. Cross-Entropy Loss is common loss for classification tasks where the goal is to predict some discrete value such as 0 or 1 (or True or False) for some tasks. Two-class classification is called binary classification and can be expressed as:

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{2.3}$$

where N is the number of samples, y_i is the actual value (0 or 1, whether the sample belongs to the positive class 1), and $\hat{y}_i$ is the predicted probability of the target class. Logarithm is important here since the logarithm of a probability has a useful property, log of 1 equals 0 which means that when the correct class y_i is 1 and the prediction $\hat{y}_i$ is close to probability of 1, the loss is low indicating a correct prediction and vice versa. A practical use-case of binary classification is a spam detector for email service.

When the loss is known, the weights needs to be adjusted so that the prediction error is minimized in each training step. This is typically done using a method called backpropagation [15], which calculates the gradient of the loss function with respect to weights. The gradients are then used to update the weights in the direction that reduces the loss through optimization algorithms like Stochastic Gradient Descent (SGD). Formally, the gradient update can be presented as:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i}$$

where w_i is weights, n is learning rate which defines how much the model should adjust the weights and L is loss which is the error of the network. This iterative process continues over multiple passes through the training data, known as epochs, until the model's performance stabilizes, indicating convergence [13],[15] and [14]. Ideally, the trained model can perform correct predictions for unseen data. Neural networks have several advanced techniques to control the learning of the network and related problems such as overfitting but those are beyond the scope of this brief introduction.

This section provided a brief introduction to neural networks and basic understanding how computers can learn functions from the data. The innovations presented in the following sections are different type of neural networks but focused on natural language processing (NLP) where the dataset typically consist of some textual data.

2.2 Non-Contextual Word Embeddings

One of the milestones in modern NLP was the development of pretrained word embeddings. Mikolov et al. **[16]** addressed this by releasing *word2vec* in 2013. The key concept of *word2vec* is the idea that the word is defined by its surroundings. The authors used a simple neural network consisting of an input layer, projection layer and an output layer, for pretraining the embeddings. Embeddings are real-valued vectors that represent words in a continuous vector space, capturing their semantic meanings and relationships. The neural network was trained with two different approaches: continuous skip-gram and continuous bag-of-words (CBOW).

Both solutions use a similar architecture for maximizing the probability of either a source word w_t (where t represents the position of the word in the sequence) based on its surroundings w_{t-1}, w_{t-2}, w_{t+1}, and w_{t+2}, or the surroundings based on a source word w_t. The surroundings are selected from both the left and right context of the source word with a certain context window.

Even though the model's objective is to predict the probability distribution over the words, the actual embeddings $\mathbf{v}_w$ are learned in the projection layer, where $\mathbf{v}_w$ represents the embedding vector for the word w. The authors demonstrated that the pretrained embeddings capture semantics and similarity of words enabling to perform simple algebraic calculations with meanings of words. For example, they showed that the vector for the word "smallest" can be derived as follows: *vector("smallest") = vector("biggest") - vector("big") + vector("small").*

The limitation of *word2vec* was that the word embeddings are *non-contextual*, meaning that each word has only one static representation vector. However, the meaning of a word depends on the context in which it appears. This issue was addressed later in NLP with contextual word embedding models. Examples of models addressing this issue are presented later in this chapter, such as BERT [1] and ELMo [17]. This thesis uses *contextual word embeddings* from the Finnish version of the BERT model [9].

2.3 Contextual Word Embeddings

In 2018, Elmo, [17], ULMFit [2] and BERT [1] were published. ELMo provided a solution to the limitation of non-contextual word embeddings. It can be considered as an improvement from bag-of-words and *word2vec* approaches to more flexible word representations which are not constant but a combination of words' left and right context. Compared to *word2vec*, ELMo took the context into account and provides a more context-sensitive representation for each word.

ELMo presented the concept of bidirectionality. It used an architecture that utilizes two unidirectional LSTMs to process text in a forward and a backward directions. ELMo was trained with language modeling task that aims to predict the next word in a sequence. Intuition for the benefit of bidirectional inputs can be demonstrated with an example "A mouse is near the cat" vs. "A mouse is near the laptop". In the examples. when considering the right context of the word "mouse", the semantic meaning of the word ("mouse") changes from an animal to a computer utility. Following the examples, quite intuitively, the quality of the source word embedding increases as the context of the word is added to the embedding itself.

In addition to introducing a new approach for encoding embeddings, ELMo also influenced training paradigms. ELMo demonstrated the incorporation of deep context-aware embeddings into supervised NLP frameworks in a *feature-based* way. *Feature-based* means that the model's lowest layer can be initialized with pretrained embeddings. This approach improved more cost efficient training by enabling the task-specific models to obtain higher results with less training compared to the models which are initialized with random vector or non-contextual embeddings. From the perspective of fine-tuning and task-specific problems, a key limitation of ELMo was that only the first layer could be initialized with pretrained embeddings, while the rest of the model parameters had to be fully trained. This is because ELMo used task-specific architectures where the pretrained embeddings served as fixed feature inputs. Thus, pretrained layers and weights learned in one task could not be fully transferred to another.

ULMFit [2] was introduced in 2018, showing a method to initialize a task-specific model by leveraging a fully pretrained model, rather than concentrating only on the first layer. It presented a three-step training paradigm that can be used for any NLP task: 1) one-time pretraining of a language model, 2) task-specific fine-tuning, and 3) fine-tuning a supervised classifier. Each step required minor changes to the architecture. The authors showed that the pretraining decreased the validation error rate in multiple benchmarks compared to a fully trained model with task specific data. With only 100 labeled examples, ULMFit achieved the same performance than a fully trained model with 10x more labeled data. From the perspective of this thesis the ULMFit's significant outcome was a new training paradigm of utilizing pretrained model in task specific problems. This approach was familiar for Computer vision community already in early 2010s [4] but was now available in NLP research too.

2.4 Transformers and Self-Attention

Transformers, introduced by Vaswani et al. in their paper "Attention is All you Need" (2017) [11] focused on solving machine translation (MT) task with a new architecture.

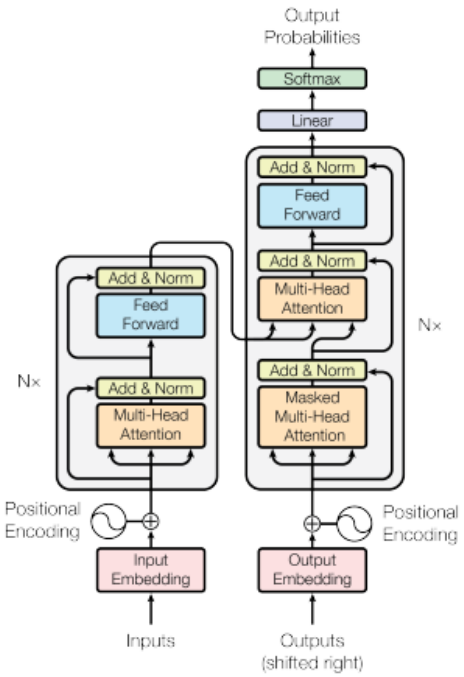

Figure 2.1: The Transformer architecture

The model architecture has been widely adapted by the research community and has been foundational in the development of the most advanced models. In this section the architecture and its key components are presented because all the models used in this thesis and SPECTER [5] are Transformer based.

The Transformer architecture is shown in Figure 2.1. The overall architecture consists of an encoder-decoder structure, where the encoder processes the input sequence, and the decoder generates the output sequence from the output of the encoder. The encoder is composed of stacked layers of multi-head self-attention and feed-forward neural networks. Since SPECTER [5] is based on the encoder of the model, this section is limited to focus on encoder only.

The encoder has six, $L = 6$, similar stacked layers linked to each other. Each layer consists of units of multi-head self-attention and fully connected feed-forward network. The self-attention mechanism enables the model to jointly attend to all the tokens in the sequence when processing each token. A token can be a word or sub-word depending on the technique of converting text to numeric values. In practice, self-attention determines which parts of the sequence are relevant for each token. The authors referred to their self-attention mechanism as **Scaled Dot-Product Attention**. It takes three inputs: query Q, keys K, and values V. It can be expressed mathematically as

follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

where $Q = XW_Q$, $K = XW_K$, $V = XW_V$, and W_Q, W_K, W_V are learned weight matrices. Here, X represents the input tokens with dimensions (T, d_{model}), where T is the sequence length and d_{model} is the embedding dimension, 512 in the original Transformer paper. The weight matrices W_Q, W_K, and W_V have dimensions (d_{model}, d_k). d_k is defined by a d_{model}/h, where h is the number of parallel attention heads. In the paper, $h = 8$, was used, thus $d_k = 512/8 = 64$.

The dimensions of the query, key, and value matrices are (T, d_k). The dot product QK^T has dimensions $(T, d_k) \times (d_k, T)$, resulting in a (T, T) matrix. After that the result is scaled and softmax is applied. The result of softmax function represents the weights for values. The weights are multiplied by V, which has dimensions (T, d_k). Thus, the input and output dimensions of a single attention layer are (T, d_k). The Transformer was originally designed to support a maximum of $T = 512$ tokens. The weight matrices are updated based on the loss function using the standard backpropagation algorithm.

Multi-head attention extends self-attention mechanism by applying several self-attention layers in parallel. The multi-head attention is presented as follows:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O \tag{2.5}$$

The dimensions of a single *head* is (T, d_k) and $(T, d_k \times h) = (T, d_{model})$ for the output of the concatenated heads. Dot product with W^O is a linear transformation. Thus, the output of multi-head attention is (T, d_{model}). The output of the attention layer is fed into the feed-forward layer. The fully connected neural network is described as follows:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.6}$$

where W is a weight matrix, x is the input embedding of dimension d_{model} and b is bias. **ReLU** (Rectifier Linear Unit) is used as an activation function. The second linear transformation produces d_{model}-dimensional output embeddings for each token t_i.

The output of both self-attention and feed-forward functions are normalized by using layer normalization [18]. The layer normalization takes the input of and the output of the previous function. This can be presented as $LayerNorm(x + Sublayer(x))$, where x represents the input sequence and $SubLayer$ represents the equation 2.6 or 2.5.

The original Transformer implementation also included the positional encoding to the input sequences. The mathematical details of the function are present in the

paper. The positional embedding which represents the order of the tokens were added to the input embedding directly. Thus, the model learns the location and the distance between tokens directly from the embedding and the framework does not need to store previous tokens in memory in order to "remember" the context as in recurrent neural networks. This approached improved the handling of long-range dependencies in input sequences.

Even thought "Attention is all you Need" [11] focused on machine translation, the legacy of the paper is the architecture that revolutionized NLP. Its lack of sequential dependencies solved many of the bottlenecks related to scaling language models. Also, the transformer architecture is utilized in models such as BERT [1] which presented a fine-tuning paradigm which instead enabled the development of models like SPECTER [5] and this thesis.

2.5 BERT

Finally, BERT was released in late 2018. BERT proposed two major approaches to language modeling 1) bidirectional encoding of input tokens and 2) masked language modeling (MLM) objective. Also the different foundational model architecture had a crucial impact in more scalable processing of text. While ULMFit and ELMo were based on LSTMs, BERT utilized the transformer architecture introduced by Vaswani et. al. [11]. However, BERT uses only the encoder part of the architecture, while the original Transformer was designed to solve machine translation with encoder-decoder architecture. Outside of the actual problems, the transformer architecture solved multiple challenges related to scaling a language model compared to the ones based on LSTMs. The higher scalability was achieved primarily by removing sequential dependencies. In practice, recurrent units were replaced with self-attention layers [1][11] and the order of tokens in a sequence was incorporated to the input vector itself [11] which enabled the model to process sequences in parallel. At the same time, the computational resources like GPUs advanced rapibly. GPUs are powerful in parallel processing and suited well for training models with transformer architecture.

Bidirectional encoding addressed the problem related to language models that utilized unidirectional encoding such as ELMo and Open AI's GPT [19]. The authors argued that tasks such as question answering would suffer from unidirectionality. This was addressed with BERT's approach to bidirectional input encoding.

Since BERT is based on the Transformer architecture [11] its input and output dimensions are based on the contraints of Transformer. Thus the maximum context length T is 512. BERT implemented the self-attention similar to Transformer paper, except with 12 parallel layers. Since $h = 12$ the d_{model} is $h \times d_k = d_{model} = 12 \times 64 = 768$

as defined in section 2.4. Later in the thesis, I will refer to the fixed-length vectors used in BERT based models. Also, while original used $L = 6$ stacked encoder layers, BERT uses $L = 12$.

The idea of masked language modeling objective is inspired of the Cloze task [20]. The model aims to predict randomly masked input tokens in the sentence from its left and right contexts in an unsupervised way. For example, consider the following:

"The quick brown fox jumps over the lazy dog."

In the masked language modeling task, some of the words in this sentence are randomly masked out, and the model is trained to predict these masked words based on the surrounding context. If words "quick" and "lazy" are masked out, the sentence would look like this:

"The `[MASK]` brown fox jumps over the `[MASK]` dog."

The model's task is to correctly predict that the masked words are "quick" and "lazy" using the context provided by the rest of the sentence. This way, BERT learns to understand the relationships between words in a sentence, capturing the meaning and structure of the language. This unsupervised learning approach allows BERT to be pretrained with a large corpora from different domains and then be fine-tuned to specific tasks.

BERT became the first representation model that was based on fine-tuning and which outperformed architectures that were separately designed for task-specific problems [1]. These kind of pretrained and fine-tunable models were later wrapped in a term *foundation model* [21]. BERT used the same architecture for all the tasks and it was fine-tunable by adding one extra output layer. In fine-tuning, all parameters were initialized from the pretrained *foundation model*. Compared to ULMFit, the pretraining and fine-tuning was more simple since BERT required less changes to the architecture in different phases. From a technical perspective the ease of use of BERT was a key reason leading to a higher adaptation of the model in AI community. The BERT's authors also suggested that tasks with low resources benefit from pretraining [1]. Since BERT was open-sourced the pretrained models could be utilized effectively though fine-tuning across different organizations even when they have just small volumes of domain specific data.

A year after the release of BERT, in 2019, Sentence BERT was published [22] . It was trained to learn embeddings from entire sentences. Sentence BERT uses Siamese network and triplet loss function where semantically similar sentences are learnt to have close vector representations [22]. While BERT focuses on token level, Sentence BERT

is trained for sentence level understanding. Training setup for learning sentence-level representations is closely related to how SPECTER is trained with document-level objective to generate document vectors which is outlined in the next section.

SciBERT, introduced by Beltagy et.al. [6] in 2019, is a variant of BERT specifically designed for scientific text. Based on the authors, deep neural models were mostly trained with annotated data at the time. Since scientific domain was lacking labeled data and the annotation is expensive, the authors addressed this by training the model with the same objectives as BERT was trained but with scientific texts. The goal of the authors was to train a model which is able to produce high-quality representations from scientific documents. SciBERT is essential for SPECTER since it utilizes SciBERT as an initial pretrained model.

2.6 SPECTER

The main focus of this section is to provide an overview of the SPECTER model and to highlight how the *token- and sentence-level* objectives used in pretraining are replaced with document-level objective in fine-tuning a new language model.

SPECTER [5] was published in 2020. Technically, SPECTER is a general-purpose text encoder that generates fixed-length *document-level* vector representations from scientific articles. SPECTER is *fine-tuned* with a *document-level* objective from pretrained SciBERT [6]. The transformer architecture of SPECTER is inherited directly from SciBERT and BERT [5]. Thus, the fixed-length vectors are 768-dimensional as decribed in 2.5. The fine-tuned model was used with *feature-based* approach in downstream tasks. This means that document embeddings were produced by transformer LM itself without further fine-tuning and the embeddings were used as features for the downstream model. The approach led SPECTER to outperform baselines in several benchmarks such as topic classification, citation prediction and user activity prediction.

The motivation for SPECTER raises from the authors' claim that BERT has a limited ability to represent documents as a whole. The key argument was that BERT was trained with masked language modeling and next sentence prediction objectives [1] which aims to build understanding from token- and sentence-level relationships from in-document data. This was believed to be sub-optimal in *document-level* tasks which are in the scope of SPECTER and this thesis. However, understanding documents as a whole is important in tasks like classifying or searching documents and SPECTER tried to solve this limitation with a document-level fine-tuning objective. Figure 2.2 below illustrates the training process that eventually optimizes and updates the model parameters from token-level to document-level tasks. On a high level, the training framework contains four levels (from top to bottom): 1) Feeding triplets to the model,

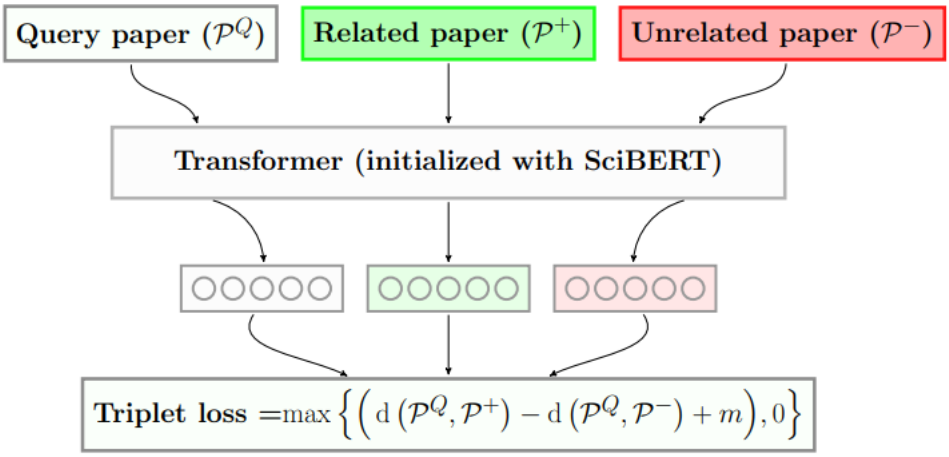

Figure 2.2: Training architecture of SPECTER [5]. The transformer LM is initialized with SciBERT but can be replaced with any Transformer LM such as Finnish BERT. The loss is computed with triplet margin loss using query, positive and negative papers in triplets.

2) encoding each document with a transformer encoder, 3) building vector representations of each paper, and 4) computing the loss through an activation function. Each level is described in the following paragraphs.

2.6.1 Model input

One of the crucial change to the SciBERT is how inputs are fed to the model in fine-tuning process. Instead of feeding input and outputs, SPECTER takes three instances as an input: query, related paper, and unrelated paper. A triplet of three papers are sampled from the document graph.

For each input instance, the title and the abstract of the paper are concatenated. Each input sequence starts with the CLS and the SEP token separates the title and the abstract. A significant change compared to the original BERT is, the CLS token is placed at the beginning of each document instead of at the beginning of a sentence, as presented in the original BERT paper [1]. The maximum sequence length used in the thesis is $T = 128$ which is determined by the median of token lengths. Since the text pieces selected from Wikipedia articles are short, the benefit of longer context is less significant. However, this approach involves a trade-off between performance and cost-efficiency. By feeding the triplet of queries, positive and negative papers, the loss function can learn the semantic similarity between documents. The relatedness is based on the citations within the documents. The citation graph is described in detail

in Section 3.

2.6.2 Initialized parameters

The transformer's parameters in SPECTER are initialized from the pretrained SciBERT. All parameters are updated during fine-tuning. The version of the model architecture is $BERT_{BASE}$. The hidden layer size H is 768, a number of stacked encoders L=12, a number of attention heads A = 12 (or h) and total parameters = 110M. The architecture is agnostic to the Transformer LM it is initialized with. The architecture is thus flexible to be used in other domains and languages by replacing data and the input of the model as long as the structure of the data is the same. The same $BERT_{BASE}$ architecture is also used in Finnish BERT [9].

2.6.3 Embeddings and Inference

Document representation is a representation of the CLS token in the last hidden layer of the transformer LM. The length of the embedding is H = 768 (also d_{model}). Each of the three papers is embedded through the model as described as follows:

$$v = \text{Transformer}(input)_{CLS} \tag{2.7}$$

The output of the transformer for CLS token is a dense vector

$$\mathbf{v}_{\text{CLS}} \in \mathbb{R}^D \quad \text{, where} \quad D = 768 \tag{2.8}$$

which is the final representation of the document at the inference time. In the training phase, each paper is embedded separately from the last hidden layer's CLS token which is also called the "pooled output" [5]. Eventually, all three vectors are used for computing a loss and optimizing the parameters through backpropagation.

At inference time, the document graph is not needed for encoding new unseen documents because SPECTER's model utilizes only the title and abstract of the input document to generate its embedding. During training, SPECTER leverages citation information to fine-tune the model and learn document representations that capture the relatedness between papers. However, once trained, the model can independently generate embeddings based on only the textual content of new documents without requiring citation data or the structure of the document graph. This capability allows SPECTER to produce embeddings for new documents, including those that have not yet been cited, making it versatile and efficient for real-time applications targeting recent scientific papers.

The model can be used as a general-purpose embedder in downstream applications. The transformer is trained to compress semantics of the document to a *fixed-length* vector. Ideally, the vector represents the content of the document but also its semantic relation to the other documents making the documents comparable to each other. Thus, the model is optimal in document level tasks such as topic classification or recommending citations for a new unseen paper.

2.6.4 Loss and Training Objective

As a loss function, SPECTER uses **a triplet margin loss** [5]. For each training instance, the model sees a triplet of a query paper P^Q, a positively related paper P^+ and a negatively related paper P^-.vThe three papers are sampled from the citation graph. The triplet margin loss is defined as follows:

$$L = \max\Big\{\Big(d(P^Q, P^+) - d(P^Q, P^-) + m\Big), 0\Big\}, \tag{2.9}$$

where L is the computed loss, d is the distance between two papers, and m is the loss margin. The authors set the margin at $m = 1$, and this value is also used in this thesis. The distance d is an L2 norm defined as follows:

$$d(P^A, P^B) = \|v_A - v_B\|_2 \tag{2.10}$$

where v_A and v_B are *fixed-length* vectors that represent documents A and B. v_A and v_B are extracted from encoder's classification token in the last hidden layer. The authors also experimented other distance functions such as normalized cosine but L2 outperformed those.

The idea of the triplet loss function is that the model tries to learn close representations from P^Q and P^+ and vice versa from P^Q and P^-. As an activation function **rectified linear activation function** or **ReLU** for short is used. If the loss is negative, it will output 0 otherwise the positive loss itself. In order to build intuition how the triplet loss works I will provide examples below.

Without margin m, when $d(P^Q, P^+) < d(P^Q, P^-)$, the positive paper is closer to the query paper than the negative paper. This is the desired outcome. In this scenario, the loss is set to zero and the model parameters do not need to be adjusted. Triplet loss enforces the margin to separate various pairs, pushing related data points close together while ensuring distant projection for unrelated ones. Thus, as long as $d(P^Q, P^+) - (P^Q, P^-) > m$, the model returns a positive loss and the parameters are adjusted. The idea of the margin is to define a minimal distance that a similar and dissimilar paper should have in vector space in relation to the query paper.

2.7 Related work

In addition to SPECTER [5], the document-level training objective with fine-tuned Transformer has been researched and implemented in other papers such as [23] and [24]. Yasunage et al. [23] proposed a model called *LinkBERT* which incorporates document links such as hyperlinks from Wikipedia data as done also in this thesis. They used a masked language modeling objective and a new task called document relation prediction. Document relation prediction is built on the idea of next sentence prediction task where the goal is to learn whether the two sequences are consecutive.

The other paper by Caciularu, Cohan et. al [24] is partially from the authors of SPECTER. In the paper, the authors proposed a new model **Cross-Document Language Model** (CDLM) for learning overlapping information from related documents such as news articles covering the same topic but with different words and expression. The pretraining was based on a set of related documents and a global attention mechanism [24].

In the background section, the context was built for this thesis. Multiple innovative approaches to text processing and language modeling were covered, and this thesis is directly based on those ideas such as Transformer and contextual word embeddings. In the next section, the data used in the thesis will be described in detail.

3. Data

In this chapter, I describe the data used in the thesis, focusing on explaining why Finnish Wikipedia is suitable for this thesis and also highlighting the key components of the dataset needed for training LM with the SPECTER framework. Also, building a document graph is covered and the structure of the graph is illustrated with visual and textual examples. Within the chapter comparison between data used in this thesis and SPECTER is made to understand the relationship of the two. After the data used for fine-tuning is described, the evaluation data and its labeling from Wikipedia's classification system is explained. Then, the problems related to unbalanced and skewed distribution in the evaluation data are addressed and also the methodology for preparing the training data is outlined.

3.1 Data Selection and Comparison

Data selection had two main objectives directly related to the research questions: to find a data source that has a network structure similar to that of scientific articles and has a domain outside of science. The main data source is Finnish Wikipedia which fulfills data selection objectives well. It has a network structure where articles are linked to each other via cross-references. This is essential for building the document graph. Also, the language changes from English to Finnish and domain from science to general. This is why the dataset is suitable for testing the flexibility and general-purpose nature of the SPECTER framework.

SPECTER requires a specific format of input data for LM training. Briefly, the idea is to feed summarizing text fields from articles which represents the whole text. SPECTER used combination of the title and the abstract for this purpose. In this thesis, the title and part of the introduction is used to represent the full document. Wikipedia data has no abstract similar to scientific papers and instead, the first paragraph of the introduction is considered to represent "the abstract". For building citation graph, SPECTER used citations but this thesis uses cross-references to other Wikipedia articles which appear within the extracted introduction. Each training instance consist of triplet of query article, positively related article and negatively related

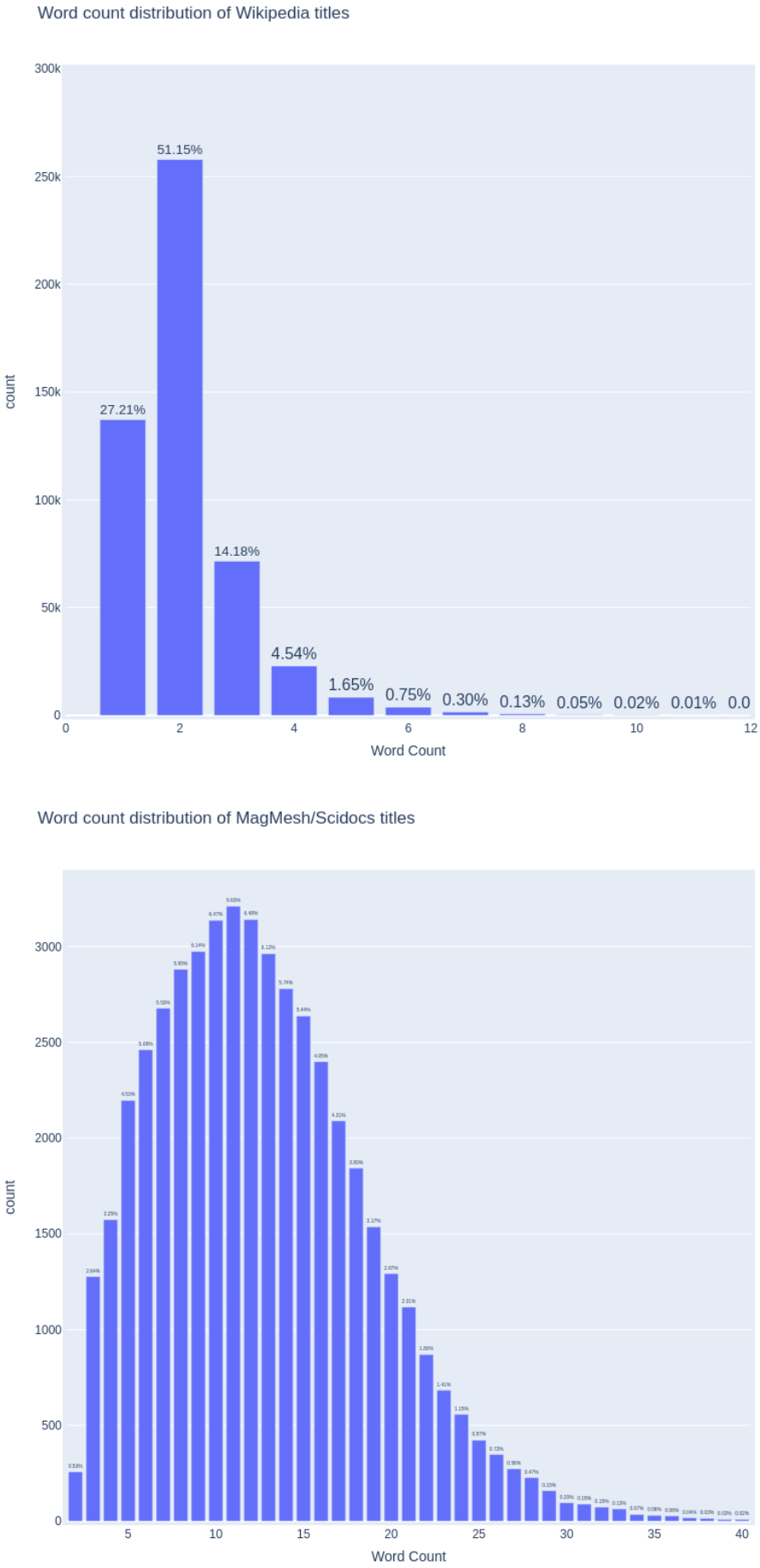

Figure 3.1: Distribution of title word counts in scientific articles and Wikipedia. The above histogram is from Wikipedia's training dataset and the below is from Scidocs' MagMesh evaluation task dataset. The mean of the distributions are 2 for Wikipedia and 12 for MagMesh.

article. These are sampled from the document graph. The sampling is described in section 3.2

SPECTER did an ablation study of which text fields represent the papers the best [5]. The authors studied how results in evaluation tasks change when other text fields such as authors or publish venues are added or dropped from model inputs. Adding author or venue to the text fields had a negative impact on results in average. The authors found out that dropping the abstract decreased the results significantly. However, the results were still reasonable. This can be due to the richness of the title in scientific domain and also the use of citation graph. While in scientific domain the title is often a sentence or a statement that describes the topic such as "Attention is all you need" [11] where as in Wikipedia the title is typically just a noun of less rich in content such as "Finland" or person's name. In Figure 3.1 the two graphs show the length distribution of the title for both scientific papers and Wikipedia articles to demonstrate the intuition about the title lenghts.

Both data sources share a similar connected structure but they also differ. The main differences are domain, language, and writing conventions. Scientific articles focus on specific research areas. The language is more concise and structured guidelines are derived from academic norms. Instead, Wikipedia spans a broader domain, open to edits from the public, resulting in varied completeness and lacks similar guidelines as academy.

Scientific articles are authored by experts and remain static after publication. In contrast, Wikipedia is a dynamic content platform where both authors and content are subject to continuous change. Scientific papers have multiple publishing venues, while Wikipedia is a single platform for all languages and texts. While Wikipedia has less conventions, it is worth mentioning that Wikipedia does have writing guidelines and policies. However, ironically, one of the five Pillars of Wikipedia is that "Wikipedia has no firm rules" and "If a rule prevents you from improving or maintaining Wikipedia, ignore it" [25] [26]. This leads to a greater variance in Wikipedia's content compared to the more uniform scientific domain, presenting a technical challenge in data preprocessing for this thesis.

Despite the differences between the data sources, the key data components are found from both. These are networked texts (articles, papers) that enable building a citation graph and in-document text fields that can be used to provide an overview of a full document.

3.2 Document Graph and Intuition on Data and Cross-References

The document graph itself (see Fig 3.3 for example) is a generic data structure and agnostic to the domain. It can be fitted to many different datasets that have links between the elements such as articles or users (e.g., social networks). Its purpose in this thesis is to store the relationship of documents in the dataset. The Wikipedia articles represent the nodes in the graph. The edges represent cross-references to other Wikipedia articles from the source article. This section describes how document graph is built from Wikipedia data, which is used for sampling the training instances (triplet of query, positive, negative). From now on, the links between articles are called as *cross-references* while in SPECTER the term *citations* was used. While SPECTER used the term *citation graph*, *document graph* is used in this thesis.

In SPECTER the citation graph was built from the full list of references. In this thesis the cross-references are selected from the first paragraph of the introduction. This choice limits the connections between source articles and cross-referenced articles. Intuitively, articles typically start with high-level information and go in depth later in the text. Thus, the cross-references in the early parts of the document provide background information for high-level topics related to the source article and the model is limited to learn only these relationships in training.

3.2.1 Intuition

The first sentence of the article *Alankomaat* (Netherlands) is given below. The hyperlinked words are called **cross-references** and they are pointing to another Wikipedia article. They represent "citations" in the thesis. To respect a wider audience, the English translation is provided in the lower sentence, in which the hyperlinks are styled with **bolding**.

> *Alankomaat eli Hollanti on maa ja itsenäinen valtio, joka sijaitsee pääosin läntisessä Euroopassa, Pohjanmeren rannalla.*
>
> In English: *The Netherlands is a **country** and an independent **state** located mainly in western **Europe**, on the shores of **the North Sea**.*

Cross-references from the article 'Alankomaat' (Netherlands)

- Maa (a country)
- Valtio (a state)

- Eurooppa (Europe)
- Pohjanmeri (the North Sea)

The above cross-references are related to the article *Alankomaat* ("Netherlands") on a high level obviously. Assumption in the thesis is that a cross-reference from article A to B suggests that B is related to A. The same assumption was also made in another paper [23] and the authors believed that hyperlinks between the articles are likely to have a high precision of relevance. Cross-referenced articles add background information to the document which is not present in the source itself. Quite clearly, all the references above are justified and provide high-level background information to the article *Alankomaat* ("Netherlands").

In the training process, information from the cross-referenced articles and their topics are compressed to the query article's document embedding through the triplet loss (Equation 2.9). For example, the linked articles listed above reveal information about geographic location, weather, language and political system of Netherlands. By skimming articles in a category like "countries in Europe", it can be seen that the articles of European countries have mutual cross-references of topics related to location, size or neighboring countries (see articles of Alankomaat, Italia, Ranska, Ukraina for example). By linking other articles to the source article through triplet loss and sampling training instances from the document graph, the relationships between the articles should transfer to the model through backpropagation (parameter update). This can be beneficial especially for articles that are short or incomplete but have cross-references which enrich the content with linked articles. It is notable, that articles without cross-references cannot be used as query articles in training.

The network of cross-references is visually illustrated in Figures 3.3 and 3.2. In Figure 3.2 the starting node is the article "Alankomaat/Netherlands" *. The figures demonstrate the complexity and the structure of the document graph. It shows what articles are related and how the networked structure starts taking shape already after following only a few cross-references from the parent node "Alankomaat/Netherlands". In Figure 3.2, the green lines link the same articles together. For example, "Valtio/State" is cross-referenced by "Alankomaat/Netherlands", "Politiikka/Politics" and "Itsenaisyys/Independency". This illustrates how a single article can be a mutual cross-reference for multiple articles while the articles are not directly cited by each other.

Mutual cross-reference of articles which are not directly linked to each other are demonstrated also in Figure 3.3 . For example, the articles "Alankomaat" (Netherlands) and "Atlantin valtameri" (Atlantic Ocean) do not cite each other but both

*English translation after "slash"

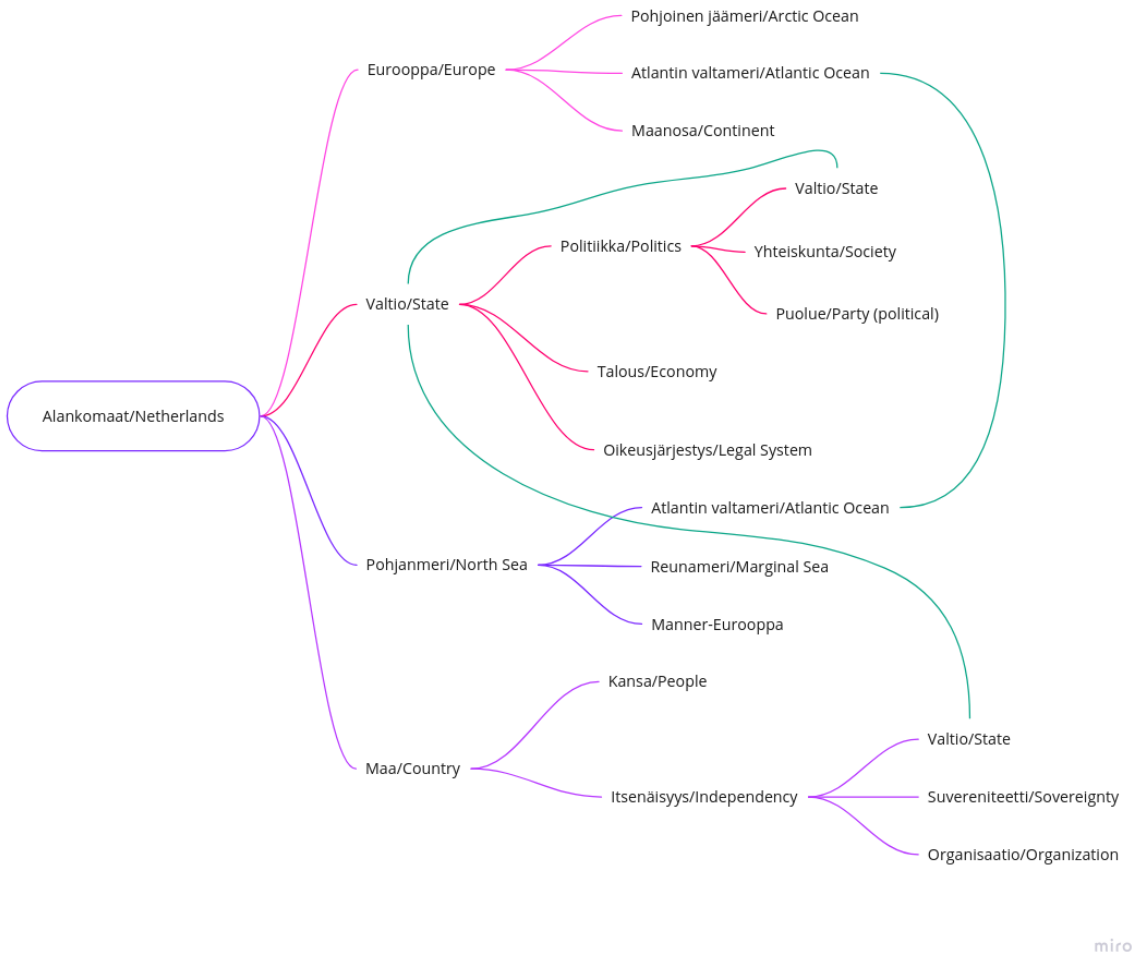

Figure 3.2: The links are from originally from Finnish Wikipedia but translated both in Finnish and English to respect the non-finnish readers. Illustration of the outgoing links from a source article "Alankomaat". By following links for 2-3 edges it can be seen that articles as "Europe" and "North Sea" are linked to each other over commonly citated article "Atlantic sea". Also article "State" is related to both "Politics" and "Netherlands"

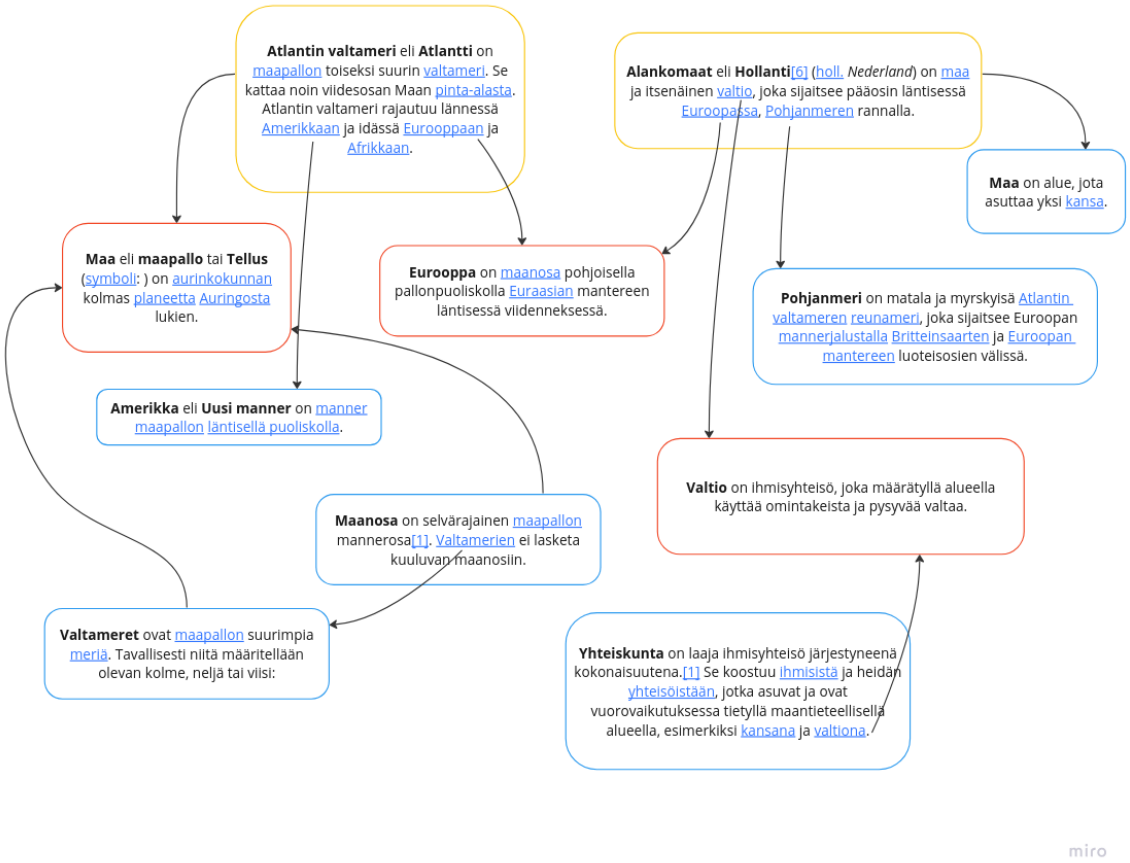

Figure 3.3: Illustration of how articles are linked and related to each other. For example "Alankomaat/Netherlands" and "Atlantin valtameri/Atlantic Ocean" are not cited by each other but both cites the article "Eurooppa/Europe". The text areas with red edges represent articles which are cross-referenced by multiple articles.

cite the article "Eurooppa" (Europe). While building training triples, the article "Eurooppa" (Europe) can become positively related article for query articles "Alankomaat" (Netherlands) and "Atlantin valtameri" (Atlantic Ocean). Through the triplet loss the two vectors of "Alankomaat" (Netherlands) and "Atlantin valtameri" (Atlantic Ocean) are pulled closer together though they are not directly related but have a mutual cross-reference.

3.2.2 Triplet Sampling Process

The training instances are sampled from the document graph. Each article a_i in the dataset has a set of cross-references that are categorized as positive or negative based on their "count" value. Positive cross-references have a count of 5, indicating a strong relationship, while negative cross-references have a count of 1, indicating a weaker or no relationship.

Formal Definition

Let A be the set of all article IDs in the dataset. For each article $a_i \in A$, let P_i and N_i be the sets of positive and negative cross-references, respectively.

$$P_i = \{p \mid \text{count}(a_i, p) = 5\}$$
$$N_i = \{n \mid \text{count}(a_i, n) = 1\}$$

Triplets are sampled such that each article a_i has five triplets (a_i, p, n), where:

- a_i is the query article.
- $p \in P_i$ is a positive cross-reference (an article with a strong relationship to a_i).
- $n \in N_i$ is a hard negative cross-reference (an article with a weaker relationship to a_i).
- n_e is an easy negative cross-reference (any random paper in the dataset that is not in P_i or N_i).

Each query article generates 5 triplets, consisting of 2 hard negatives and 3 easy negatives.

Sampling Process

1. **Initialization**: For each article a_i, initialize an empty list of triplets T_i.
2. **Positive Sampling**: Randomly select a positive cross-reference $p \in P_i$.
3. **Negative Sampling**:
    - **Hard Negatives**: Randomly select 2 hard negative cross-references $n_h \in N_i$.
    - **Easy Negatives**: Randomly select 3 easy negative cross-references n_e from the dataset that are neither in P_i nor in N_i.
4. **Triplet Formation**: Form the triplets (a_i, p, n_h) and (a_i, p, n_e) and add them to T_i.
5. **Repeat**: Repeat steps 2-4 until five triplets have been sampled for each article a_i.

Formally, the triplet sampling for each article a_i can be described as follows:

$$T_i = \{(a_i, p_j, n_{h_j}), (a_i, p_k, n_{e_k}) \mid p_j \in P_i, n_{h_j} \in N_i, n_{e_k} \in \text{dataset} \backslash (P_i \cup N_i), j = 1, 2; k = 1, 2, 3\}$$

Easy Negative Definition

Let E_i be the set of easy negatives for article a_i, defined as:

$$E_i = \{e \mid e \in \text{dataset} \setminus (P_i \cup N_i)\}$$

The selection of 2 hard-negatives and 3 easy negatives is based on empirical tests done in the SPECTER paper. When the article works as a query article it has different combinations of positive and negative articles. As *word2vec* authors [16] trained word embeddings on top of the idea - a word is defined by its surroundings - the similar analogy can be applied here - the document (embedding) is defined by its citations (surroundings).

3.3 Data Collection and Preprocessing

Next, data collection and preprocessing are discussed. Wikipedia data dumps are publicly available and the Finnish Wikipedia data dump* was downloaded from https://dumps.wikimedia.org/. The data was then decompressed, resulting in the raw data dump file in a structured xml format.

After decompression, the data had to be preprocessed. The preprocessing aims to transform the data suitable for training a model. The first step was to extract the articles from the full data dump and then extract relevant parts from the articles needed to train the LM. The full data dump included other data types than articles such as metadata, user data and templates. Each data type was labeled with a dedicated namespace identifier which made article extraction a straightforward process. From extracted articles, the title, introduction, and cross references were separated. Technically the extraction was performed with a Python script that was mainly based on WikiExtractor [27] but in this thesis the repository was customized to focus on the title, the first paragraph of the introduction, and the cross references within the introduction. The final extractor is available in my GitHub profile to replicate the extractor step†.

The preprocessing included several steps of conditional data cleaning. Wikipedia has its own templates for writing articles [28] which include rules on how to style text (bold, italic, etc.), how to represent a word with a hyperlink or how to present headings, and so on. Below are examples of how a simple sentence with different styling and cross-references is represented and how it should be cleaned. In the below example the goal is to clean RAW to CLEANED.

*https://dumps.wikimedia.org/fiwiki/

†Link to data extraction: https://github.com/jokineno/wikiextractor

1. (RAW) "'Amsterdam"' on [[Alankomaat|Alankomaiden]] [[paakaupunki]].
2. (CLEANED) Amsterdam on Alankomaiden paakaupunki.

The first sentence (1. RAW) is in the raw format and it starts with the '"**Amsterdam**"'. Worth noting is that, the word is surrounded by three single quotes. Three single quotes is Wikipedia's technique to **bold** words in the web user interface. In the preprocessing, the quotes were removed so that words with or without additional styling represent the same word. This is done before tokenization. The overview of tokenization is presented later in this chapter.

Next, there is [[*Alankomaat*|*Alankomaiden*]] where "Alankomaat" is a cross-reference to another Wikipedia article "Alankomaat". "Alankomaiden" (after the vertical line) is the actual word displayed to the reader on the web. The cross-reference can also be displayed as [[paakaupunki]](Capital) if the actual word in the article is in the same format as the cross-referenced article. Wikipedia has multiple similar syntaxes to add metadata such as hyperlinks or styling to the text. In preprocessing, all styling components (quotes, brackets, vertical lines, etc.) are considered as irrelevant data, and cleaned resulting text data without any additional styling components. Eventually, the example data is cleaned as displayed in Example 2.

3.4 Training Data - Fine-Tuning

After cleaning the data, the document graph is built. The document graph is used for sampling training triplets. The training data contains triplets of query article, positively related article, and negatively related articles. The query articles are sampled from all of the Wikipedia articles using random sampling. However, a constraint for an query article is that it must have cross-references. The positively related article is cross-referenced by the query article, as demonstrated in Figure 3.3. By default, the negatively related article is any random article that is not cross-referenced by the query article. Randomly sampled negative papers were denoted as "easy-negatives" in SPECTER [5]. SPECTER authors found that the algorithm learning can be improved by augmenting triplets with *hard negatives*. A *hard negative* article in SPECTER was defined as a citation of a citation. This means that an article is not directly cross-referenced by the query article, but it is cross-referenced by an article which is cross-referenced by the query article. To give an example, in Figure 3.3 the article "Politiikka/Politics" is a hard negative to "Alankomaat/Netherlands" since "Politiikka/Politics" is not cross-referenced by "Alankomaat/Netherlands" but is cross-referenced by "Valtio/State" which instead is cross-referenced by "Alankomaat".

Essentially, this enforces the algorithm to learn semantic difference between papers that are related at some level.

Training instances are created by generating 5 triplets from each query article as described in Section 3.2.2. In total, this resulted 2.1M training triplet instances for training and 230K instances for validation. The data were split into train and validation sets in 90-10 relation.

The final dataset is divided into training and validation sets by random split. In this thesis, each query article is exclusively assigned to one set only to avoid data leakage. However, the articles used as positive and negative examples for the triplet loss are allowed to leak in other sets as well. This approach recognizes that articles serving as positive or negative examples can be relevant to more than one query article. This supports model's ability to differentiate between relevant and non-relevant articles.

In the triplet generation process, each article text (title + a paragraph of introduction) was tokenized. The tokenization is a typical step in natural language processing before feeding data to a training algorithm. In the process, the text is split into smaller units called tokens. The tokens are essentially words or sub-words or even just characters. For each token there is a corresponding numerical ID and thus, the tokenized text can be represented numerically to the neural network.

For optimal results of fine-tuning, the selected tokenizer should be the same as that used for pretraining. This is because the tokenizer has a finite vocabulary that it is trained with. By changing the tokenizer, the words could be split to different tokens which results in producing different embeddings from the same input. Below is an example illustrating this issue. The first tokenizer is the same which was used for pretraining Finnish BERT with Finnish text and the second is the tokenizer which was used for pretraining the original BERT.

- Sentence: "Fine-tuning LLM is important."
- Tokens (Finnish BERT): ['Fin', '##e', '-', 'tun', '##ing', 'LL', '##M', 'is', 'imp', '##ort', '##an', '##t']
- Tokens (Original BERT): ['Fine', '-', 'tuning', 'LL', '##M', 'is', 'important']

Here, it can be seen that for the word "Fine-tuning" the first tokenizer produces five tokens and the second only three tokens. As the transformer maps tokens to embeddings,

Obviously, the tokenizer used in Finnish BERT produces more tokens. This is because it's optimized for Finnish language which is rich in spelling and morphology. Also, Finnish BERT is trained with Finnish language and using the tokenizer for English text is suboptimal.

In this thesis, the TurkuNLP tokenizer was used for preprocessing Wikipedia data [9]. It was also used in the original Finnish BERT paper [9]. Especially in Finnish language, tokenization helps machines interpret variations in spelling and morphology.

Below is an example of the structure of the document graph. It is presented as a JSON object. The graph is a dictionary, where the keys at the root level represent the IDs of the query articles. The values associated with these keys are themselves dictionary objects. In inner dictionaries, each key is an ID of the article, and the associated value ("count": <int>) indicates whether the article is considered positive or negative: a count of 5 represents a positive article, while a count of 1 represents a negative article. This information is subsequently used in the sampling to create training triplets.

```
"<article id>": {
    "<positive-cross-reference-id-1>": {
      "count": 5
    },
    "<posive-cross-reference-id-2>": {
      "count": 5
    },
    "<negative-cross-reference-id-3>": {
      "count": 1
    },
    "<negative-cross-reference-id-4>": {
      "count": 1
    }
    ...
}
```

3.5 Evaluation Data - Topic Classification and Cross-Reference Prediction

This section covers the data used in the evaluation. The evaluation framework itself will be described in the Chapter 4. The WikiSpecter is evaluated in two supervised tasks: topic classification and cross-reference prediction. The both tasks require labeled data. For topic classification, Wikipedia's internal classification systems is used for building a labeled dataset. For cross-reference prediction, the evaluation dataset is sampled directly from the articles and their references.

In topic classification, the articles are labeled with an article category which are

derived from Wikipedia's general classification system. *General classification system* is the sub-category of the highest top-level category in Finnish Wikipedia. The following is a path to the system from the top-level category. English translation is in parentheses.

> *Pääluokat (Main categories) > Wikipedian Luokat (Categories of Wikipedia) > Yleinen luokittelujärjestelmä (General classification system).*

The *general classification system* has 10 *top-level* topics [29]. The topics cover a subset of articles and each article belongs to the topics or their sub-topics. Below is a list of *top-level* topics. English translations are after "/" character.

- 0. Yleiset tieteet / General Sciences
- 1. Filosofia, Psykologia / Philosophy, Psychology
- 2. Uskonto / Religion
- 3. Yhteiskuntatieteet / Social sciences
- 4. Maantiede / Geography
- 5. Luonnontieteet / Natural sciences
- 6. Soveltavat tieteet, tekniikka / Applied sciences, technology
- 7. Taiteet, liikunta, huvit / Arts, sports, amusement
- 8. Kirjallisuus, kielitiede / Literacy, language science
- 9. Historia - History

Each topic has at least one sub-topic. Here, the *sub-topics* are denoted as *second-level topics* in the following sections. For example, class *0. Yleiset tieteet* has four *second-level* topics as follows:

- 00 Kirja-ala, kirjoitus / Book industry, writing
- 02 Kirjastotoimi, kirjastotiede, informatiikka / Library operations, library science, informatics
- 06 Yleinen kulttuuripolitiikka / General cultural policy
- 07 Viestintä, joukkoviestintä / Communication, mass communication

Table 3.1: Distribution of articles over top-level topics.

title	article count	%
7 Taiteet, liikunta, huvit	1041	29.28
9 Historia	718	20.20
5 Luonnontieteet	596	16.77
4 Maantiede	467	13.14
6 Soveltavat tieteet, tekniikka	268	7.54
3 Yhteiskuntatieteet	159	4.47
8 Kirjallisuus, kielitiede	128	3.6
2 Uskonto	94	2.64
0 Yleiset tieteet	55	1.55
1 Filosofia, psykologia	29	0.82

In total, there are 76 *second-level* topics. The article distribution over both *top-level* and *second-level* topics are imbalanced, which is a typical issue in real-world datasets. This is taken into account when choosing evaluation metrics and sampling data for training and evaluation.

In both tasks the minimum limit of documents per topic is set to 10. For top-level classification task, this does not have any impact but for the second-level topics, few topics are filtered away due to low number of documents. The maximum number of documents per label is set to 300 for top-level topics, which is approximately 10x the lowest occurring topic, and 200 for second-level topics, which is 20 times the lowest occurring topic. After this filtering the dataset is shuffled and split to train and test sets in 70-30 relation.

When a dataset is imbalanced, other metrics than accuracy are beneficial for evaluating the model in classification tasks. *F1-score* [30] is a standard metric for evaluating models with unbalanced data distributions, as it balances with major and minor labels compared to accuracy alone. The *F1-score* is a harmonic average of precision and recall [31]. The article distribution over the topics is presented in Table 3.1. The distribution of the *second-level* topics are visualized in Figure 3.4

In topic classification the data are split in train and test sets with 70-30 relation. Ideally, the train and test splits should have a balanced distribution over the articles. In order to avoid unseen label classes in the test set, the split sets are sampled from each topic separately.

In the cross-reference prediction task, the dataset is based on 1000 articles. For each article in the evaluation dataset, 30 articles are chosen as candidate articles. 5 out of 30 articles are cross-referenced by the source article and 25 are randomly selected.

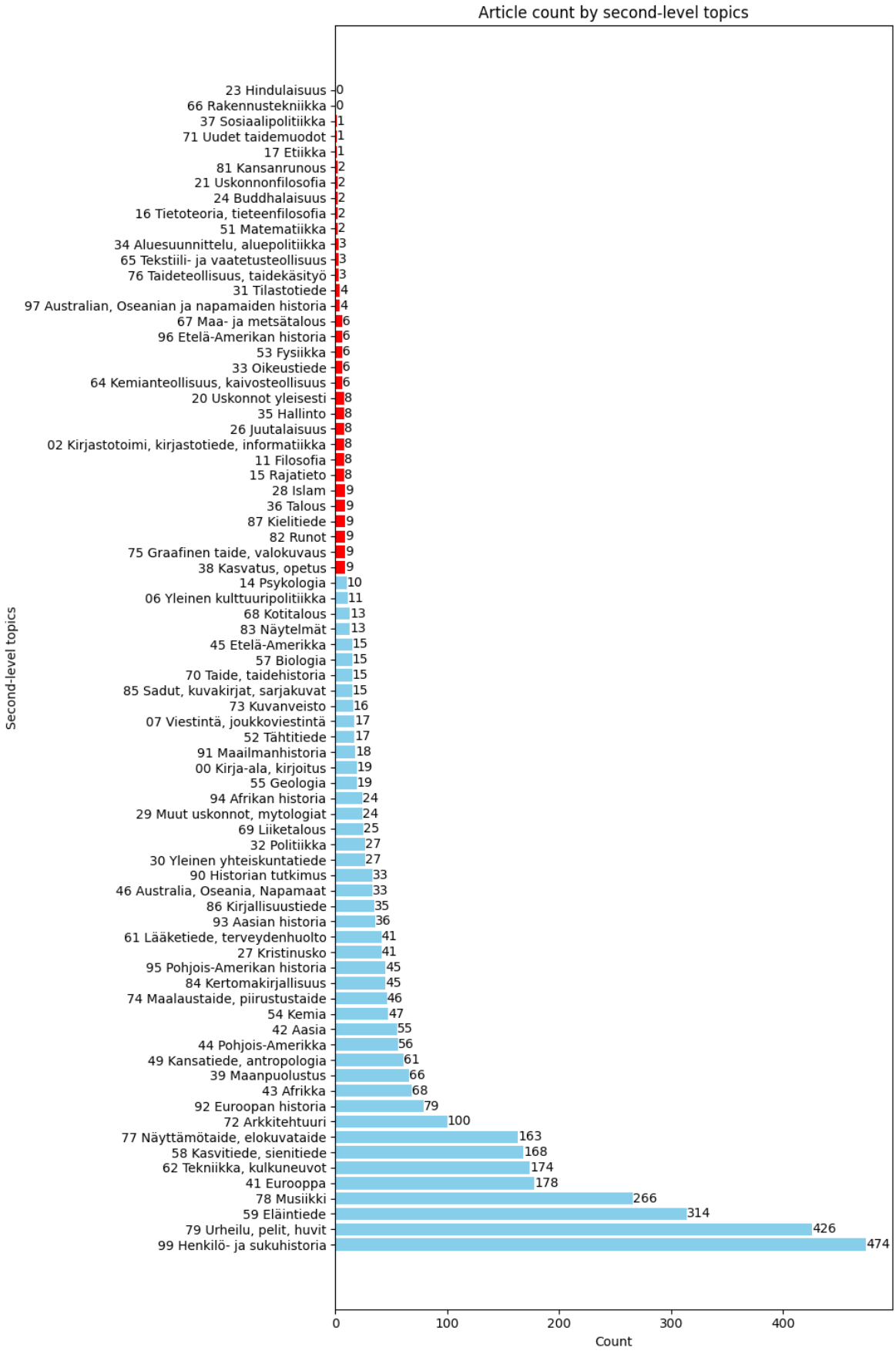

Figure 3.4: Distribution of the second-level topics in Finnish Wikipedia's general classification system. The ones marked with red were filtered due to low number of occurences.

This results in 30K articles in the evaluation dataset. The articles are sampled from the holdout dataset which is not included in fine-tuning the WikiSpecter. The task is described in detail in Section 4.3. Briefly, the idea is that the model should rank the related articles higher than the unrelated ones. The ranking is based on the distance between embedding vectors.

4. Experiments and Evaluation Framework

This section first outlines the fine-tuning process, the computational resources utilized for training, and the rationale behind selecting the baseline models. Secondly, the evaluation framework is introduced.

4.1 Training and Implementation

In the training and implementation part, the focus is on fine-tuning a pretrained model with new data and objective. The data collection and preprocessing is done prior to model training but is left outside of this chapter as it was already handled in the Chapter 3.

In fine-tuning, the initial model parameters are derived from the pretrained Finnish BERT [9] and all model parameters (n = 120M) are updated in training using triplet loss. The objective is to learn document-level relationships between the Finnish Wikipedia articles and utilize them in downstream task.

The essential training parameters are as follows. The margin in the triplet loss function is set to m=1 and loss distance is calculated with the L2-norm as in SPECTER. The SPECTER authors did experimentations for finding the optimal margin and they eventually used m=1. As an optimizer, Adam [32] is used with the hyperparameters proposed in the original BERT paper [1]. The model is trained with 2 epochs with a batch size of 16. Also bigger batch sizes were tested but they caused memory errors in validation steps. The epoch is set to 2 as in SPECTER but also to respect the computational resources. The maximum sequence length is set to 128 which is close to the median of tokenized input lenghts. The token lenght distribution is visualized in Figure 4.1 Each epoch took around 6 hours of training time resulting in 12 hours total training time. Also, the training loss decreased over the training steps indicating a successful learning progress. The parallelized computation on 4 GPUs significantly increased the training speed.

In general, the training parameters are recommendations from the earlier works

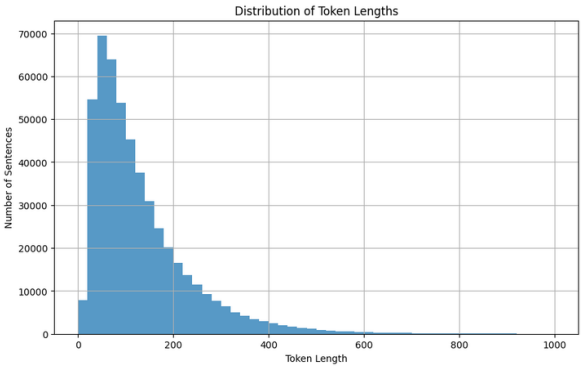

Figure 4.1: Distribution of token lengths

such as the original BERT paper and SPECTER and this thesis leaves room for further parameter optimization for this particular fine-tuning experimentation.

The model training was performed in the Puhti cluster of CSC (IT Center for Science)*. In this thesis, a single node in the cluster was reserved for training. The nodes in Puhti cluster have 4 Nvidia's V100 GPUs, each with 32 GB of memory. My training constraints were limited to 72 hours of training time using a maximum of 370 GB of memory.

The actual technical implementation utilizes AllenAI's Specter repository†. Specter is built on top of custom modules such as model and data processing, but the foundational implementation is based on the AllenNLP library. At the time of writing the thesis, the version of AllenNLP (v 0.9.0) used in Specter had some deprecated dependencies. These dependencies such as *pytorch transformers*‡ library were partly updated to the more current transformers library (4.2.0) in order to upload *TurkuNLP/bert-base-finnish-cased-v1*§ tokenizer and pretrained model directly from Huggingface API. The implementation supports also loading the model from binary file but the compatibility with Huggingface API is flexible for further experimentations by only changing the pretrained model and tokenizer in the configuration file.

4.2 Models

This thesis aims to reproduce the SPECTER model by utilizing data outside the scientific literature to assess its suitability for various domains and languages. To achieve this, a pretrained model is chosen to accommodate the new domain and language. Given that, SPECTER focuses on scientific documents, its parameters were initial-

*https://research.csc.fi/csc-s-servers
†https://github.com/allenai/specter
‡https://huggingface.co/transformers/v1.2.0/index.html
§https://huggingface.co/TurkuNLP/bert-base-finnish-cased-v1

ized from SciBERT [6] which was already pretrained with scientific data. Following this logic, in this thesis, the pretrained model was chosen to support the domain and the language. Therefore, WikiSpecter was initialized from Finnish BERT parameters. Finnish BERT is a Finnish variant of BERT [1] and is pretrained with 21 million documents extracted from Finnish news, online discussions, and internal crawl [9] including Finnish Wikipedia articles. Since Finnish BERT already captures knowledge from the domain and language of fine-tuning objective, it suits well to the task. The tokenization of training instances (query, positive, negative) was performed with the Turku neural parser as in the original Finnish BERT pretraining [9].

The fine-tuned WikiSpecter is compared against two baseline models. The purpose of baseline models is to validate the impact of fine-tuning and the approach in general. The performance of WikiSpecter and baseline models is measured in two tasks - topic classification and cross-reference prediction. The results in the benchmarks directly contribute to the research questions about the applicability and performance of SPECTER beyond the scientific domain. For this purpose Finnish BERT [9] and Finnish Sentence BERT [10] are chosen as they both are already established and validated models in Finnish language and general domain. They are trained with *token-* and *sentence-level* objectives, whereas WikiSpecter is trained with document-level objectives.

Given that Finnish BERT serves as a foundational model for WikiSpecter, its performance against Wikispecter is crucial to demonstrate the benefit of the fine-tuning process. Despite being trained with a distinct objective, it has been exposed to data from Finnish Wikipedia during its pretraining phase.

For the second baseline model, Finnish Sentence BERT [10] is chosen. Its training objective is closely related to that of SPECTER, albeit with a distinct focus on capturing the semantic similarities between sentences. In contrast, SPECTER is designed to understand and learn similarity of documents while compressing the whole document into an embedding.

4.3 Evaluation Framework

The evaluation framework in the thesis is based on the SciDocs framework [5], which was originally designed to evaluate the performance of SPECTER on various benchmarks. Originally, the framework contains datasets and multiple evaluation tasks. In this thesis, the performance of the model is evaluated in two tasks, cross-reference prediction and article topic classification. The evaluation data is directly obtained from Wikipedia without additional human annotation. The performance is measured with the same metrics as in SPECTER but with different data and models. The focus of the

evaluation is on measuring the performance at the document level and the impact of fine-tuning Finnish BERT with the objective of document training compared to baseline models. This is in line with the research question and the analysis of the generic nature of the training framework. The results provide a direct indication how learned vector representations can be applied from fine-tuning to other tasks.

For evaluating WikiSpecter, no further fine-tuning or adjustments are needed. The embeddings serve as document features in classification tasks or alternatively as vectors in cross-reference prediction in which their distance can be measured directly in the vector space. While the document graph was crucial for sampling the training triplets, at inference time it is not needed even for embedding unseen Wikipedia articles. All in all, the fine-tuned model works as a general-purpose embedder which produces document vectors, and they can be used for downstream tasks as such.

4.4 Article Topic Classification

Article topic classification task requires a labeled dataset. The topic classification as an evaluation task follows a common supervised learning setting. The training data consist of labeled samples where each input (e.g., an article) has a corresponding target label (e.g., a topic). The model's training objective is to learn the relationships between inputs and outputs and eventually map the input to the correct output at inference time. Here, the Wikipedia articles are mapped to their topics, which are derived from Wikipedia's general classification system without additional human annotation. The train/test data for the classifier is produced by feeding the article first through the transformer-based language model (WikiSpecter, baselines), resulting in 768 dimensional *fixed-length* vectors. These vectors represent articles and are used directly as features for training the classification model.

The classification model chosen for the task is the linear support vector classifier (LinearSVC), the same as in SPECTER. Finding the best-performing classifier is beyond the scope of this work, and the goal is only to evaluate model performance when trained with different embeddings from each model. In LinearSVC training, the model's objective is to find a hyperplane in an embedding space that separates the data in classes as purely as possible. The model is trained with a three-fold cross-validation using the squared hinge loss. In addition, the training tries to minimize the generalization error by maximising the margin between the classes and thereby enabling the model to classify new unseen data which is used for testing the model performance.

Since the distribution of data over the class labels is unbalanced, the metrics selected for the tasks are selected accordingly. Macro *F1-score* [30] is used as one of the metrics to evaluate models in tasks with unbalanced datasets to decrease the impact

of uneven distribution. The *F1-score* is a harmonic average of precision and recall [31]. It provides deeper insights to the model performance compared to accuracy only which can be misleading when few classes dominates the dataset. However, also the accuracy is used as another classification metric which is simply correct classifications out of all predicted samples.

4.5 Cross-Reference Prediction

Cross-reference prediction is framed as a ranking task. The goal is to rank which articles are cross-referenced by a given query article from a set of candidate articles. In the evaluation dataset for each query article, 30 articles are selected as candidates. Out of these, 5 are cross-referenced by the query article, and 25 are randomly sampled non-cross-referenced articles or *hard-negatives*. Ideally, the model should rank the cross-referenced articles higher than the uncited ones. In total, 1000 query articles are sampled from the dataset which results in dataset size of 1000 x 30 = 30K articles. Also, this task does not require any additional downstream model or fine-tuning the WikiSpecter specifically for the task. Similarly to topic classification, the articles are embedded through the transformer LM, but now the embeddings represent vectors instead of features.

The comparison of the articles are based on vector representation v obtained from the Transformer LM. After embedding the articles, their distance is measured using the L2 distance. The closest article in vector space is considered to be the most similar to the query article in semantic space and vice versa. Visually, the cross-referenced candidates should form a cluster.

Additionally, a rule-based baseline is added to the evaluation, which sets a relevance score for candidates by the proportion of unique words in the query article that appear in the title of the candidate article. In contrast to BERT models here the larger score means higher relevance while in vector space the smaller distance represents closer relatedness. In order to optimize the accuracy of the matches, all sentences were first lemmatized with the Spacy library* , since the Finnish is a morphologically rich language.

For evaluating the results, standard ranking metrics **Mean Average Precision** (MAP) and **normalized Discounted Cumulative Gain** (nDCG) are used as evalution metrics. nDCG is often used to evaluate recommendation and information retrieval systems. The nDCG metric ranges from 0 to 1, with a score of 1 indicating a perfect match with the ideal order, while lower scores are signals of a poorer ranking. MAP is

*https://spacy.io/models/fi

another typical evaluation metric for search algorithms and recommenders.

5. Results and Discussion

The results of topic classification are shown in Tables 5.1 and 5.2. In the article topic classification task, it can be observed that WikiSpecter achieved higher *macro F1-score* and *accuracy* than the baseline models. However, in the second-level classification task, WikiSpecter obtained the highest *F1-score*, but Sentence BERT reached the highest accuracy with a +0.49 point difference to WikiSpecter. While accuracy is a metric of overall correctness, the higher *macro averaged F1-score* indicates that WikiSpecter handles minor classes better, which is crucial for balanced performance across all classes. In comparison to FinBERT, which is not fine-tuned with cross-references, WikiSpecter obtained higher performance in both tasks, which directly demonstrates the impact of fine-tuning. In order to understand misclassifications and the semantic capabilities of the model in-depth, the error analysis is provided in the following subsection.

It should be noted that all models performed better in the second-level topic classification task. Second-level topics are likely more detailed and more consistently labeled than broader top-level topics. This increased specificity might make it easier for the models to distinguish between the topics because the defining features of each category are more detailed and less overlapping. On the other hand, when the mapping of articles to labels in training data is consistent across different levels in class hierarchy tree, the model should obtain higher performance on top-level topics than on second-level. This is simply because fine-grained classification with more labels is more difficult in theory. Also, there is less data in the deeper levels per class, which could result in underfitting and poor performance, while the case here is opposite. This indicates that the article-to-label mapping may have some inconsistencies. In order to provide more insights, deeper analysis on prediction errors is done. I will go through incorrect model outputs and check whether these make sense semantically.

5.1 Error Analysis of Misclassifications

WikiSpecter's top-level results are visualized in confusion matrix in Figure 5.1. The Y-axis represents the true labels and the X-axis represents the predicted labels. It

Table 5.1: Top-level Classification

Model	F1	Accuracy
FinBERT	59.86	73.70
Finnish Sentence BERT	56.32	71.63
WikiSpecter	**65.83**	**76.12**

Table 5.2: Second-level Classification

Model	F1	Accuracy
FinBERT	67.90	80.66
Finnish Sentence BERT	68.02	**81.39**
WikiSpecter	**71.33**	80.90

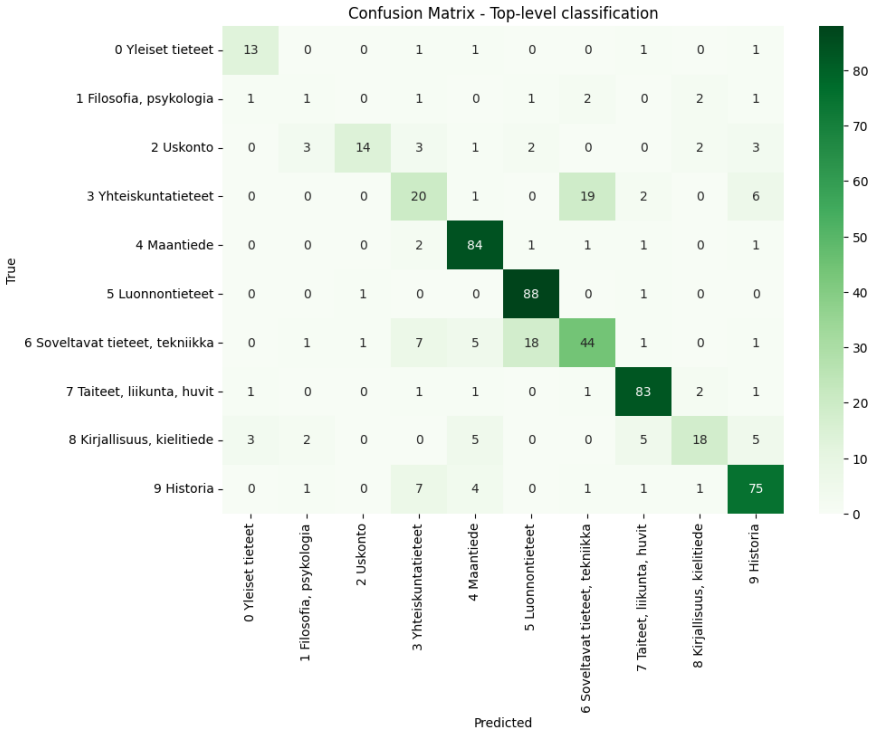

Figure 5.1: Confusion Matrix of top-level classifications

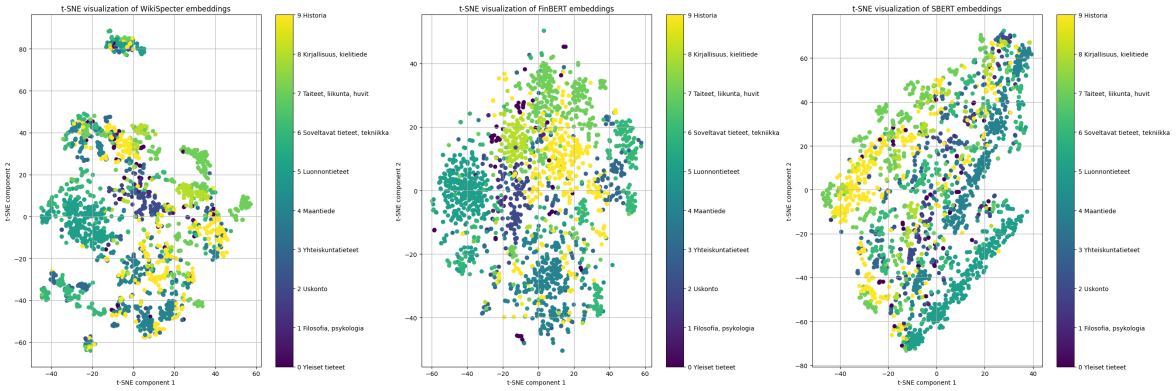

Figure 5.2: t-SNE visualization of article embeddings and their top-class topics of Wikipedia's general classification system. WikiSpecter is on the left hand side, Sentence BERT on the right hand side and Finnish BERT in the middle.

is noteworthy that while some predictions are technically incorrect, a subjective evaluation reveals that these misclassifications may still be semantically relevant. For example, many of the articles labeled as *Yhteiskuntatieteet (Social Sciences)* are classified incorrectly 19 times as *Soveltavat tieteet, tekniikka (Applied Science, Technology)*, which is intuitively distant from the correct class. In addition, the same class was predicted as *Historia (History)* for 6 times, which is semantically closer to how classes are related, for example, as school subjects.

After examining the article titles of incorrect predictions, some of the articles misclassified as *Applied Sciences and Technology* or *History* belong to the sub-class *National Defense (army related)*, which has articles about war machinery such as tanks, fighter planes, and also about different battles. While the predictions are technically incorrect, semantically they make sense. Also, class *Kirjallisuus, kielitiede (Literature)* has articles about writers that were classified as *History*. Well-known writers from previous centuries can be considered as historic persons. It seems that articles can overlap on multiple topics while they are only labeled to a single topic. Figure 5.2 also supports this interpretation. It shows t-SNE [33] visualizations of article projections in 2-dimensional space for articles in top-level classification task for WikiSpecter (left-hand side) and the baseline models. It can be observed that WikiSpecter clusters the topics into smaller groups having higher separation of clusters compared to the baseline models. The embeddings in History class (yellow dots) are overlapping with other topics as interpreted earlier in this section.

Even for humans, it is not self-evident what the correct topics are for the previous examples. From this perspective, incorrect predictions may remain relevant contextually, but the single-class classification itself requires a deeper analysis to gain true insights of the model's performance and the quality of the vectors.

The model's predictions can be divided into three categories: 1) correct, 2) in-

Table 5.3: Cross-reference Prediction

Model	MAP	nDCG
Rule-based baseline	60.08	79.02
Finnish BERT	48.92	68.74
Finnish SentBERT	60.30	78.96
WikiSpecter	**75.72**	**88.2**

correct but semantically correct, and 3) incorrect, semantically incorrect. Since the measure of semantic correctness is subjective here, the numerical evaluation is beyond the scope of the results section. However, it seems that understanding the true performance of the model requires deeper analysis than simply stating misclassifications as incorrect. Essentially, what is interesting, is whether the fine-tuning paradigm can create a model that captures meaningful relationships across topics in new language and perform well on unseen data.

The analysis shows that in most cases incorrect model output is due to inconsistencies in mapping of second-class labels into the top class in Finnish Wikipedia's general classification system. Other types of error are purely incorrect, which can be rooted in an unbalanced dataset in this task and some variation in the test data such as short introduction.

5.2 Cross-Reference Prediction

The results of the cross-reference prediction are presented in Table 5.3. In this task, WikiSpecter achieved significantly higher score on both MAP and nDCG than the baseline models. The rule-based model, which ranks articles based on the frequency of emmatized word occurrences in the title of candidate articles, was more effective in identifying relevant articles according to both metrics compared to FinBERT. Also, it outperformed SentBERT on nDCG score. The rule-based model's straightforward approach captured strong lexical matches and the model seem to be particularly powerful since majority of the articles are linked to the title term which occur in the articles as such as described in Chapter 3. However, the rule-based model learns semantic similarities insufficiently and thus its sub-optimal solution for high-quality ranking. WikiSpecter instead learnt these similarities effectively and obtained the highest results in the task.

WikiSpecter was fine-tuned with signals to external articles (cross-references), which is the objective that is measured in cross-reference prediction. As the cross-reference prediction is essentially a ranking task, the fine-tuning objective is optimal

for the task. The learnt representations take into account both content and cross-reference relation and in this task, no additional classifier was trained. The closeness of article vectors were measured with L2 distance in vector space only.

The performance of WikiSpecter shows that the model can rank cross-references for new unseen articles by showing the title and part of introduction to the model only. The results are also aligned with those achieved in SPECTER which indicate that the replication of fine-tuning in a new domain and language is implemented successfully.

In general, the results in both tasks shows that fine-tuning has a positive effect on the model's performance in document-level tasks. In this context, this means that the model can learn high-quality vector representations with a triplet loss function, leverage the learned representations from one task, and apply them successfully to another. Also, the presented results work as an evidence that the SPECTER framework is applicable outside the scientific domain and that similar results to those achieved by SPECTER can be obtained in the general domain which was one of the research objectives of this thesis. Additionally, performance on the benchmarks indicates that cross-references in Finnish Wikipedia have high precision of relevance and help the model to learn relationships between different topics even though the conventions of Wikipedia writing are more flexible compared to scientific domain.

5.3 Further Research and Potential Applications

This thesis closely replicated the methodology of the SPECTER paper, including the selection of hyperparameters and training architecture based on empirical studies. However, the optimal settings identified in the scientific domain may not directly translate to the general domain. This leaves room for further experimentation, including testing different hyperparameters, exploring various similarity scores, adjusting the number of positive and negative documents, and determining the optimal number of training epochs.

The articles were encoded as a concatenation of the title and the first paragraph of the introduction. The impact of using other text fields or a longer context, such as the full introduction, was examined. SPECTER authors suggested [5] that using the full text of documents could improve the quality of learned representations and consequently enhance performance across evaluation tasks. Although BERT-based models support up to 512 tokens, this study used a context length of 128 tokens. Future studies could explore utilizing the full 512-token capacity of BERT for more comprehensive text coverage. While longer texts beyond 512 tokens might offer additional benefits, BERT's limitation to 512 tokens would need to be addressed, potentially through text chunking or other strategies.However, while the complete texts of scientific articles may

not always be freely accessible, Wikipedia data is generally open for all. The availability enables straightforward experimentation on top of this thesis with different text lengths, offering a clear path for more experimentation.

The training dataset was built by sampling triplets from the document graph. In this thesis, the graph was built from a limited set of cross-references which appear only in the first paragraph of the introduction. Thus, the cross-references in the later parts were not visible and available for the training instance sampler. Intuitively, the embeddings capture information more from high-level topics than details that occur later in the articles. By allowing the model to utilize more cross-references in triplet loss, deeper insights and more complex relationships could be captured during training.

The WikiSpecter was initialized from a pretrained model. In order to serve the Finnish language and general domain, SciBERT [6] was replaced with Finnish BERT [9] since the pretrained model was already trained with data in Finnish. The impact of the selected pretrained model was left uncovered on the results. However, it seems obvious that the initial model should already capture knowledge from the domain and language prior to fine-tuning. Finnish BERT fulfills these conditions, but other models could be experimented.

As shown in the thesis, fine-tuning is a cost-efficient way to train models to solve narrow problems with a small amount of domain-specific data. Intuitively, similar fine-tuning with the SPECTER framework could be applied to other networked data sources like news data or social networks. In addition, incorporating data from other modalities beyond text, such as images, into document embeddings could lead to improved performance in document-level tasks. Studying other modalities are interesting especially on data sources having low textual data available. Both SPECTER and WikiSpecter were evaluated in tasks with a single language. In order to build multilingual general-purpose tools, the framework's applicability to multi-lingual contexts could be studied to attract wider audiences.

WikiSpecter is also a potential model for various applications in many domains. For example, the model is a potential tool for automating the cross-reference suggestion when writing a new article. Such tool would also be possible to fit to other data sources like news data, which has a similar document links than SPECTER and WikiSpecter, by fine-tuning a model using a SPECTER framework in a new domain. Also, the model could be an efficient tool in searching documents which share the semantic similarity. The ability to compare semantically similar articles can also be used for verifying and fact-checking.

6. Conclusions

This thesis addressed two main research questions. Firstly, it investigated whether the structure of Finnish Wikipedia data could be utilized to enhance language model training. Secondly, it examined the generalizability of the SPECTER training framework, aiming to support the claim that this framework and the associated document graph are agnostic to both domain and language. This was demonstrated by showing that fine-tuning with this framework improves language model performance in document-level tasks beyond scientific literature.

The experimental results in this thesis demonstrate that other data sources, particularly Finnish Wikipedia, can be effectively utilized to train a new language model with the SPECTER framework, surpassing baseline performance in benchmarks. In practice, the fine-tuned WikiSpecter functions as a general-purpose model, and its learned representations can be successfully applied to downstream tasks without the need for additional fine-tuning.

The SPECTER framework can be effectively used if the following key components are present in the data. First, textual components, such as the title and summary of the document (e.g., abstract or introduction), are required to encode the document into the Transformer language model as an input. Second, references or hyperlinks to other related items, in this context articles, are needed to build a document graph. These key components were found in Finnish Wikipedia, and structurally, the dataset suited this task well.

To transform the raw data into suitable transformer inputs, a significant amount of data preprocessing was required, which is a common step in NLP research. Despite various differences between the data sources, fine-tuning improved the language model's performance in topic classification and citation ranking tasks in the new domain and language, compared to baseline models. The positive performance indicates that the fine-tuning was successfully implemented and that the model is capable of producing accurate document representations. Additionally, the cross-references from one Wikipedia article to another are relevant to the source topics, as the semantic relationships are captured by the model through these links.

The semantic quality of the embeddings was directly measured in the evaluation

tasks and further assessed through additional manual analysis of misclassifications. In summary, the document graph and the SPECTER training framework were successfully applied to a new domain and language, supporting the second research question about the data and domain-agnostic nature of the SPECTER framework and document graph.

As of writing the thesis, NLP as a research field is thriving and has reached multiple pivotal milestones in language understanding and generation. The field is attractive and current as the effectiveness of large language models has been demonstrated both in the academic research but also in the commercial applications in multiple modalities by tech companies. Large language models are already everyday tools for various professionals in a wide range of areas having clear benefits but also concerns. In general, these tools realize the impact of NLP research across domains.

Finally, I want to thank my thesis supervisor, Professor Lidia Pivovarova, for invaluable support and inspiration throughout the thesis process. Also I want to show gratitude to IT Center for Science for granting me access to the computing resources which significantly enhanced the computational aspect of this thesis.

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[5] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *ACL*, 2020.

[6] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. pages 3613–3618. Association for Computational Linguistics, 2019.

[7] Liat Ein Dor, Yosi Mass, Alon Halfon, Elad Venezian, Ilya Shnayderman, Ranit Aharonov, and Noam Slonim. Learning thematic similarity metric from article sections using triplet networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. Graph-based neural multi-document summarization.

In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[9] Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*, 2019.

[10] Jenna Kanerva, Filip Ginter, Li-Hsin Chang, Iiro Rastas, Valtteri Skantsi, Jemina Kilpeläinen, Hanna-Mari Kupari, Jenna Saarni, Maija Sevón, and Otto Tarkka. Finnish paraphrase corpus. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa'21)*, pages 288–298. Linköping University Electronic Press, Sweden, 2021.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] Understanding searches better than ever before. https://blog.google/products/search/search-language-understanding-bert/. Accessed: 2024-01-01.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[15] ROBERT HECHT-NIELSEN. Iii.3 - theory of the backpropagation neural network**based on ânonindentâ by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593â611, june 1989. Â© 1989 ieee. In Harry Wechsler, editor, *Neural Networks for Perception*, pages 65–93. Academic Press, 1992.

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

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[18] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, OpenAI, 2018.

[20] Wilson L Taylor. "Cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[21] Introducing the center for research on foundation models. https://hai.stanford.edu/news/introducing-center-research-foundation-models-crfm, 2021. Accessed: 2024-03-01.

[22] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[23] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. LinkBERT: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[24] Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew Peters, Arie Cattan, and Ido Dagan. CDLM: Cross-document language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2648–2662, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[25] Wikipedia:list of policies and guidelines. https://en.wikipedia.org/wiki/Wikipedia:List_of_policies_and_guidelines. Accessed: 2024-03-06.

[26] Wikipedia:ignore all rules. https://en.wikipedia.org/wiki/Wikipedia:Ignore_all_rules. Accessed: 2024-03-06.

[27] Giusepppe Attardi. Wikiextractor. https://github.com/attardi/wikiextractor, 2015.

[28] Template:main. https://en.wikipedia.org/wiki/Template:Main. Accessed: 2024-03-06.

[29] Luokka: Yleinen wikiluokitusjarjestelma. https://fi.wikipedia.org/wiki/Luokka:Yleinen_wikiluokitusj%C3%A4rjestelm%C3%A4. Accessed: 2024-03-06.

[30] Nancy Chinchor. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding*, MUC4 '92, pages 22–29, USA, 1992. Association for Computational Linguistics.

[31] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15(93):3221–3245, 2014.

Appendix A. Relationship Between Top-Level to Second-Level Topics

- 0 Yleiset tieteet / General Sciences
  - 00 Kirja-ala, kirjoitus / Publishing, Writing
  - 02 Kirjastotoimi, kirjastotiede, informatiikka / Librarianship, Library Science, Informatics
  - 06 Yleinen kulttuuripolitiikka / General Cultural Policy
  - 07 Viestintä, joukkoviestintö / Communication, Mass Communication
- 1 Filosofia, psykologia / Philosophy, Psychology
  - 11 Filosofia / Philosophy
  - 14 Psykologia / Psychology
  - 15 Rajatieto / Parapsychology
  - 16 Tietoteoria, tieteenfilosofia / Epistemology, Philosophy of Science
  - 17 Etiikka / Ethics
- 2 Uskonto / Religion
  - 20 Uskonnot yleisesti / Religions in General
  - 21 Uskonnonfilosofia / Philosophy of Religion
  - 23 Hindulaisuus / Hinduism
  - 24 Buddhalaisuus / Buddhism
  - 26 Juutalaisuus / Judaism
  - 27 Kristinusko / Christianity
  - 28 Islam / Islam
  - 29 Muut uskonnot, mytologiat / Other Religions, Mythologies
- 3 Yhteiskuntatieteet / Social Sciences
  - 30 Yleinen yhteiskuntatiede / General Social Science
  - 31 Tilastotiede / Statistics
  - 32 Politiikka / Politics

  - 33 Oikeustiede / Law
  - 34 Aluesuunnittelu, aluepolitiikka / Regional Planning, Regional Policy
  - 35 Hallinto / Administration
  - 36 Talous / Economics
  - 37 Sosiaalipolitiikka / Social Policy
  - 38 Kasvatus, opetus / Education
  - 39 Maanpuolustus / National Defense
- 4 Maantiede / Geography
  - 41 Eurooppa / Europe
  - 42 Aasia / Asia
  - 43 Afrikka / Africa
  - 44 Pohjois-Amerikka / North America
  - 45 Etelä-Amerikka / South America
  - 46 Australia, Oseania, Napamaat / Australia, Oceania, Polar Regions
  - 49 Kansatiede, antropologia / Ethnology, Anthropology
- 5 Luonnontieteet / Natural Sciences
  - 51 Matematiikka / Mathematics
  - 52 Tähtitiede / Astronomy
  - 53 Fysiikka / Physics
  - 54 Kemia / Chemistry
  - 55 Geologia / Geology
  - 57 Biologia / Biology
  - 58 Kasvitiede, sienitiede / Botany, Mycology
  - 59 Eläintiede / Zoology
- 6 Soveltavat tieteet, tekniikka / Applied Sciences, Engineering
  - 61 Lääketiede, terveydenhuolto / Medicine, Healthcare
  - 62 Tekniikka, kulkuneuvot / Engineering, Vehicles
  - 64 Kemianteollisuus, kaivosteollisuus / Chemical Industry, Mining Industry
  - 65 Tekstiili- ja vaatetusteollisuus / Textile and Clothing Industry

  - 66 Rakennustekniikka / Construction Engineering
  - 67 Maa- ja metsätalous / Agriculture and Forestry
  - 68 Kotitalous / Home Economics
  - 69 Liiketalous / Business Economics
- 7 Taiteet, liikunta, huvit / Arts, Sports, and Recreation
  - 70 Taide, taidehistoria / Art, Art History
  - 71 Uudet taidemuodot / New Forms of Art
  - 72 Arkkitehtuuri / Architecture
  - 73 Kuvanveisto / Sculpture
  - 74 Maalaustaide, piirustustaide / Painting, Drawing
  - 75 Graafinen taide, valokuvaus / Graphic Art, Photography
  - 76 Taideteollisuus, taidekäsityö / Decorative Arts, Crafts
  - 77 Näyttämötaide, elokuvataide / Performing Arts, Film Art
  - 78 Musiikki / Music
  - 79 Urheilu, pelit, huvit / Sports, Games, Entertainment
- 8 Kirjallisuus, kielitiede / Literature, Linguistics
  - 81 Kansanrunous / Folk Poetry
  - 82 Runot / Poems
  - 83 Näytelmät / Plays
  - 84 Kertomakirjallisuus / Narrative Literature
  - 85 Sadut, kuvakirjat, sarjakuvat / Fairy Tales, Picture Books, Comics
  - 86 Kirjallisuustiede / Literary Studies
  - 87 Kielitiede / Linguistics
- 9 Historia / History
  - 90 Historian tutkimus / Historical Research
  - 91 Maailmanhistoria / World History
  - 92 Euroopan historia / European History
  - 93 Aasian historia / Asian History
  - 94 Afrikan historia / African History

- 95 Pohjois-Amerikan historia / North American History
- 96 Etelä-Amerikan historia / South American History
- 97 Australian, Oseanian ja napamaiden historia / Australian, Oceanian, and Polar Regions History
- 99 Henkilö- ja sukuhistoria / Personal and Family History