FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Semantic Textual Similarity in Czech
Student:	Jiří Bednář
Supervisor:	Ing. Daniel Vašata, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

The task of Semantic Textual Similarity (STS) is to determine how semantically similar two pieces of text are. One usually compares two sentences, paragraphs, or even whole documents. Recently, there have been two main approaches to solving this problem using deep neural networks. The first approach is the cross-encoder. Its input is the concatenation of two sentences to compare and it predicts their similarity. The second is the bi-encoder, which encodes each sentence into a vector space such that the vectors of two semantically similar sentences are close to each other. The advantage of the second approach is that it can be used for fast semantic search, where we only need to compare vectors. Cross-encoder, on the other hand, tends to be more accurate. There are plenty of known methods to train such models, but they usually need a lot of labeled data. Such data is not always available, for example, in the Czech language, and could be very expensive to annotate.

The aim of the thesis is to review the state-of-the-art methods that can encode a Czech sentence into a vector space such that the vectors of two semantically similar sentences are close to each other, i.e., they can be used as bi-encoders in the STS task. Due to the small amount of annotated data in the Czech language, the work should mainly focus on unsupervised learning methods.

The specific points of the assignment are:
1. Show that standard methods like Masked Language Modeling do not work well for the STS task.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 11 February 2023 in Prague.

2. Research the current sentence embedding methods suitable to be used as bi-encoders for the STS task.
3. Use these methods to train several Czech STS models. Build a validation dataset and evaluate them.
4. Compare the results of STS models with other pre-trained models on downstream tasks such as sentiment analysis, retrieval, or relevance and observe their behavior on different amounts of available labeled data.
5. Explore methods for fine-tuning STS models and evaluate their benefits.

Bachelor’s thesis

SEMANTIC TEXTUAL SIMILARITY IN CZECH

Jiří Bednář

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Vasata Daniel, Ph.D.
May 11, 2023

Contents

List of Figures

List of Tables

List of code listings

First, I want to thank my amazing supervisor, Ing. Daniel Vašata, Ph.D. for the insightful feedback, constant encouragement, and helpful direction. I am also grateful to Mgr. Jakub Náplava, Ph.D. His valuable input, constructive criticism, and constant support have played a major role in shaping the outcome of this research. Their expertise and guidance have greatly contributed to the development and refinement of this work. Furthermore, I would like to acknowledge the generous resources and facilities provided by Seznam.cz. Thank you all for your contributions.

Declaration

In Prague on May 11, 2023 ....

Abstract

Recent significant advancements in semantic textual similarity (STS) have primarily been driven by the availability of annotated data for English, a luxury that Czech and other low-resource languages often lack. In this thesis, we investigate the challenges and potential improvements in solving the STS problem for the Czech language. Our research explores advancements in neural networks, including the Transformer architecture and pre-trained language models such as BERT, RoBERTa, and ELECTRA. We provide an extensive study of techniques and models for STS, as well as methods for generating sentence embeddings. Additionally, we discuss Cross-encoder and Bi-encoder architectures, along with advanced training methods like SimCSE, TSDAE, Trans-Encoder, and Multilingual distillation. We present our STS models trained using these techniques and evaluate their performance on STS and two downstream tasks. Through our analysis, we highlight our best STS model, which sets multiple state-of-the-art results, demonstrating the potential for future advancements in STS for low-resource languages.

Keywords Czech, Semantic Textual Similarity, Neural Networks, Transformer, Pre-trained Language Models, BERT, RoBERTa, ELECTRA, sentence embedding, Cross-encoder, Bi-encoder, SimCSE, TSDAE, Trans-Encoder, Multilingual distillation, state-of-the-art, low-resource languages

Abstrakt

Nedávné pokroky v problému sémantické textové podobnosti známé jako STS se uskutečnili především díky dostupnosti velkého množství anglických anotovaných dat, což je luxus, který čeština a další méně rozsáhlé jazyky často postrádají. V této práci se zabýváme výzvami a možnými zlepšeními při řešení problému STS pro češtinu. Zkoumáme pokroky v oblasti neuronových sítí, včetně architektury Transformeru a předtrénovaných jazykových modelů, jako jsou BERT, RoBERTa a ELECTRA. Poskytujeme rozsáhlou studii technik, modelů pro STS, a také metod pro generování embeddingů vět. Dále se zabýváme architekturami Cross-encoder a Bi-encoder spolu s pokročilými metodami trénování, jako jsou SimCSE, TSDAE, Trans-Encoder a vícejazyčná destilace. Představujeme naše STS modely natrénované pomocí těchto technik a vyhodnocujeme je na STS a dvou dalších úlohách. Analyzujeme náš nejlepší STS model, který stanovuje několik state-of-the-art výsledků, což ukazuje potenciál pro budoucí pokrok v oblasti STS pro jazyky s menší dostupností dat.

Klíčová slova čeština, sémantická textová podobnost, neuronové sítě, Transformer, předtrénované jazykové modely, BERT, RoBERTa, ELECTRA, větný embedding, Cross-encoder, Bi-encoder, SimCSE, TSDAE, Trans-Encoder, vícejazyčná destilace, state-of-the-art

List of Abbreviations

BE	Bi-Encoder
BPE	Byte-Pair Encoding
CFD	Czech Facebook Dataset
CE	Cross-Encoder
FFN	Feed Forward Neural network
GRU	Gated Recurrent Units
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
MNRL	Multiple Negative Ranking Loss
NLP	Natural Language Processing
NN	Neural Network
NLI	Natural Language Inference
NSP	Next Sentence Prediction
OOV	Out-Of-Vocabulary
PCA	Principal Component Analysis
PLM	Pre-trained Language Model
RRN	Recurrent Neural Network
STS	Semantic Textual Similarity

Introduction

This introductory chapter sets the stage for the thesis, which investigates the task of semantic textual similarity (STS) in the context of natural language processing. It outlines the importance and challenges of STS, as well as its applications. It highlights the difficulties faced by low-resource languages like Czech and the need for alternative training methods.

Natural Language Processing (NLP) is a rapidly evolving field of computer science that focuses on developing algorithms and models to enable machines to understand human language. One fundamental task in NLP is determining the degree of semantic similarity between two pieces of text, a challenge known as Semantic Textual Similarity (STS). STS plays a crucial role in various applications, including information retrieval, text classification, question answering, and machine translation. In the context of information retrieval, STS can be applied to tasks such as document clustering or search result ranking, where the goal is to identify and retrieve the most relevant documents based on semantic similarity to a given query. Examples of STS in information retrieval include Latent Semantic Analysis (LSA) [1] and its variants, as well as more recent methods based on deep learning, such as BERT and other transformer-based models which we present in this work.

STS is a difficult problem due to several reasons, such as the inherent ambiguity of natural language, the presence of synonyms, and the complexity of linguistic structures. For example, an easy STS case might involve comparing the sentences:

The cat is on the mat.
The feline is on the rug.

where the semantic similarity is apparent due to the use of synonymous words. In contrast, a more challenging example could be comparing the sentences:

The coastal city was left in ruins after the tempest.
The storm's aftermath led to devastation in the seaside metropolis.

where the similarity is less obvious due to the different linguistic structures.

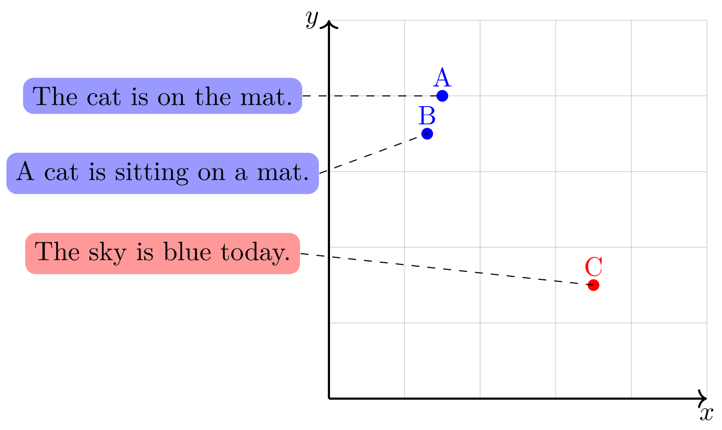
Despite these difficulties, significant advancements have been made in STS for English, mainly due to the availability of annotated data. This has facilitated the development of high-performance supervised models, which typically outperform other methods. However, the same progress cannot be observed for Czech or other low-resource languages, which lack comparable amounts of labeled data or already trained public models. This necessitates alternative methods for training STS models in Czech, especially those that do not require expensive manual annotation.

The aim of this thesis is to investigate and compare various deep learning methods for training STS models in Czech, primarily focusing on unsupervised and semi-supervised techniques. We specifically examine STS models that generate meaningful sentence embeddings, enabling them to map each sentence to a vector space where semantically similar sentences are close, as depicted in Figure 1. By utilizing available resources such as monolingual and bilingual corpora, we train these models without the need for large amounts of annotated data. Additionally, we explore the effectiveness of STS models in fine-tuning various downstream tasks and compare them with other techniques.

Chapter 1 delves into the complexities of the STS task, outlines the steps for addressing it, and discusses related challenges. We utilize Neural Networks (NNs) to tackle this task, explaining why they are well-suited for the problem. We provide a historical overview of deep NN-based approaches for STS is Section 1.1, describing various NN models and their architectures, with a focus on transformer and transformer-based models. Our investigation in Chapter 2 includes training techniques for both labeled and unlabeled data for solving STS, with a focus on Bi-encoder models.

We present state-of-the-art English models in Section 1.3 and discuss the transfer of knowledge from these models to develop new models capable of understanding the Czech language. In Chapter 3, we introduce the training and test datasets, including a new STS test dataset and its creation process. Additionally, two other NLP tasks are presented as downstream tasks in Section 3.2, along with their respective test datasets, where we also explain the metrics used to evaluate the models.

The subsequent Chapter 4 details our experiments with the training of STS models using the aforementioned methods, discussing hyperparameter search and the practical challenges of training. Finally, in Chapter 5, we report the evaluation results, compare the models based on the methods employed, and provide an in-depth discussion of these results. A summary of all findings is presented at the end of the work.



■ **Figure 1** A 2D visualization of the sentence embedding space for an STS model. Semantically similar sentences are positioned closely together (A, B), while semantically dissimilar sentences are farther apart (A, C). This representation illustrates the model's ability to capture the semantic similarity of sentences.

Chapter 1

From STS to Neural Networks

In this chapter, we explore the various techniques and models for solving the STS problem, focusing on the advancements in neural networks for NLP and their application to sentence embeddings. We delve into the different types of pre-trained language models including the state-of-the-art sentence embedding models. We introduce the Hugging Face Model Hub as a platform where these models can be found. Additionally, we examine strategies for converting word embeddings to sentence embeddings. We then investigate the limited availability of Czech pre-trained models. Through this comprehensive overview, we provide a solid foundation for understanding the development and application of neural networks and PLMs in solving the STS problem.

Semantic textual similarity is a complex problem involving the comparison of two pieces of text, referred to as sentences, and predicting their semantic similarity. Formally, our goal is to find a similarity score $s(A, B)$, where s is a similarity function that takes A and B as inputs and returns a similarity score in the range of $\langle 0, 1\rangle$[1], with 0 representing no similarity and 1 meaning identical in meaning.

One of the main challenges is transforming sentences into a format that computers can understand. Several approaches involve mapping each word to a fixed-length vector representation known as a word embedding, which captures the semantic meaning of the word. The embedding vectors must be trained based on the location of each corresponding word in a text. Typically, the number of word embeddings is restricted by a pre-selected vocabulary. Pre-trained word embeddings like Word2Vec [2] and GloVe [3] have limitations in handling out-of-vocabulary (OOV) words, as they assign the same vector to all unfamiliar words. FastText [4] embeddings address this limitation by viewing words as n-grams, allowing it to handle OOV words. However, these approaches assign identical vectors for the same word in different contexts, leading to confusion. For example:

- The construction workers used a **crane** to lift the steel beams into place on the skyscraper.
- We spotted a beautiful **crane** wading in the pond, gracefully extending its long neck to catch fish.

Converting word embeddings into sentence embeddings, which are fixed-size vectors for sentences, is crucial. It enables the representation of an entire sentence's meaning in a fixed-size vector, facilitating the comparison of sentences with varying lengths and capturing the semantic relationships between them. This conversion is essential for effectively solving STS and other

[1]The similarity score range may vary (for example $\langle 0, 10\rangle$), but it can always be normalized to $\langle 0, 1\rangle$.

NLP tasks that require a comprehensive understanding of textual context. One common technique to create sentence embeddings is to simply average all word embeddings from the sentence. However, it lacks information about word positions within the sentence, which can be a critical factor, as demonstrated in the following example:

- At first, the film was terrible, but it turned out to be amazing.
- At first, the film was amazing, but it turned out to be terrible.

BERT embeddings [5] address these issues by effectively managing OOV words and attributing unique vectors to identical words depending on their contextual relevance. Its embeddings are contextualized and incorporate surrounding word and positional information. Using BERT embeddings has been demonstrated to be highly efficient and remains in use today [6, 7]. We delve into BERT embeddings in detail later in this chapter.

By using sentence embeddings, we can view the STS task as determining the similarity $s(A, B) = f(concat(eA, eB))$, where eA and eB represent the sentence embeddings of A and B, respectively. Neural networks have been shown to be more adept at handling complex data, and we explore various approaches for utilizing sentence embeddings and neural networks to effectively solve STS.

1.1 Evolution of Neural Networks in NLP

For many years, neural networks have been essential tools in NLP, boasting a rich history of development and advancement. Early neural networks used for NLP were typically shallow and relied on hand-engineered features [8]. However, with the advent of deep learning techniques and the availability of large datasets, neural networks have become increasingly powerful and widely used in NLP applications.

One of the earliest types of neural networks used for NLP was the Recurrent Neural Network (RNN) [9]. Initially used for speech recognition tasks, RNNs were quickly adopted for language modeling tasks, where they showed significant improvements over traditional n-gram models [10]. RNNs were particularly well-suited for modeling sequences of data, such as sentences or documents, and were able to capture short-term dependencies in text.

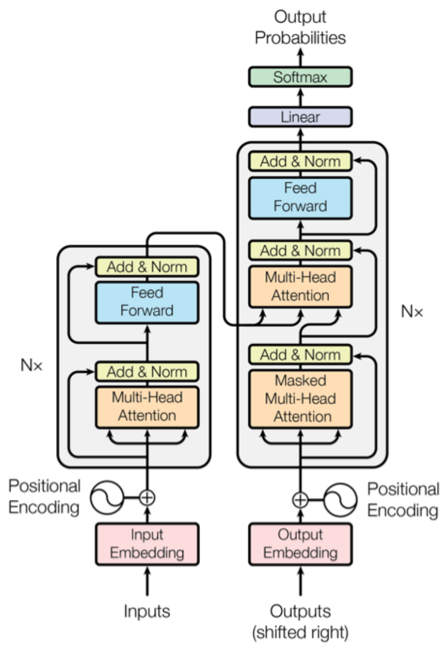
A significant issue with RNNs was the difficulty in training them to learn long-term dependencies. To address this, researchers developed more advanced architectures, such as Long Short-Term Memory (LSTM) [11] and Gated Recurrent Units (GRUs) [12], which are better at capturing them.

In recent years, a new type of neural network called Transformers [13] has emerged, becoming the state-of-the-art approach for many NLP tasks. Introduced in 2017, Transformers gained popularity due to their ability to capture long-term dependencies and process input sequences in parallel, making them much faster than RNN-based models. The Transformer architecture is still widely used in almost every state-of-the-art NLP model today [5, 14].

1.1.1 Transformer

The Transformer is designed to process sequential input data. In contrast with RNNs, it can process the whole input sequence at once with its attention mechanism, which is the key component for its architecture. The attention mechanism allows for contextual understanding of any position within the input sequence. This means that if the input is a natural language sentence, the Transformer does not need to process each word individually.

A Transformer model consists of an encoder and a decoder. The encoder takes the input sequence and applies a series of self-attention and feed-forward layers to produce a sequence of hidden representations, which can be seen as contextualized representations of input words.



■ **Figure 1.1** A detailed illustration of the Transformer architecture [13], with the encoder on the left and the decoder on the right.

The decoder then takes the hidden representations from the encoder and generates an output sequence by applying masked self-attention and encoder-decoder attention layers. The self-attention layer can access all previous states and weigh them according to a learned measure of relevance, providing relevant information about far-away tokens. The entire architecture is depicted in Figure 1.1. For detailed information about the architecture, please refer to [13].

1.1.2 Pre-trained Language Models

The Transformer architecture was quickly adopted, leading to the development of two new language model architectures. The first was an Encoder-based model, which involved stacking multiple encoder layers from Transformers to extract features from text and leveraging them for classification or regression tasks. The second category comprised Decoder-based models that used stacked decoders from Transformers primarily for tasks such as text generation or summarization. However, the original Transformer architecture remained focused on tackling machine translation. The main focus of our work is on Encoder-based models.

1.1.2.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers [5], is one of the most widely used Encoder-based models. Introduced in 2018, BERT can handle single or multiple natural sentences as input sequences with up to 512 tokens, effectively addressing a diverse set of downstream tasks. To convert sentences into tokens, BERT employs WordPiece embeddings[15] with a vocabulary size of 30,000 tokens. On average, English text consists of approximately 2-3 tokens per word. This process, called tokenization, uses a tokenizer, which also stores the vocabulary. The first token of each input sequence is always a special classification token (*CLS*), whose final hidden representation is used for classification tasks. To distinguish between two sentences, a *SEP* token is inserted between them. Each token's embedding is subsequently enriched with positional embedding and passed through the 12 encoder layers, the default configuration for BERT's base architecture. Another important constant for BERT base is the hidden size of 768, indicating the embedding size of each token and its hidden representations. There are various sizes of pre-trained language models, such as *small*, *base*, and *large*, which differ mainly in their model complexity, number of layers, and hidden size. The *base* size is commonly used and has proven to be very effective in numerous applications. While larger models often outperform smaller ones, they can require significant computational resources and take longer to train, especially on a single GPU.

BERT pre-trains deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Two unsupervised tasks were proposed for this purpose: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, 15% of input tokens are masked i.e. they are replaced with *MASK* token, and the model is trained to predict these tokens. In NSP, the model is provided with two sentences, with the second sentence being either the following sentence or a random sentence from the training corpus in 50% of instances. The model is then tasked with predicting whether the two sentences are consecutive or not.

Pre-trained language models (PLMs) offer the key benefit of efficient fine-tuning after computationally expensive pre-training. Using the pre-trained BERT encoder to encode input sentences, training efforts can focus solely on the classifier or regressor that takes the specialized *CLS* representation generated by BERT. Moreover, training the entire pre-trained model often yields better performance than using a newly initialized model without pre-training.

1.1.2.2 RoBERTa

RoBERTa: A Robustly Optimized BERT Pretraining Approach [16] is another highly popular pre-trained model. This replication study of BERT pretraining measured the impact of key hyperparameters and training data size, revealing that BERT was significantly undertrained and could be greatly improved. RoBERTa is trained with longer sequences, larger batches, and more data than BERT, and it completely removes the NSP objective. In contrast to BERT's sentence pairs, RoBERTa uses doc-sentences as input. Each input consists of full sentences sampled contiguously from a single document, with a maximum length of 512 tokens. The input sequence begins with the *CLS* token, and each full sentence is separated by the *SEP* token. These inputs are typically longer and provide more contextualized text representation than the sentence pairs used in BERT. By eliminating the need for sentence pairs, RoBERTa can focus on extracting richer contextual information from single, continuous text segments. The authors of RoBERTa have demonstrated that eliminating the NSP objective, when combined with doc-sentences as input, can either match or slightly enhance the performance of language models.

Other modifications in RoBERTa include dynamically changing the masking pattern every epoch, enhancing robustness, and using a variant of Byte-Pair Encoding (BPE) [17] that employs a smart implementation [18] of BPE with bytes instead of Unicode characters as subword units, resulting in a vocabulary of only 50k units that can still effectively encode large amounts of text.

1.1.2.3 ELECTRA

The authors of ELECTRA [19], which stands for Efficiently Learning an Encoder that Classifies Token Replacements Accurately, aimed to enhance the classic MLM learning objective to be more efficient when given the same amount of data and computation. They proposed a new pre-training approach called Replaced Token Detection, in which the model learns to distinguish ”real” input tokens from plausible but synthetically generated ”fake” replacements. Therefore, instead of masking the tokens with *MASK*, they are replaced with high-quality negative samples generated by a smaller generative network.

The key advantage of this task is that the model learns from all input tokens instead of just the small masked-out subset, making it more computationally efficient. Additionally, there is no discrepancy between the pre-training and fine-tuning like in MLM. The ELECTRA architecture is similar to BERT, but it has one extra fully connected layer between the input embeddings and the first attention layer. This allows for smaller word embeddings than the hidden size, making it more efficient in terms of parameters while still performing just as well. Take ELECTRA Small, for instance, which employs a hidden size of 256 and an embedding size of 128. The fully connected layer, which enables the projection from embedding to hidden representations, requires an additional 128 x 256 parameters. However, it saves 30,000 x 128 parameters in the word embedding space.

1.2 Obtaining Sentence Embeddings from PLMs

There are various methods to extract sentence embeddings from pre-trained language models. The most straightforward approach is to average all word embeddings, which, in the case of PLMs, are the last hidden representations of words. To obtain the sentence embedding, we perform an element-wise $MEAN$ operation on all word embeddings. Alternatively, we can use the MAX operation to generate a sentence embedding that contains only the most significant features across all words. Another method is to use the last hidden representation of the *CLS* token, similar to the NSP objective. The process of converting word embeddings to sentence embeddings is known as pooling. Strategies such as *CLS*, *MEAN*, and MAX pooling have proven to be highly effective and continue to be widely used. In this study, we experiment with these pooling strategies and assess their effectiveness in STS tasks.

If one prefers not to train an encoder model from scratch, the Hugging Face Model Hub [20] offers an opportunity to utilize existing public PLMs. This open-source platform provides a wide range of pre-trained models for natural language processing tasks, streamlining research and application development. Numerous pre-trained English models can be found on the platform, such as *bert-base-uncased* [5] and *xlm-roberta-base* [21], the latter of which was trained on up to 100 languages.

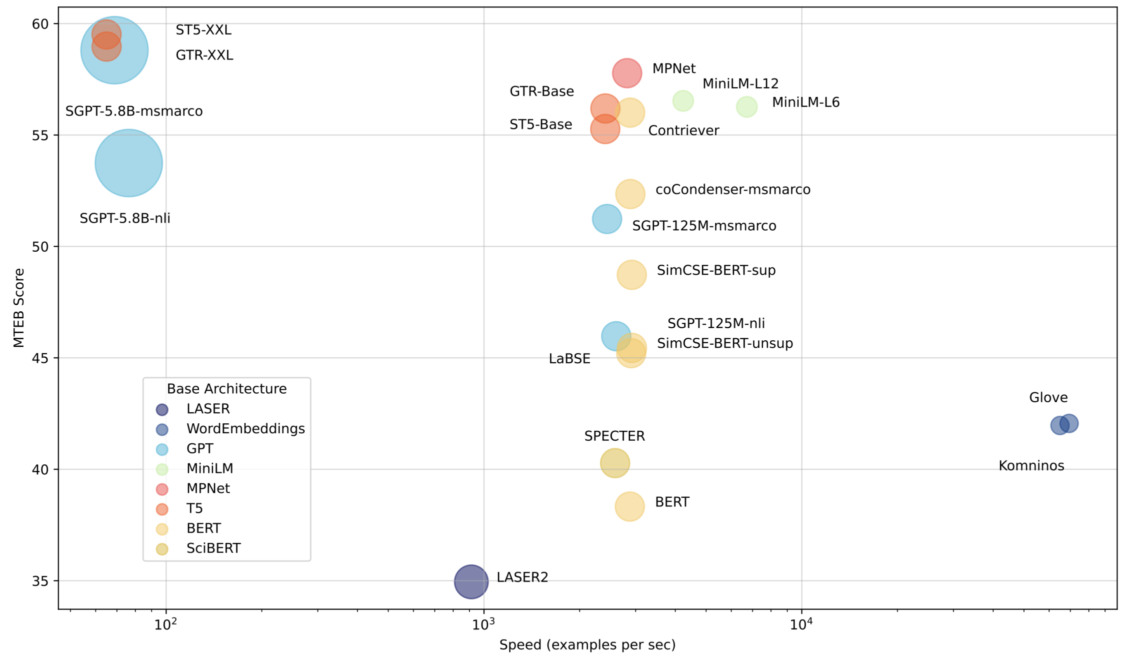
While these PLMs generate high-quality sentence embeddings, there is room for improvement. Several techniques can be employed to further enhance the quality of the embeddings. We will describe these techniques in detail in Section 2.

1.3 State-of-the-Art Sentence Embeddings

In this section, we examine state-of-the-art English models for generating high-quality sentence embeddings, the methods employed for their training, and their comparative performance. We also explore the popular open-source framework Sentence Transformers [22], which is utilized in most of the experiments discussed later in this work.

First, let us consider the Sentence Transformers framework. This framework offers a straightforward interface for generating state-of-the-art sentence, text, and image embeddings. It includes several pre-trained transformer models and supports fine-tuning on custom datasets. The performance of these models' sentence embeddings and semantic search is evaluated across 20 diverse datasets (see [22]). The top-performing model for these tasks is *all-mpnet-base-v2* [23]. As its name implies, it is a base-sized model fine-tuned from the *microsoft/mpnet-base* model [24]. The model was fine-tuned using a contrastive objective: given one sentence from a pair, the model should predict which of a set of randomly sampled sentences was actually paired with it in the dataset. Formally, the model computes the cosine similarity for each possible sentence pair in the batch and then applies cross-entropy loss by comparing it to the true pairs. Further details on contrastive learning will be discussed in the upcoming chapters. The model was fine-tuned using concatenated data from multiple datasets, with a total of over 1 billion sentence pairs.

Another notable model is *all-MiniLM-L6-v2* [25], a small-sized model with just 6 layers and an embedding size of 384. It is five times faster than the *all-mpnet-base-v2* model while still



■ **Figure 1.2** Performance, speed, and size of produced embeddings (size of the circles) of different embedding models. Embedding sizes range from 1.2 kB (Glove / Komninos) to 16.4 kB (SGPT-5.8B) per example. Speed was benchmarked on STS15 using 1x Nvidia A100 80GB with CUDA 11.6. [27]

providing high-quality embeddings. The process of creating this model was rather intricate. First, the BERT base model was distilled using a specialized technique [26]. The resulting model, *MiniLM-L12xH384*, has nearly identical performance to the original BERT base but with 3 times fewer parameters. Subsequently, *MiniLM-L6xH384* was created by retaining only every second layer from the previous model. Finally, *MiniLM-L6xH384* was fine-tuned using the same contrastive objective and data as *all-mpnet-base-v2*.

A recent paper, ”MTEB: Massive Text Embedding Benchmark” [27], provides another comparison of the best sentence embedding models. MTEB is a benchmark that covers 8 embedding tasks, spanning a total of 56 datasets and 112 languages. The model comparison includes those we have already discussed and others, some of which are considerably larger. For instance, the *SGPT-5.8B-msmarco* model is 50 times larger than the *microsoft/mpnet-base*. The model comparison is illustrated in Figure 1.2, which apart for model performance shows its relation to actual inference speed.

I would like to reiterate that these models were trained using vast amounts of supervised or self-supervised data, which is not available for the Czech language. Consequently, we cannot employ these techniques directly for Czech.

1.4 Czech Pre-trained Models

While there is an abundance of pre-trained English models, the availability of pre-trained Czech models is limited. Some of the publicly accessible Czech models include *RobeCzech* [28], *FERNET* [29], *Czert* [30], and *Small-e-czech* [31]. These models are trained using BERT, RoBERTa, and ELECTRA architectures without any additional enhancements specifically for sentence embeddings. Our aim is to improve these models' sentence embeddings, thereby enhancing their performance for the STS task. We utilize these models for reference, evaluation, and training purposes.

For training, our primary choice is the *Small-e-czech* model, due to its smaller size and reduced computational requirements. Employing a small-sized model enables more efficient research while still yielding meaningful results. With 10 times fewer parameters than other base-sized Czech PLMs, *Small-e-czech* is better suited for environments with limited computational resources.

An alternative to using Czech pre-trained models is to employ pre-trained multilingual models. Examples of such models include *BERT-Base, Multilingual Cased* [32], *LaBSE* [33], and *xlm-roberta-base* [21]. These models exhibit strong performance not only in English but also in other languages they were trained on (for example Czech). However, their performance across languages varies, depending on the distribution of languages in the training corpus. As a consequence, multilingual models may not achieve optimal performance for languages with lower representation in the training data, compared to models specifically trained on those languages. We do not fine-tune any multilingual models in this study. Instead, we compare them with our Czech models, specifically focusing on the *LaBSE* model, which has already been fine-tuned for STS tasks.

Chapter 2

Advanced Training Methods for STS Models

This chapter introduces bi-encoder and cross-encoder architectures, presenting state-of-the-art methods for training bi-encoders for STS tasks. It covers both supervised and unsupervised techniques and delves into knowledge distillation and transfer approaches between languages.

Neural networks are a powerful tool for solving a variety of tasks, including those involving sentence pairs. In this chapter, we present two approaches to tackle sentence pair problems, following notation of [34]: Cross-encoders and Bi-encoders. While the Cross-encoder approach is robust, it has limitations. The Bi-encoder method addresses these limitations.

2.1 Cross-encoders

Cross-encoder (CE) models are designed to **jointly** encode sentence pairs into a fixed-length representation, then use a method to generate the desired output, such as a similarity score in the case of STS. The task of determining the similarity between two input sequences using a CE can be expressed as $s(A, B) = M(A, B)$, where M represents a pre-trained encoder model. This model takes the concatenated input sequences A and B and produces a direct output of their similarity score. A notable example of a task solved as CE is the NSP task in BERT, which is a binary classification problem. CE technique involves passing a sentence pair, separated by a *SEP* token and preceded by a *CLS* token. The final *CLS* hidden representation is passed to a classification head, which consists of a single fully connected layer that predicts the most probable class. *MAX* or *MEAN* operation can be used in place of CLS to obtain the hidden representations. However, the CE model fails to generate sentence embeddings, limiting its practical application. The CE architecture is depicted in Figure 2.1.

2.2 Bi-encoders

An alternative approach to resolving STS is to utilize Bi-encoders (BE). BE consists of two encoders that **independently** transform input sentences A and B into fixed-length sentence embeddings, which are then used in a function to generate the desired output. In the context of STS, the degree of similarity can be defined as $s(A, B) = f_{\text{sim}}(eA, eB) = f_{\text{sim}}(M_1(A), M_2(B))$. Here, eA and eB refer to the sentence embeddings of input sentences A and B, respectively, which are produced by models M_1 and M_2. These models are frequently identical, and when

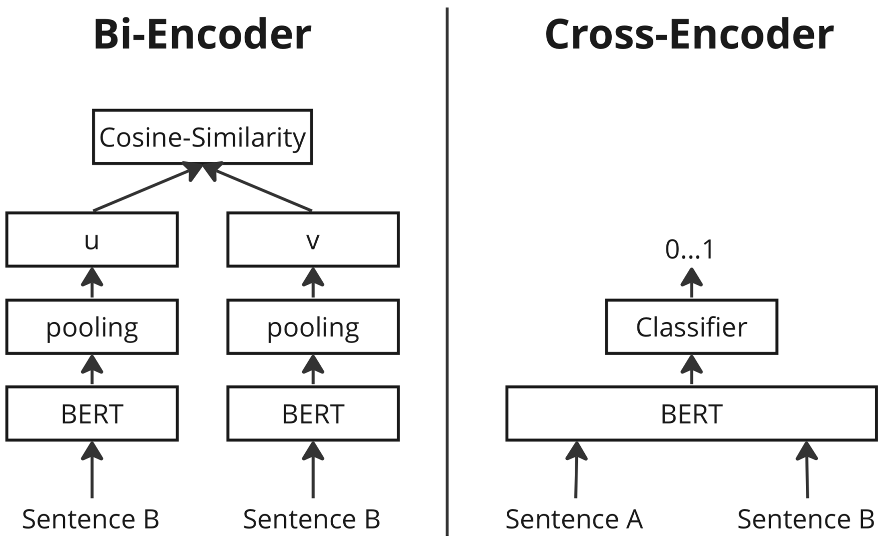


Figure 2.1 The difference between the cross-encoder and bi-encoder architecture, taken from [22].

this is the case, the BE model is typically referred to as a Siamese model. For the purposes of this project, we will solely utilize Siamese BE models, as they are generally easier to train and still have the potential to result in superior performance. Therefore, it is assumed that all BE models discussed from this point onwards are Siamese, unless otherwise stated explicitly. The similarity function f_{sim} is responsible for assessing the degree of similarity between the encoded sentence representations. Interestingly, the cosine similarity function is often sufficient for this purpose, however there are many other sophisticated methods, some of which we explore later. The BE architecture is depicted in Figure 2.1.

2.3 Which One to Use?

The choice between BE and CE models depends on the specific requirements of the given task. BE models are often simpler and faster to train, making them a good choice for tasks where efficiency is important. The primary advantage of using BE models, and the reason for their design, is their effectiveness in efficient semantic search. By representing each input sentence as a vector, the sentences can be encoded beforehand using BE. The similarity can then be computed by merely comparing their respective vectors. This comparison is typically very fast, allowing for scalable and efficient search for the most similar sentences. As a result, bi-encoder models are well-suited for tasks such as information retrieval, recommendation systems, and clustering.

Cross-encoder models, on the other hand, are not suitable for efficient semantic search. This is because, to find the most similar sentences, the entire model must be run for each sentence pair, making the process significantly slower than computing the similarity function directly, as done with BE models. However, CE models, which have access to both input sentences simultaneously, can capture more complex interactions between them than bi-encoder models. For tasks that require a deeper understanding of the relationship between the input sentences, cross-encoder models are often more effective and outperform BE models.

In this study, both approaches will be utilized and compared, but the primary focus will be on training the best-performing Bi-encoder model.

2.4 Train a Bi-encoder

This section reviews some of the latest techniques used to train optimal Bi-encoder models for STS and other related tasks, with a particular emphasis on unsupervised methods. The experimental section of this work offers in-depth testing and evaluation of every method discussed.

2.4.1 SBERT

The issue of CE and their infeasibility in tasks such as semantic search was initially examined in the publication "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" in 2019 [34]. This paper introduced various networks and objective functions to train the Sentence-BERT (SBERT) model, one of the earliest BE used in NLP. The choice of network and target function depends on the availability of training data. Our focus will be on two configurations: unsupervised and supervised. SBERT's performance was evaluated on common STS datasets[1] using the Spearman rank correlation metric. As the authors mention, the Pearson correlation is poorly suited for STS [35] (see Figure 2.2).

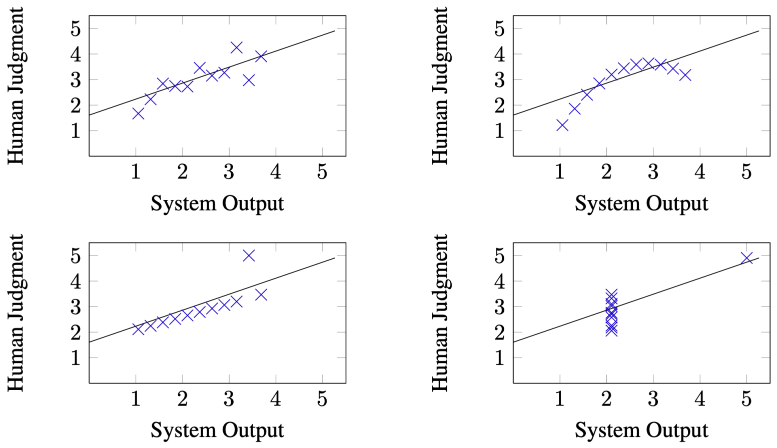


Figure 2.2 Despite all four systems having the same Pearson correlation coefficient of 0.816, they produce different outputs when compared to the gold standard derived from human judgment. The Pearson correlation coefficient is sensitive to non-linear relationships and outliers, and is not always a reliable measure of the quality of STS systems. The paper [35] emphasizes that humans would judge the quality of the four STS systems differently, even though they achieve the same Pearson correlation coefficient.

The unsupervised[2] SBERT is trained as follows. A pre-trained encoder model[3] is fine-tuned on the SNLI [36] dataset using the BE setup. SNLI is a collection of 570,000 sentence pairs labeled with contradiction, entailment, and neutral categories. By default, SBERT uses *MEAN* pooling to generate sentence embeddings, and for training purposes, it employs a 3-way classification objective function, defined as:

$$o = \text{softmax}(W_t \times (u, v, |u - v|))$$

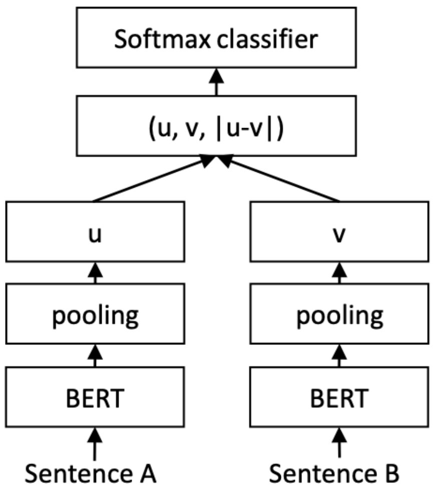
where the $(u, v, |u - v|)$ is the concatenation of the sentence embeddings for the input pair u, v, along with their element-wise absolute difference. This concatenation is then multiplied by

[1] These are not available in Czech.

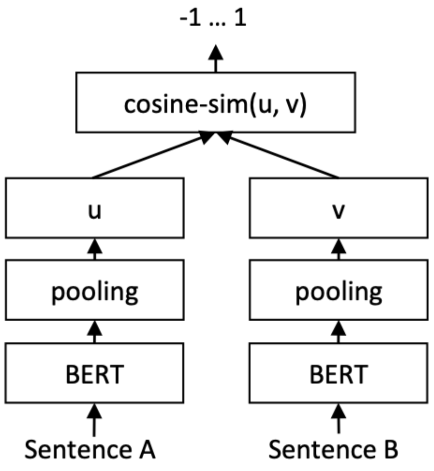
[2] By "unsupervised," we mean the absence of utilizing train data from the STS task. It cannot be considered entirely unsupervised, as it incorporates train data from a different yet similar task, namely natural language inference.

[3] Such as BERT, RoBERTa, ELECTRA, or others.

trainable weights $W_t \in R^{3n \times k}$, with n being the dimension of the sentence embeddings and k denoting the number of labels. The cross-entropy loss is optimized during training. The entire structure is depicted in Figure 2.3.



■ **Figure 2.3** SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two networks have tied weights (siamese network structure). [34].



■ **Figure 2.4** SBERT architecture at inference, which is used with the regression objective function. [34].

The supervised SBERT is trained similarly but on the STS training dataset using a regression objective function. The cosine similarity between the two sentence embeddings, u and v, is computed, and the mean squared error (MSE) is applied as the loss. This is depicted in Figure 2.4. The best performance was achieved by combining both approaches, i.e., pre-training SBERT on NLI using the classification objective and then training on STS using the regression objective. It is important to note that in both cases, the cosine similarity function was used to compare the similarity of the two sentence embeddings. Other best-performing methods use complex regression functions to map sentence embeddings to a similarity score, but because this function works pair-wise, they are not scalable if the collection of sentences reaches a certain size.

In unsupervised STS, BERT sentence embedding performs poorly, even worse than average GloVe embeddings. However, after fine-tuning with SBERT, it successfully outperforms even the best model on almost all STS datasets. In supervised STS, BERT used as a cross-encoder sets a new state-of-the-art performance by using a simple regression method for the output. This confirms our previous claim that CE is more powerful than BE. As with the supervised SBERT, we can see a significant boost in performance if we first pre-train the CE on the NLI dataset. For a full comparison of models, please refer to the SBERT paper [34].

Additionally, the authors evaluated the embeddings produced by SBERT in SentEval [37], a popular toolkit for evaluating the quality of sentence embeddings. These embeddings are used as features for a logistic regression classifier. Although the purpose of SBERT sentence embeddings is not to be used for transfer learning for other tasks, they still managed to achieve the best performance on 5 out of 7 SentEval tasks. This suggests that SBERT can effectively capture semantic information through sentence embeddings. However, as the training datasets are not available in Czech, it is necessary to explore unsupervised techniques that utilize raw sentences without any explicit relationships or labels further.

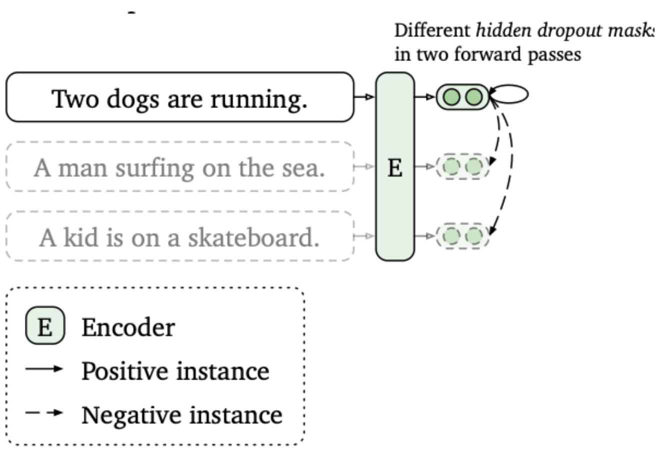


Figure 2.5 Unsupervised SimCSE [38] predicts the input sentence itself from in-batch negatives, with different hidden dropout masks applied.

2.4.2 Unsupervised SimCSE

In the paper "Simple Contrastive Learning of Sentence Embeddings" (SimCSE) [38], an unsupervised method for training STS models was proposed. This method takes an input sentence and predicts itself in a contrastive objective, with only standard dropout used as noise. Surprisingly, this simple method works quite well.

Contrastive learning is a technique used to learn representations by bringing together similar examples and pushing apart dissimilar ones. It assumes a set of paired examples (x, x^+), where the pairs are semantically related. The approach in their work uses a contrastive framework [39] with a cross-entropy objective and in-batch negatives [40] to train the model. The objective is calculated for each pair of examples in a mini-batch. The training objective for positive pair (x_i, x_i^+) with a mini-batch of N pairs is calculated as:

$$l_i = -\log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^+)/\tau}} \tag{2.1}$$

Here, h_i and h_i^+ are the encoded representations of x_i and x_i^+, sim() represents the cosine similarity, and τ is the temperature hyperparameter. The entire training process consists of encoding the sentences using a PLM and then fine-tuning all the parameters using the contrastive learning objective. The main idea of SimCSE is that it uses the same sentence for both x_i and x_i^+. Therefore, the only difference between h_i and h_i^+ is the influence of independently sampled dropout masks for x_i and x_i^+. They achieve this by feeding the same sentence to the model twice and obtaining a different result due to the dropout (see Figure 2.5). Other strategies for generating positive pairs were examined, but none of them performed as well as the one mentioned. The results are depicted in Table 2.1. Additionally, different training objectives were evaluated (such as randomly sampling one sentence from the two or three next sentences), and as shown in Table 2.2, the contrastive objective performed the best.

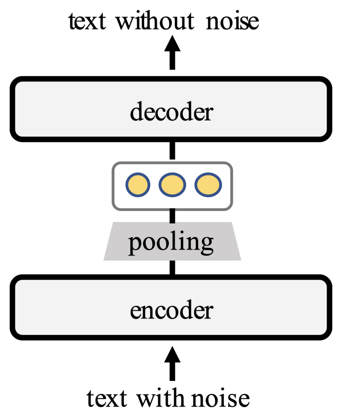
Apart for the unsupervised SimCSE, a supervised version of SimCSE was proposed, which uses STS training data and a slightly modified training objective. The supervised version even outperforms SBERT on STS tasks [38]. However, we will focus on unsupervised methods.

■ **Table 2.1** Comparison of data augmentations for SimCSE on STS-B [41] development set (Spearman's correlation). The results were taken from [38]

Data augmentation			STS-B
None (unsup. SimCSE)			**82.5**
Crop	*10%*	*20%*	*30%*
	77.8	71.4	63.6
Word deletion	*10%*	*20%*	*30%*
	75.9	72.2	68.2
Delete one word			75.9
w/o dropout			74.2
Synonym replacement			77.4
MLM 15%			62.2

■ **Table 2.2** Comparison of different unsupervised objectives (STS-B development set [41], Spearman's correlation). The two columns denote whether one encoder or two independent encoders are used. The results were taken from [38].

Training objective	f_θ	$(f_{\theta_1}, f_{\theta_2})$
Next sentence	67.1	68.9
Next 3 sentences	67.4	68.8
Delete one word	75.9	73.1
Unsupervised SimCSE	**82.5**	80.7



■ **Figure 2.6** Architecture of TSDAE. [42]

2.4.3 TSDAE

The TSDAE paper [42] introduces an unsupervised approach for learning sentence embeddings using a Transformer-based Sequential Denoising Auto-Encoder (TSDAE). The training process involves adding specific noise (e.g., deleting or swapping words) to input sentences, encoding them into sentence embeddings, and attempting to reconstruct the original sentences without the noise, using only sentence embeddings as a reference. This approach differs from the classical transformer encoder-decoder setup, where the decoder has access to all contextualized word embeddings. This modification creates a bottleneck that should force the encoder to produce high-quality sentence representations.

After experimenting with different configurations on the STS benchmark dataset, the authors discovered the best setup. They used word deletion with a ratio of 0.6 as input noise, the *CLS* token as the sentence representation, and tied the encoder and decoder parameters during training, which not only increased performance but also lowered memory requirements. The authors emphasized that performance on STS does not always correlate with performance on other downstream tasks. Other unsupervised approaches were often evaluated only on the STS dataset, and their performance on downstream tasks or domain-specific tasks remained unknown. To address this, they evaluated and compared TSDAE with other unsupervised methods on three different tasks (Information Retrieval, Re-Ranking, and Paraphrase Identification) for heterogeneous domains and various text styles. The results can be seen in Table 2.3. TSDAE works well as pretraining for specific tasks and exhibits superior STS capabilities compared to MLM. As noted in previous chapters, MLM pretraining performs poorly for STS tasks but is very effective in downstream tasks and domain adaptation. While SimCSE or CT [43] may outperform TSDAE on STS, they are often unsuitable for fine-tuning a specific task.

■ **Table 2.3** Performance (Spearman's rank correlation ×100) on the STS benchmark test [41] set and average performance on multiple specific tasks. We can see that some methods are better on STS, although Refer to [42] for details.

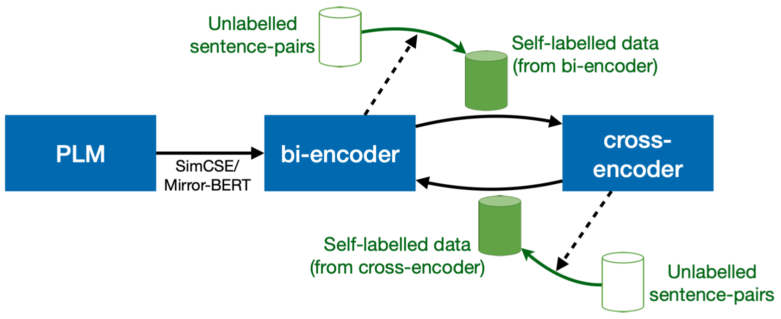
Method	**STSb**	**Specific Tasks**
Unsupervised method		
TSDAE	66.0	**55.2**
MLM	47.3	52.9
CT	73.9	52.4
SimCSE	73.8	50.6
BERT-flow	48.9	49.2
Out-of-the-box supervised pre-trained models		
SBERT-base-nli-v2	83.9	52.0
SBERT-base-nli-stsb-v2	**87.3**	52.3
USE-large	80.9	54.7

2.4.4 Knowledge Distillation

Knowledge distillation [44] is technique aimed at improving model performance by transferring knowledge from larger, more complex models (teacher models) to smaller, simpler ones (student models). The primary goal is to reduce computational costs while maintaining accuracy. The fundamental concept of knowledge distillation involves training a more powerful teacher model for a given task and then using its predictions to train a weaker student model. One method to train the student model is to have it learn to mimic the teacher model's behavior by minimizing the difference between their respective predictions.

In the context of training an STS BE, this approach can be applied by first training a superior CE and then using it as the teacher model for knowledge distillation. There are several techniques available for transferring knowledge from the CE to the BE. The most straightforward method teaches the bi-encoder to predict the cross-encoder's output using simple regression and a MSE objective (used in [31]). However, it is worth noting that training a robust cross-encoder in an unsupervised manner can be challenging.

2.4.5 Trans-Encoder


Unlabelled sentence-pairs
Self-labelled data (from bi-encoder)
PLM
SimCSE/ Mirror-BERT
bi-encoder
cross-encoder
Self-labelled data (from cross-encoder)
Unlabelled sentence-pairs

■ **Figure 2.7** A graphical illustration of the self-distillation learning scheme in TRANS-ENCODER. [45]

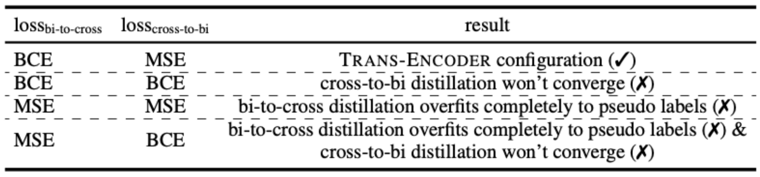
A more sophisticated method for training unsupervised STS Bi-encoders through knowledge distillation was introduced in the paper ”Trans-Encoder: Unsupervised sentence-pair modelling through self- and mutual-distillations” [45]. The training process involves transforming a PLM into a robust bi-encoder, which serves as an initialization point. This bi-encoder generates pseudo-labels for given unlabeled sentence pairs. A CE is then trained on these sentences, capturing deeper and more complex interactions while iteratively re-labeling the sentences with more accurate labels. Finally, the re-labeled sentences are used to train a more powerful bi-encoder model than the initial one. This process is called self-distillation because the BE labels the sentence pairs for itself with its CE form. The entire training process is illustrated in Figure 2.7. Notably, the final distilled BE can be used as the initial BE model in another iteration, making the entire process iterative.

In terms of training specifics, a BERT model pre-trained with the SimCSE objective on the target corpus (the STS train dataset) is used as the bi-encoder's initial model. Any technique that transforms BERT into an effective bi-encoder could be employed, but the authors choose SimCSE as the default method. The cross-encoder's learning objective in the next step is to minimize the KL-divergence [46] between its predictions and the self-labeled scores from the bi-encoder. This is equivalent to optimizing the (soft) binary cross-entropy (BCE) loss:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{n=1}^{N}(y_n \cdot \log(\sigma(x_n)) + (1-y_n)\cdot \log(1-\sigma(x_n))) \tag{2.2}$$

where N is the data-batch size; $\sigma(\cdot)$ is the sigmoid activation; x_n is the prediction of the cross-encoder; y_n is the self-labeled ground-truth score from the bi-encoder.

Interestingly, the trained CE often outperforms the initial model even though it learned from labels produced by itself. The authors explain that this is likely because the CE directly discovers the similarity between two sentences through its attention heads, finding more accurate cues to justify the relevance score, and is thus able to better generalize the problem. From a knowledge distillation perspective, we can view the BE and CE as the teacher and student, respectively. In this case, the student outperforms the teacher, not due to greater model capacity, but smarter task formulation. The objective for sequential BE training is MSE. The authors specifically use a different loss function than in the *CE* training because breaking the student-teacher discrepancy can harm generalization ability. They examine different combinations of training objectives, but find that the BCE-MSE method is the most effective, as shown in Figure 2.8. To further improve performance, the authors propose an enhancement called mutual distillation. The only difference is that they train not one but multiple *BEs* or *CEs* in each step and use averaged pseudo-labels for the next step. This approach achieves a performance boost in most experiments and makes training more stable, but it is more computationally intensive.



$\text{loss}_{\text{bi-to-cross}}$	$\text{loss}_{\text{cross-to-bi}}$	result
BCE	MSE	TRANS-ENCODER configuration (✓)
BCE	BCE	cross-to-bi distillation won't converge (✗)
MSE	MSE	bi-to-cross distillation overfits completely to pseudo labels (✗)
MSE	BCE	bi-to-cross distillation overfits completely to pseudo labels (✗) & cross-to-bi distillation won't converge (✗)

Figure 2.8 Comparison of different loss function configurations in TRANS-ENCODER. [45]

The main issue with the Trans-Encoder method is that it relies on unlabeled training pairs from the target domain, i.e., the STS training dataset sentence pairs. Later in this work, it will be demonstrated that this technique does not perform well with random sentence pairs sampled from the target domain.

2.4.6 Multilingual distillation

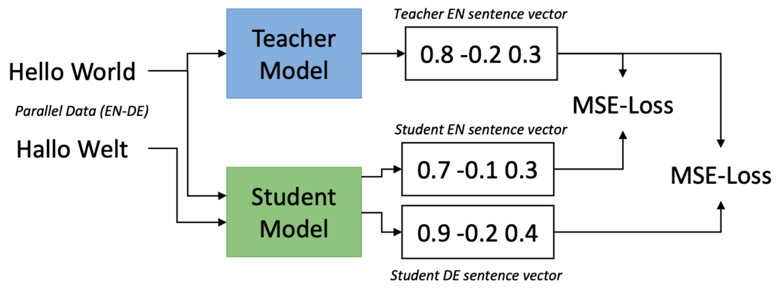


Figure 2.9 Given parallel data (e.g. English and German), train the student model such that the produced vectors for the English and German sentences are close to the teacher English sentence vector. [47]

Multilingual distillation is a technique that enables the extension of existing sentence embedding models to new languages. While multilingual models like LaBSE [33] can directly handle multiple languages, including Czech, their performance is often suboptimal compared to models specifically trained for Czech language. This work demonstrates that the multilingual distillation method outperforms LaBSE in the specific language. The method, proposed in the paper ”Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation” [47], leverages the expertise of a strong teacher model trained in one language to train a student model in the target language.

The process begins by selecting a teacher model that has been appropriately trained in a source language (e.g., English) and initializing a student model to be trained on the teacher's output. A bilingual dataset containing sentences in both the source language and their translations in the target language (e.g., Czech) is required for this purpose. The teacher model encodes sentences in the source language, while the student model encodes sentences in both languages. The training objective is to minimize the output loss, calculated as follows:

$$\frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}}[(M(s_j)-\hat{M}(s_j))^2+(M(s_j)-\hat{M}(t_j))^2] \tag{2.3}$$

Where the M and $\hat{M}$ is the teacher and the student respectively. The s_j is the English sentence and t_j the Czech one, both from mini-batch $\mathcal{B}$.

During training, the student model attempts to mimic the teacher model's behavior in both languages simultaneously. As a result, the student model should produce embeddings for the target language sentences similar to the embeddings produced by the teacher model for the source language translations. Upon completion of the training, the student model becomes bilingual, capable of generating high-quality embeddings for both the source and target languages. The entire training procedure is depicted in Figure 2.7.

The main advantage of this method is that it allows for the transfer of knowledge from high-performing models to other languages, effectively creating multilingual models capable of producing high-quality sentence embeddings. The primary limitation, however, is the need for a sufficiently large and diverse bilingual dataset, which may not be readily available for all language pairs. It is also important to note that the performance of the student model may be influenced by the quality of the translations in the bilingual dataset and the choice of teacher model.

Chapter 3

Data and metrics

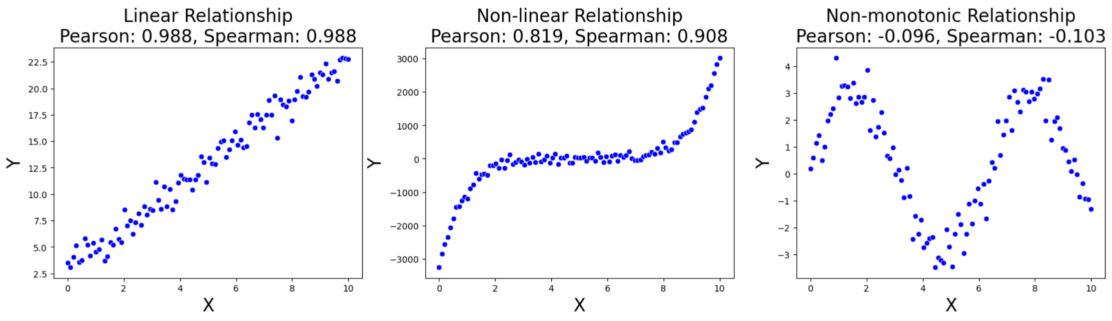
In this chapter, we introduce the train datasets for our STS models and present the test data for model evaluation. We provide a brief overview of the evaluation tasks and metrics used for assessing model performance on both STS and downstream tasks.

To ensure proper evaluation of STS models, we use appropriate test datasets for accurate performance comparison. Following the approach in [42], we assess models on both STS datasets and downstream tasks. We conduct two types of evaluations: intrinsic evaluation using two STS test datasets and extrinsic evaluation using two downstream test datasets. Thanks to previous research, we have access to labeled Czech sentence pairs suitable for both evaluation types.

3.1 STS Test Dataset

We incorporate data from [48] and [49]. The former is a large dataset containing 138,556 human-annotated sentence pairs from Czech journalistic sources. The sentence pairs were chosen by the annotators who were instructed to select three sentences (most similar, least similar, and something in between) from reports for each summary sentence, with the goal of creating a balanced dataset. The dataset includes a test set of 1100 sentence pairs labeled by an average of 9 annotators, ensuring high reliability. We use this test set for STS evaluation and refer to it as the *CNA* dataset. The latter consists of 1425 sentence pairs, translated into Czech from the English STS SentEval dataset [37] and manually annotated. These sentences fall into two categories: image descriptions (*SVOB-IMG* dataset) and headlines (*SVOB-HL* dataset). Table 3.1 shows examples from the *CNA*, *SVOB-IMG*, and *SVOB-HL* datasets.

For evaluation, we use the Spearman's correlation metric averaged across all three STS tests. As discussed in Section 2.4.1, this metric is better suited than Pearson correlation for the STS task. Pearson correlation (r) measures the linear relationship between two continuous variables, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation). A value of 0 indicates no correlation. Spearman correlation (ρ) assesses the strength and direction of the monotonic relationship between two continuous or ordinal variables, also ranging from -1 (perfect negative monotonic correlation) to 1 (perfect positive monotonic correlation). Unlike Pearson correlation, Spearman correlation is robust to non-linearities and outliers, as it is based on the ranks of the data values. See Figure 3.1 for a visual comparison of the two correlation metrics.



■ **Figure 3.1** The scatter plot displays both Pearson and Spearman correlation coefficients. The Spearman coefficient is better suited for measuring non-linear monotonic relationships.

■ **Table 3.1** Samples from the *CNA*, *SVOB-HL* and *SVOB-IMG* datasets [48, 49]. The similarity scores are from range $\langle 0, 5 \rangle$.

Dataset	**Sentence 1**	**Sentence 2**	**Similarity**
CNA	S Klausem se Van Rompuy setká dnes odpoledne.	Van Rompuy, který dnes návštěvou Prahy uzavřel dvouměsíční cestu po evropských metropolích, se rovněž setkal s prezidentem Václavem Klausem.	2.7
CNA	Ruské diplomaty vypoví mimo jiné i Spojené státy.	Proti ruským diplomatům vystoupily i Spojené státy, které jich vyhostily 60.	3.5
SVOB-HL	Muž byl obviněn poté co byl MP zraněn při útoku	Muž obviněn z vraždy v bytě	1.4
SVOB-HL	Ehud Olmert, bývalý izraelský premiér, byl odsouzen k šesti letům vězení za korupci	Ehud Olmert byl odsouzen k šesti letům v Izraeli	3.8
SVOB-IMG	Muž v černém obleku surfuje sám po ničivých vlnách.	Surfař v černém potápěčském obleku jede na bílé vlně v oceánu.	4.3
SVOB-IMG	Muž na bruslích sjíždí řadu schodů.	Muž na lyžích sjíždí zasněžený kopec.	0.8

3.2 Downstream Test Datasets

Following [42], we investigate STS model behavior on other NLP tasks, namely sentiment analysis and relevance ranking, in addition to STS. We select these tasks due to the availability of Czech language datasets for each.

3.2.1 Sentiment Analysis

Sentiment analysis classifies the sentiment expressed in text data (reviews, social media posts, customer feedback) as positive, negative, or neutral (see Figure 3.2). We use the Czech Facebook dataset (*CFD*) [50] for evaluation, containing 2,587 positive, 5,174 neutral, and 1,991 negative posts. We follow the approach used in [28] and employ *10-fold cross-validation* for evaluation, measuring performance using *macro-averaged F1* scores to report both mean and standard deviation across the folds.

Macro-averaged F1 score, a performance metric for multi-class classification tasks like senti-

Table 3.2 Samples from the Czech Facebook dataset (*CFD*) [50].

Comment	Label
vůbec o tom nepochybuji...budu přímo nadšená!!!!..už se těšííííím	positive
spíš by mě zajímalo kolik lidí už stojí ve frontě, nemá někdo informace z předních linií?	neutral
Jestli to není podvod...mě ještě nic nepřišlo a platba je zaplacena...	negative

ment analysis, computes the *F1* score for each class individually and then calculates the average across all classes, equally weighting each class. The F1 score is the harmonic mean of *precision* and *recall*, offering a balanced measure of a model's performance[1].

Precision quantifies the proportion of true positive predictions over the total number of positive predictions made by the classifier, measuring how many positively labeled instances identified by the classifier are actually true positives. *Recall*, on the other hand, quantifies the proportion of true positive predictions over the total number of actual positive instances in the data, measuring how many actual positive instances the classifier correctly identifies. The F1 metric is then calculated as:

$$F1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

where *precision* and *recall* are calculated as:

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.$$

3.2.2 Relevance Ranking

Ranking sorts documents by relevance to a query, a fundamental problem in Information Retrieval with applications in search engines and recommender systems. We use the DaReCzech dataset [31] containing 1.6 million Czech user query-document pairs with manually assigned relevance levels (Figure 3.3). The dataset is divided into three sets: training, development, and testing. Each document in the dataset consists of the concatenation of its title, URL, and Body Text Extract (BTE). The BTE is generated by filtering the HTML body of a webpage using an internal model of the Seznam.cz search engine, meaning it excludes headers, menus, and other content without any information. We replicate their rankings for comparison, using the *Precision at 10 (P@10)* metric on test dataset.

P@10 measures the proportion of relevant documents among the top 10 retrieved documents for a query. It is commonly used in information retrieval to assess the effectiveness of ranking algorithms. For example, consider a ranking system that retrieves the following 10 top-ranking documents for a query, where "R" denotes a relevant document and "N" denotes a non-relevant document:

$$R, R, N, R, N, R, R, N, R, N$$

[1] A good classifier should have both high *precision* and high *recall*.

There are 6 relevant documents (R) among the top 10. Thus, the P@10 for this query would be 0.6 (i.e. 6/10). This metric provides a simple way to evaluate the performance of ranking systems by focusing on the top results, which are typically the most important for users.

■ **Table 3.3** Examples from the *DaReCzech* test dataset [31]. The documents shown are created by concatenating the title, URL, and BTE, and are shortened to 128 tokens to reflect what the model can observe during training.

Query	**Document**	**Label**
karamel recept	title: karamel recepty url: labuznik.cz/ingredience/karamel bte: Recepty - karamel Karamel je lepkavá, tahavá hmota zlatohnědé barvy, kterou vyrábíme z cukru neustálým mícháním, nejlépe metlou, na středním stupni teploty. Hustotu poté upravujeme horkou vodou a tukem, který musí být pokojové teploty. Používáme ho v gastronomii do teplé i studené kuchyně. Díky...	1.00
kdy se otevřou obchody	title: ve čtvrtek se otevřou obchody služby končí zákaz nočního vycházení url: kurzy.cz/zpravy/568356 ve ctvrtek se otevrou obchody sluzby konci zakaz nocniho vychazeni bte: Ve čtvrtek se otevřou obchody, služby, končí zákaz nočního vycházení Od čtvrtka 3. prosince 2020 se přesune Česká republika z režimu 4 do režimu 3 protiepidemického...	0.00

3.3 Training datasets

We primarily employ pre-trained models for STS, fine-tuning them in an unsupervised manner. For this purpose, we leverage a portion of a large non-public corpus acquired by Seznam.cz, called the *Czech corpus*. It contains post-processed texts from Czech webpages of varying quality. These diverse, cleaned sentences lack a specific domain, making them suitable for testing unsupervised methods.

For our experiments with multilingual distillation from English to Czech, we need a high-quality bilingual dataset to train the model. We choose the *czeng20-csmono* dataset within Czeng 2.0 [51], which contains 50 million bilingual sentence pairs. Additionally, we experiment with the *czeng20-enmono* dataset, comprising 70 million English sentences with corresponding Czech synthetic translations.

To further optimize the STS model, we utilize training data from the *CNA* dataset.

Chapter 4

An Experimental Study of STS

In this chapter, we discuss the experimental work related to training STS models. The methods used, implementation details, hyperparameters, and practical considerations are covered. We provide a detailed procedure for training the most optimal STS model using these techniques. It is important to note that the primary results are not presented in this chapter but are reserved for the next chapter.

We initiate the training of STS models one by one, evaluating the results to determine the subsequent steps. By the end of this chapter, we aim to have trained a few of STS models for evaluation and discussion. The process begins with unsupervised training using SimCSE. Before diving into the training, we examine the general implementation details.

4.1 Implementation

Experiments were conducted using Python and various machine learning libraries, including PyTorch, Hugging Face Transformers, and Sentence Transformers [52, 20, 34]. PyTorch is an open-source machine learning library that efficiently handles tensor computing with strong GPU acceleration. Hugging Face Transformers is a Python library for state-of-the-art natural language processing (NLP), offering easy-to-use interfaces to pre-trained language models and facilitating fine-tuning on custom datasets. Sentence Transformers is a Python library that provides pre-trained models for generating high-quality embedded sentences and offers several methods for training bi-encoders or cross-encoders. The FAISS (Facebook AI Similarity Search) [53] library was also used for efficient similarity search and clustering of high-dimensional vectors.

All models were trained on a single NVIDIA RTX A4000 GPU with 16 GB RAM capacity.

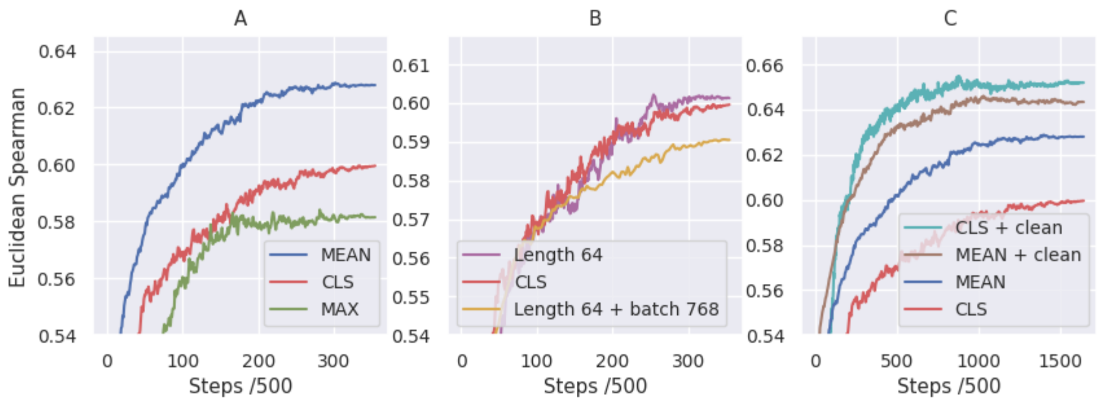
4.2 SimCSE

For training SimCSE, the Sentence Transformers library was utilized. Different configurations of hyperparameters were tested, such as pooling function, learning rate, batch size, objective function, and sequence length. However, due to computational demands, a comprehensive grid search was not performed. Instead, an incremental tuning strategy [54] was employed to find optimal hyperparameters. Initially, the *small-e-czech* model was trained with *CLS* pooling and a multiple negative ranking loss (MNRL) objective on a portion of the Czech corpus sentences. MNRL is a contrastive objective used in SimCSE, often referred to as the InfoNCE loss [55].

In the first set of experiments, only one hyperparameter was changed at a time, with others remaining constant. Results indicated that *MEAN* pooling outperformed other pooling functions, as shown in Figure 4.1 A. Adjusting hyperparameters like sequence length, batch size, or learning rate did not yield significant improvements, as depicted in 4.1 B.

Another set of experiments involved cleaning the sentences and changing the training objective. The sentences were preprocessed by filtering out those that were too short (< 5 words) or too long (> 30 words), as well as those containing excessive symbol characters. Additionally, the sentences were split using the Python NLTK library [56]. Using cleaned sentences significantly benefited SimCSE. Interestingly, *CLS* pooling performed better than *MEAN* pooling when using cleaned sentences, as illustrated in Figure 4.1 C. We experimented with alternative objective functions and similarity measures, including *MegaBatchMarginLoss*, *OnlineContrastiveLoss*, and *MultipleNegativesSymmetricRankingLoss* as objectives[1], and *Euclidean distance* as the similarity function. However, none of these alternatives resulted in significant improvements.

In the final experiment, a learning rate of 2e-5, a batch size of 256, *CLS* pooling, the AdamW optimizer [57], a sequence length of 128, and pre-processed sentences were used. The MNRL and cosine similarity remained unchanged.



■ **Figure 4.1** The performance of various experiments was evaluated on *CNA*, *SVOB-IMG* and *SVOB-HL* datasets, and their average performance was measured using Spearman's correlation coefficient and the Euclidean similarity function: the similarity score was obtained by calculating the Euclidean distance between two sentence embeddings, and the metric was calculated as Spearman's correlation with true labels. Plot A compares the pooling functions. Plot B compares different batch sizes and sequence lengths, where "CLS" have sequence length of 128 and batch size 256. Plot C shows the influence of the sentence preprocessing on the SimCSE training. The training with preprocessed sentences are marked as "clean".

4.3 TSDAE

The Sentence Transformers library, which contains the original TSDAE implementation, was used in this experiment. The *small-e-czech* was used as an initial PLM. The focus was on optimizing the learning rate and pooling function during hyperparameter search. A maximum sequence length of 512 tokens and a batch size of 16 were set as the default configuration. Due to the memory-intensive nature of TSDAE, this batch size was the largest possible for a 16 GB RAM system with a sequence length of 512. The model was trained using the AdamW optimizer with a learning rate of 5e-5. In comparison to SimCSE, TSDAE training proceeded at a much slower pace, which restricted the exploration of various configurations. Nevertheless, limited experimentation revealed that the *CLS* pooling method was vastly superior to other pooling techniques, and adjusting the learning rate did not yield further improvements in performance.

[1]All of these can be found in Sentence Transformer Library [22].

Experiments were conducted on the ELECTRA small model without pre-training (i.e., randomly initialized). The results showed that using TSDAE as pre-training led to poor performance during subsequent fine-tuning on downstream tasks. In fact, the performance was worse than that of a randomly initialized ELECTRA model without pre-training. We speculate that the trained embeddings from a separate TSDAE are too complex to be used as features for general NLP tasks. While it is sufficient for the TSDAE objective, it requires a complex decoder (the model itself, since the weights are tied) to decode these features. However, this is not the case in usual fine-tuning scenarios, where we only have a simple classifier or regressor at hand. This problem addresses recent auto-encoders models with shallow decoder, for example SimLM [58] or RetroMAE [59].

4.4 Multilingual distillation

The objective of this experiment was to distill an English model into a Czech model using Multilingual Distillation. The *mpnet* [2] model served as the teacher, while a newly initialized BERT small model acted as the student. Since the student model is multilingual and should be capable of encoding both English and Czech sentences, the Czech tokenizer was unsuitable. To address this issue, we employ the union of English BERT and *small-e-czech* vocabularies.

To tackle the discrepancy in embedding size between mpnet (768) and BERT small (256), a Principal Component Analysis (PCA) projection was implemented to reduce *mpnet's* embedding size while maintaining high quality. PCA was trained on the NLI dataset corpus. An alternative approach was tested to train a student with a different embedding size. Specifically, a linear projection without an activation function was appended to the smaller student model, increasing its embedding size to 768, and then training was initiated. After training, the projection was removed, resulting in a trained student model with the desired 256 embedding size. The PCA and linear projection are depicted in Figure 4.2. While the FFN approach performed slightly better on downstream tasks, it was somewhat weaker on the STS task. Therefore, we ultimately opted for the PCA approach as our final setup.

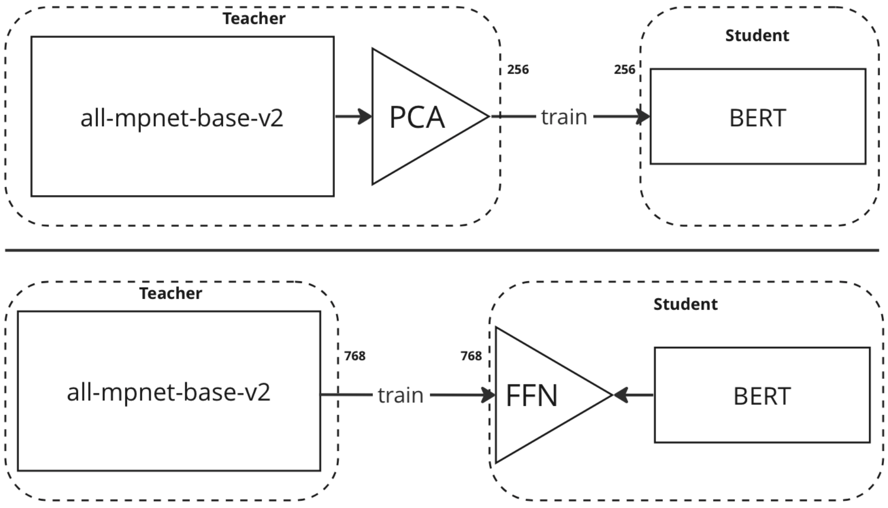
A bilingual dataset was necessary for training the model. Various datasets, including *czeng20-csmono* and *czeng20-enmono* were experimented with, both independently and in combination. We found that the quality of translations significantly affected the effectiveness of the training process. Additionally, the synthetic Czech language in the second dataset was unsuitable for training. The best results were achieved using only the *czeng20-csmono* dataset.

During the hyperparameter search, a smaller batch size surprisingly worked well. For the final experiment, a batch size of 32, a larger learning rate of 1e4, a cosine scheduler, and a warmup of 10% of training steps were used. The model was trained for five epochs on the *czeng20-csmono* dataset. The resulting model is called **czen-bert-small**, as it is a bilingual BERT model. Although attempts were made to follow up with SimCSE and TSDAE training, we discovered that these techniques did not provide any additional performance improvement for an already robust BE; in fact, they could diminish it. While SimCSE with *czen-bert-small* occasionally showed slight improvements in STS performance, it consistently undermined performance in downstream tasks, making it an unworthy trade-off. TSDAE was even more detrimental, negatively affecting both STS and downstream performance.

4.5 Trans-encoder

We implemented the iterative training process in the Trans-Encoder using the authors' source code [45]. Initially, we found that using a new model (PLM) during each iteration was more effective than initializing the CE with a BE, and vice versa. This indicated that the BE might

[2]Specifically, the *all-mpnet-base-v2* [23] model.



■ **Figure 4.2** The difference in multilingual distillation using PCA and one layer of Feed-Forward Neural network (FFN). The PCA trains students on embedding size of 256, meanwhile the FFN trains the student on embedding size of 768.

not be a suitable match for CE training. As a result, the *small-e-czech* model could be initialized as the CE for each iteration, however to further leverage knowledge distillation, *RobeCzech* was chosen as the CE. This approach distilled a stronger CE to BE and a larger model to a smaller one.

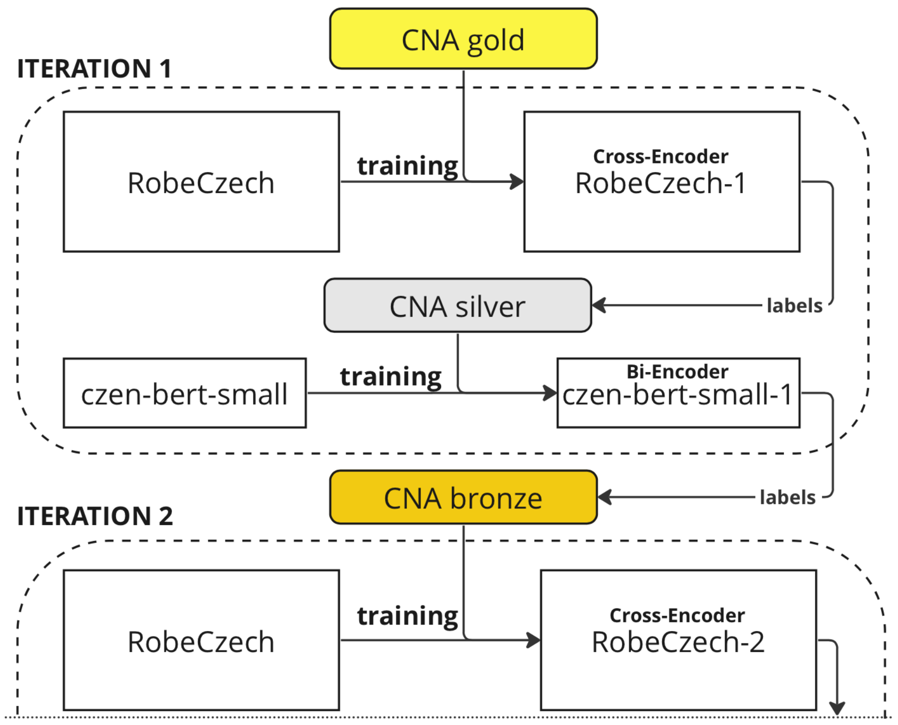
The CNA train dataset was first used to train a robust RobeCzech CE, referred to as *RobeCzech-1*. After the Trans-encoder process, a strong BE, *czen-bert-small*, was trained on the *CNA* train sentences' silver labels produced by *RobeCzech-1*. This led to a significant performance improvement on the CNA test dataset, showcasing the power of knowledge distillation. The trained *czen-bert-small* using labels produced by *RobeCzech-1* is called *czen-bert-small-1*. This entire process was the first iteration of the Trans-Encoder as depicted in Figure 4.3.

The distillation of *czen-bert-small-1* into a newly initialized *RobeCzech* CE resulted in an improved *RobeCzech-2*, further proving the effectiveness of the Trans-encoder concept. The *czen-bert-small-2* was then trained as the final Trans-encoder BE model. Additional iterations did not yield further improvements.

Although the resulting *czen-bert-small-2* model performed well on the *CNA* test dataset, its performance on the *SVOB-IMG* and the *SVOB-HL* declined. To address this, more unlabeled sentence pairs were needed to improve the model's generalization capabilities. The first approach, randomly selecting two sentences from the Czech corpus, did not produce significant results. It was hypothesized that these sentence pairs were unlikely to be semantically similar, leading to an excess of low probability labels.

An alternative approach involved paraphrase mining. Sentences were shortened and cleaned, reducing symbols and extraneous information. Sentence embeddings were calculated and stored in a FAISS index. From this index, 50 closest sentences were randomly sampled for each anchor sentence. Five pairs were then randomly selected from these, forming a new dataset with approximately one million unlabeled sentence pairs. This dataset was labeled with *czen-bert-base* and combined with the *CNA* train pairs, which retained their original labels. The resulting model, *paraphrase-czen-bert-small*, maintained the same performance level as *czen-bert-small-2* while also enhancing its performance on the other datasets.

The Trans-Encoder pipeline is highly complex, and it was not possible to execute all possible experiments. Further research is necessary, particularly concerning the strategy for creating unlabeled pairs for Trans-Encoder training. However, we believe that this technique holds great potential.



■ **Figure 4.3** Visualization of the first iteration of the Trans-encoder pipeline. The *RobeCzech* model is trained as a CE on CNA training dataset [50] gold labels (*CNA-gold*). The model is then used to re-label the CNA dataset, resulting in *CNA-silver*, which is used to train the *czen-bert-small* model as a BE. The trained BE *czen-bert-small-1* model is then used to re-label the CNA dataset again, resulting in *CNA-brone*, which is utilized in subsequent iterations.

Chapter 5

Evaluation

This chapter will cover the primary findings on STS and downstream tasks for STS model and other PLMs, and provide a detailed discussion of the results.

5.1 Evaluation of Existing Pre-trained Language Models

Table 5.1 Evaluation of the PLMs on the STS datasets [48, 49]. The final score is measured as average Speraman correlation $\times 100$ on all the datasets. *Random models are randomly initialized (with random weights, without additional training) ELECTRA models with corresponding size. First two models are small, other are *base sized*. *LaBSE*, as supervised model trained as BE, sets the upper bound for BEs. *Czert-B-sts-CNA (CE)* [48] is a CE model trained on CNA training dataset.

	Spearman			
Model	SVOB-IMG	SVOB-HL	CNA	**Average**
Random small model*	60.0	56.3	66.4	60.9
Small-e-czech	28.3	31.6	57.0	38.9
Random base model*	64.3	55.0	68.3	62.5
RobeCzech-base	67.9	58.2	75.7	67.2
Czert-B-base-cased	**77.4**	**64.3**	**84.3**	**75.3**
Czert-B-sts-CNA (BE)	71.5	58.5	73.1	67.7
LaBSE	85.1	77.3	85.7	82.7
Czert-B-sts-CNA (CE)	81.1	74.0	88.8	81.3

First, we analyze the performance of existing PLMs on STS (see Figure 5.1). The scores are calculated as the best Spearman rank correlation coefficient, considering all combinations of similarity and pooling functions[1]. This chart highlights the maximum potential of each model in the BE setup. The performance of a CE model trained on the CNA training dataset is also included in the chart for comparison. It's worth mentioning that most trained models perform the best with Euclidean or Cosine similarity and *CLS* pooling.

The chart indicates that *small-e-czech*, or ELECTRA pre-training, is not well-suited for STS, as it shows the lowest performance among all models. Surprisingly, even a randomly initialized ELECTRA model strongly outperforms *small-e-czech*. The random model, with no pre-training, performs exceptionally well on STS, setting a strong baseline.

[1]We used cosine similarity, Manhattan distance, Euclidean distance, and dot product as similarity functions, and employed *CLS*, *MEAN*, and *MAX* for pooling.

Before discussing other pre-trained models, it should be noted that the last six models have a *base* size with an embedding size of 768. The *RobeCzech-base* (*RobeCzech*) model, which uses RoBERTA pre-training, performs noticeably better than ELECTRA and the random model in base size, however the *Czert-B-base-cased* (*Czert*) model, a BERT-like model, outperforms other PLMs, making it the more suitable pre-training for STS. The *Czert-B-sts-CNA* model was trained on the *CNA* training dataset as a CE. Although the CE training improves its performance on STS when used as CE, it degrades the quality of sentence embedding.

Although RoBERTA and BERT show reasonable performance on STS, there is a significant difference in performance compared to LaBSE, which was directly trained for strong sentence embeddings, even though LaBSE is not specifically trained for the Czech language. This demonstrates the power of labeled or a huge amount of semi-supervised data.

We also assess the performance of PLMs on the *CFD* dataset (Table 5.2) using the evaluation methodology described in [28]. To obtain sentence embeddings, we apply the encoder with *CLS* pooling, and a softmax-activated classification layer generates the sentiment prediction. Our models are trained with AdamW, a batch size of 64, and a fixed learning rate of 5e-5, without performing a search for the optimal learning rate. Initially, we train for one epoch with frozen encoder weights, only updating the weights of the classification layer. Following this, we train for an additional three epochs without freezing the encoder weights, simultaneously updating both the encoder weights and classification layer weights. During training, we perform 10-fold cross-validation, reserving random 10% of the training data in each iteration to form a development set. We report the average performance and standard deviation across all folds. Table 5.2 demonstrates that the performance of the *base* models, which have more parameters, is indeed clearly superior to that of the *small* models. Pre-training greatly improves the results of subsequent fine-tuning, as evidenced by the significant differences between the randomly initialized models and the pre-trained ones. Although the LaBSE model outperforms other models on STS tasks, this advantage does not necessarily transfer to less similar tasks such as sentiment analysis, resulting in comparable results to other pre-trained language models.

■ **Table 5.2** The performance on the *CFD* [50] was evaluated using 10-fold macro F1, with the standard deviation measured on 10 folds (following [28]). The table is divided into smaller models at the top and base models at the bottom.

Model	**CFD**
Random small model	55.3 ±2.7
Small-e-czech	75.5 ±1.2
Random base model	70.1 ±3.6
Czert-B-base-cased	78.8 ±1.2
RobeCzech-base	**80.1** ±1.1
LaBSE	79.7 ±1.4

5.2 Evaluation of Unsupervised STS Bi-Encoders

We follow with an evaluation of models that underwent fully unsupervised fine-tuning using SimCSE and TSDAE. The results of our experiment demonstrate that integrating SimCSE with the *small-e-czech* model leads to a significant improvement in performance on the STS task, as depicted in Table 5.3. Surprisingly, TSDAE outperforms SimCSE on STS, which contradicts the findings on English reported in the TSDAE paper [42]. It's important to note, however, that we use smaller models in our study. We also trained SimCSE with *Czert* and *RobeCzech* for comparison, but the gains on STS are not as substantial as with the *small-e-czech* models. As before, all of these models fall short of the LaBSE model's performance. We further evaluate the models on the *CFD* and *DaReCzech* dataset, with the results shown in Table 5.4.

Table 5.3 The STS models are evaluated on three datasets, namely SVOB-IMG, SVOB-HL, and CNA [48, 49], with the average Spearman correlation ×100 used as the final score for each model.

	Spearman			
Model	SVOB-IMG	SVOB-HL	CNA	**Average**
Small-e-czech	28.3	31.6	57.0	38.9
SimCSE-small-e-czech	65.1	54.4	77.9	65.8
TSDAE-small-e-czech	**76.5**	**65.1**	**80.6**	**74.0**
Czert-B-base-cased	77.4	64.3	84.3	75.3
SimCSE-RobeCzech-base	80.7	68.1	85.9	78.2
SimCSE-Czert-B-base-cased	79.9	67.5	85.7	77.7
LaBSE	85.1	77.3	85.7	82.7

For evaluation on relevance ranking, we employ the code from the *DaReCzech* paper and follow their exact setup. However, we do not use knowledge distillation for model training (but include the results in Table 5.4). We only use the custom regression head, which combines cosine similarity, Euclidean distance, and one FFN. This entire regression head can be seen as an example of a more complex similarity function. We also apply the weighted *CLS* pooling, which takes the weighted average of *CLS* hidden representations on all layers, with the weights learned during training. We train the model on the *DaReCzech* training dataset and select the checkpoint based on the evaluation of their development dataset. We train the model using a learning rate of 5e-5 and a batch size of 64, and evaluate it on the test dataset.

We observe that although SimCSE and TSDAE significantly improve performance on STS, they negatively affect performance on tasks dissimilar to STS, i.e., those not using sentence pairs (*CFD)*. However, in terms of performance on information retrieval, specifically relevance ranking, this pre-training indeed helps with the performance. We even outperform the best model from the DaReCzech paper, which uses knowledge distillation from CE on this task.

Table 5.4 The performance on the *CFD* [50] and *DaReCzech* [31] dataset is measured as F1 and P@10 respectively (×100). [a]The *Siamese-Cosine* model is the best performing model from [31] without knowledge distillation. [b]The *Siamese-Cosine-distilled* is the best model from their paper. [c] The Oracle represents the highest attainable P@10 score for the DaReCzech dataset. This is due to the fact that not all queries have at least 10 relevant documents associated with them.

Model	**CFD**	**DaReCzech**
Random small model	55.3 ± 2.7	37.9
Small-e-czech	**75.5** ± 1.2	44.73
SimCSE-small-e-czech	73.9 ± 1.4	45.32
TSDAE-small-e-czech	74.6 ± 1.3	**45.52**
Siamese-Cosine[a]	-	44.72
Siamese-Cosine-distilled[b]	-	45.49
Oracle[c]	-	59.3

5.2.1 The Best STS Models

In the previous sections, all models utilized only the Czech corpus, indicating they were entirely unsupervised. The following models, however, leverage pre-trained supervised models, bilingual corpora, and even the *CNA* training dataset to achieve the best possible performance. Figure 5.5 presents the evaluation results. The table includes the *TSDAE-small-e-czech* model, which is the best-performing small model from previous section, as well as the LaBSE model, a multilingual supervised base model that serves as our upper bound.

Table 5.5 Evaluation of the best STS models on three datasets, namely SVOB-IMG, SVOB-HL, and CNA [48, 49]. The performance is measured as Spearman's correlation ×100.

	Spearman			
Model	SVOB-IMG	SVOB-HL	CNA	**Average**
TSDAE-small-e-czech	76.5	65.1	80.6	74.0
czen-bert-small	**91.7**	**86.9**	88.0	88.8
czen-bert-small-2	88.9	84.6	**91.9**	88.4
paraphrase-czen-bert-small	91.1	85.3	91.8	**89.4**
LaBSE	85.1	77.3	85.7	82.7

The *czen-bert-small* model, trained solely using multilingual distillation, exceeded our expectations in its effectiveness, even surpassing the performance of the base-sized *LaBSE* model. This shows that multilingual distillation can effectively transfer knowledge between languages. The *czen-bert-small-2* and *paraphrase-czen-bert-small* models were developed by combining the *czen-bert-small* model with Trans-Encoder, an advanced knowledge distillation technique. Although this did not lead to a significant performance improvement, any minor enhancements are valuable when the STS model is already strong. We assess the performance of these models on downstream tasks in Table 5.6. We observe that the *czen-bert-small* model not only learned powerful features for STS, but also for downstream tasks, outperforming all other models in both sentiment analysis and relevance ranking. The Trans-Encoder training did not further contribute to the performance on downstream tasks.

Table 5.6 This table presents the performance (F1 ×100 and P@10 ×100) of the top STS models on downstream tasks [50, 31]. *The *LaBSE-CE* model, which is the *LaBSE* model trained as a CE on DaReCzech, serves as the upper bound.

Model	**CFD**	**DaReCzech**
TSDAE-small-e-czech	74.6 ± 1.3	45.52
czen-bert-small	**79.2** ± 0.9	**46.15**
paraphrase-czen-bert-small	78.7 ± 1.2	46.10
Siamese-Cosine-distilled	-	45.49
LaBSE-CE*	-	46.97

5.3 The Impact of Training Data Size

DaReCzech is a vast collection of annotated query-document pairs for relevance ranking. However, for other similar tasks, we may not always have access to such a large labeled dataset. To address this, we conduct experiments using different amounts of labeled data, ranging from 1000 to all 1.4 million pairs, to fine-tune our model. We then observe the effect of our STS pre-training on this task with varying amounts of data. Table 5.1 presents the results of our experiments. The reported results are averaged over four runs with different random seeds for model initialization and data subset. They include the *P@10* metric and the standard deviation across all runs.

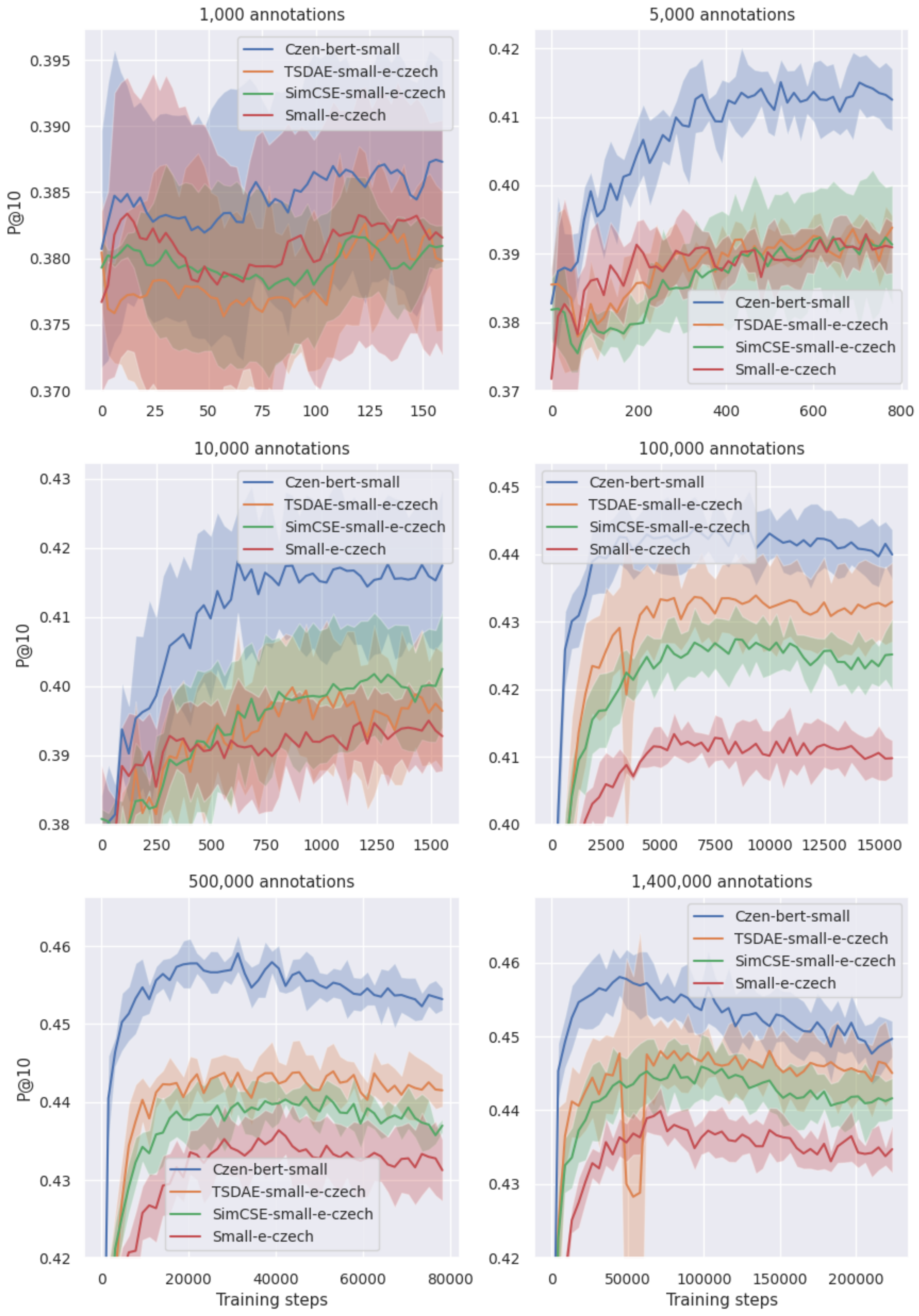
We observe that when using too few annotated examples (1000), all models perform similarly, unable to learn much from the limited data. However, when we increase the number of annotations to at least 5000, the *czen-bert-small* model demonstrates a significant performance boost. Although other methods such as SimCSE and TSDAE also show benefits, their impact is not clear until annotations reach a larger scale (100,000). The initial hypothesis behind this experiment is that pre-training with special methods would not improve performance with sufficient fine-tuning data, but could be advantageous with a smaller number of annotations. Our results confirm that this approach is indeed beneficial, both for a smaller number of annotations and for larger annotation datasets.

5.4 Exploring the Best STS Model

This section highlights specific examples from our top-performing STS model, *paraphrase-czen-bert-small*. Figure 5.2 demonstrates the notable relative improvement when transitioning from *small-e-czech* to *paraphrase-czen-bert-small*. The graph clearly indicates that ELECTRA pre-training is not effective for STS, whereas TSDAE can lead to substantial improvement.

Figure 5.7 presents the largest prediction errors made by the *paraphrase-czen-bert-small* model on our STS datasets. By examining these mistakes, we can identify areas for model improvement. The model typically overestimates the similarity score and never underestimates it.

In the final Table 5.8, we incorporate a large query dataset containing over 26 million real user queries sourced from Seznam.cz. An FAISS index is created for these queries, enabling the search for the most similar ones given specific query samples. The objective is to determine whether the STS model can capture the query semantics and accurately retrieve the most similar ones. Initially, the 30,000 most similar queries for a given query are identified and sorted. Then, queries with different order numbers (ranging from 1 to 30,000) are examined. This approach allows us to visualize even more distant examples.



■ **Figure 5.1** Evaluated performance of STS pre-trained models on the *DaReCzech* [31] relevance ranking task using different amounts of labeled data for fine-tuning, ranging from 1,000 to 1.4 million (the entire training dataset). The reported performance metrics represent the average *P@10* score of four experimental runs, along with the standard deviation across all runs. The y-axis displays the *P@10* metric, while the x-axis represents the number of training steps.

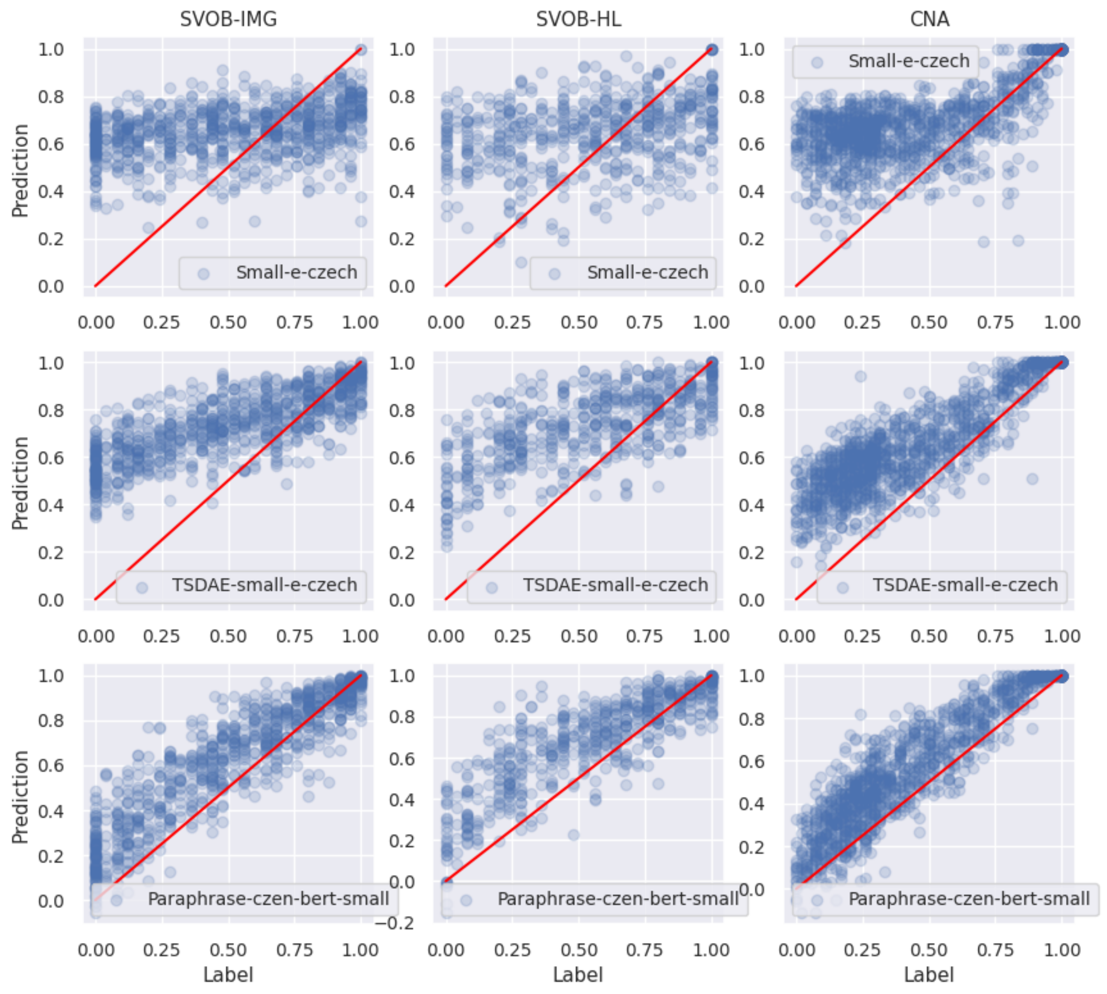


Figure 5.2 Correlation between labels and predictions on STS datasets [48, 49] for different models. The red line visualizes the ideal predictions.

Table 5.7 Biggest mistakes of the *Paraphrase-czen-bert-small* model on three STS datasets [48, 49].

Dataset	Sentence 1	Sentence 2	Label	Prediction
SVOB-IMG	• Dva černobílí psi si hrají spolu venku.	• Dvě děti a černý pes si hrají venku ve sněhu.	0.2	0.772
	• Dítě skáče do bazénu.	• Pes skákající do bazénu.	0.24	0.772
	• Dva psi hrající si ve sněhu.	• Dvě děti hrající si se psem.	0.24	0.764
	• Černobílé foto staré vlakové stanice.	• Černobílé foto motorky ležící na zemi.	0.04	0.563
	• Dva ptáci škádlící se v trávě.	• Dva psi si spolu venku hrají.	0.04	0.557
SVOB-HL	• Čínské akcie při pátečním poledni uzavřely výše	Čínské akcie v pátek otevřely níže	0.2	0.846
	• Tokio bude hostit olympijské hry 2020.	• Zápas přidán na olympijské hry v Tokiu 2020.	0.279	0.907
	• FAA pokračuje v zákazu amerických letů do Tel Avivu	• FAA zpřísní zákaz amerických letů do Tel Avivu	0.36	0.946
	• Útok NATO "zabíjí civilisty" v Afghánistánu	• Dva vojáci NATO byli zabiti v Afghánistánu	0.16	0.718
	• FAA pokračuje v zákazu US letů do Tel Avivu.	• FAA ruší zákaz U.S. letů do Tel Avivu.	0.36	0.899
CNA	• Průměrná mzda v podniku se pohybuje kolem 21.400 Kč, tedy téměř 6000 Kč pod celostátním průměrem.	• Průměrná mzda v podniku se pohybuje kolem 22.700 Kč a na konci roku má být zhruba 23.000 Kč měsíčně, tedy více než 4000 Kč pod celostátním průměrem.	0.240	0.817
	• "STAN ohlásil spolupráci s Daliborem Dědkem, ta nadále trvá.	• STAN zároveň oznámil odchod Dalibora Dědka z čela Ústecké kandidátky.	0.203	0.722
	• U obchodu s ropou a zemním plynem, jehož výsledek kladně ovlivnil bilanci zahraničního obchodu, se meziročně snížil schodek o 1,8 miliardy korun a u obchodu s chemickými látkami o 1,6 miliardy.	• O miliardu se pak prohloubil deficit bilance obchodování s koksem a rafinovanými ropnými produkty.	0.222	0.704
	• Přesné příčiny smrti inženýra má v Německu po převezení ostatků určit pitva.	• Německá strana je o tom přesvědčena, i když tělo oběti bylo podle neoficiálních zdrojů nalezeno s průstřelem v těle.	0.185	0.662
	• Právě Schapiro byl velvyslancem USA v době, kdy byl Nikulin zatčen.	• Zprávu o vydání Nikulina uvítal i bývalý americký velvyslanec v Praze Andrew Schapiro.	0.222	0.698

■ **Table 5.8** Examples of most similar queries for a given anchor query retrieved using the *Parahprase-czen-bert-base* model. The rank represents the query position in a sorted dataset based on the similarity score, as labeled by the model. This non-public dataset of queries consists of 26 million user queries obtained by Seznam.cz.

Anchor Query	**Most Similar Query**	**Rank**	**Similarity score**
Co dělat v Brně o víkendu?	co o víkendu v brně	1	0.936
	tipy na víkend brno	5	0.881
	kam o víkendu v brně	10	0.868
	na co do města brna	50	0.821
	kam o víkendu s detmi domů brno a okolí	100	0.793
	rozhoz brno	1 000	0.717
	Víkend v praze	2 500	0.690
	Hrazdíra Brno	10 000	0.654
	masopust v brně 2023	20 000	0.633
	Brno dolní nadrazi	30 000	0.619
Jaký je nejlepší způsob, jak odstranit skvrny z oblečení?	jak odstranit skvrny od oblečení	1	0.964
	jak vyčistit skvrny na oblečení	5	0.918
	jak odstranit mastný flek z oblečení	10	0.902
	odstranit mastné skvrny z prádla	50	0.823
	jak odstranit skvrny od fixy	100	0.798
	jak odstranit čokoládu z oblečení	1 000	0.667
	odstranění nálepky z oděvu	2 500	0.621
	Ruční prášek na praní.	10 000	0.556
	blaklader oděvy	20 000	0.524
	ODĚVNÍ KOŽENKA	30 000	0.505
What is the best way to remove stains from clothes?	jak odstranit skvrny od oblečení	1	0.930
	jak vyčistit skvrny na oblečení	5	0.905
	jak odstranit plísňové skvrny na oděvech	10	0.875
	jak vyčistit krev rez z oblečení	50	0.811
	jak vyprat tuž z oblečení	100	0.783
	čisté bílé oblečení praní	1 000	0.667
	jak prát oblečení na motorku	2 500	0.621
	jak vyprat kožene pracovní rukavice	10 000	0.556
	pohlcovac pachu na textil	20 000	0.523
	SMOG dámské džíny	30 000	0.504
boulder	Boulder	1	1.000
	bouldery	5	0.975
	co je Boulder	10	0.944
	Boulder Boskovice	50	0.851
	boulder vítkovice	100	0.814
	adam ondra lístky bouldering	1 000	0.692
	skála čsfd	2 500	0.658
	kameny z údolí	10 000	0.601
	lavinová lopata	20 000	0.566
	den braven kamenny na koberec	30 000	0.544
matematika	MATEMATIKA	1	1.000
	Matematiky	5	0.989
	matematické	10	0.983
	in matematika	50	0.935
	matematicky celkem	100	0.902
	Matematika - Úměrnosti	1 000	0.814
	tabulky matematické	2 500	0.756
	čssz kalkulacka	10 000	0.620
	co je desetinné číslo	20 000	0.512
	výpočet mm2	30 000	0.466

Conclusion

The primary aim of this thesis was to investigate STS, with a focus on the challenges faced by low-resource languages, such as Czech. Our work aimed to evaluate the performance of various pre-trained language models in solving the STS problem, demonstrate that common pre-training methods are not ideal for this task, identify potential improvements to enhance the performance of these models, and use these improvements to train new superior Czech STS models.

To achieve this, we conducted a comprehensive study of techniques and models for solving the STS problem, focusing on the role of neural networks, the Transformer architecture, and pre-trained language models such as BERT, RoBERTa, and ELECTRA. Our study also delved into the different strategies for generating sentence embeddings and the limited availability of Czech pre-trained models. We explored various language models and methods for obtaining sentence embeddings, including state-of-the-art English models and their comparative performance. The Cross-encoder and Bi-encoder architectures were presented, along with advanced methods for training Bi-encoders for STS tasks such as SimCSE or TSDAE. We investigated techniques such as knowledge distillation and transferring knowledge between languages.

Despite the challenges faced, such as limited resources for the Czech language and the complexity of the problem, our research goals were accomplished. We collected data for evaluation for both STS dataset downstream tasks. We evaluated PLMs on these tasks. We confirmed that methods like MLM or Replaced Token Detection are not ideal as stand-alone unsupervised methods for training models for the STS task or information retrieval without further fine-tuning. However, they can very effectively improve the performance for models that are subsequently fine-tuned using supervision on tasks like Sentiment Analysis.

We successfully trained various Czech STS models and confirmed that with unsupervised techniques like SimCSE or TSDAE, we can effectively create Bi-encoders from PLMs, improving performance on both STS and Information Retrieval, although these techniques negatively affect performance while fine-tuning on tasks that do not leverage sentence pairs. We also assessed the impact of multilingual distillation, revealing that our *czen-bert-small* model performed better on the STS task than other models, including LaBSE, the multilingual base model trained on a large supervised dataset. This finding indicated the potential benefits of using multilingual distillation for solving STS tasks in low-resource languages.

We hope our research provides valuable insights into the challenges and potential improvements for STS tasks in low-resource languages, paving the way for future advancements in natural language processing for languages like Czech.

Bibliography

1. DEERWESTER, Scott; DUMAIS, Susan T; FURNAS, George W; LANDAUER, Thomas K; HARSHMAN, Richard. Indexing by latent semantic analysis. *Journal of the American society for information science*. 1990, vol. 41, no. 6, pp. 391–407.
2. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 2013.
3. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.
4. BOJANOWSKI, Piotr; GRAVE, Edouard; JOULIN, Armand; MIKOLOV, Tomas. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*. 2017, vol. 5, pp. 135–146.
5. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2018.
6. AKHTER AL AMIN, Saad Hassan; ALM, Cecilia; HUENERFAUTH, Matt. Using BERT Embeddings to Model Word Importance in Conversational Transcripts for Deaf and Hard of Hearing Users. *LTEDI 2022*. 2022, p. 35.
7. LEE, Xi Jie; YAP, Timothy Tzen Vun; NG, Hu; GOH, Vik Tor. Comparison of Word Embeddings for Sentiment Classification with Preconceived Subjectivity. In: *International Conference on Computer, Information Technology and Intelligent Computing (CITIC 2022)*. Atlantis Press, 2022, pp. 488–502.
8. COLLOBERT, Ronan; WESTON, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.
9. RUMELHART, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *nature*. 1986, vol. 323, no. 6088, pp. 533–536.
10. MIKOLOV, Tomas; KARAFIÁT, Martin; BURGET, Lukas; CERNOCKỳ, Jan; KHUDANPUR, Sanjeev. Recurrent neural network based language model. In: *Interspeech*. Makuhari, 2010, vol. 2, pp. 1045–1048. No. 3.
11. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation*. 1997, vol. 9, no. 8, pp. 1735–1780.

12. CHO, Kyunghyun; VAN MERRIENBOER, Bart; GULCEHRE, Caglar; BAHDANAU, Dzmitry; BOUGARES, Fethi; SCHWENK, Holger; BENGIO, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. 2014.
13. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. *Advances in neural information processing systems*. 2017, vol. 30.
14. ROGERS, Anna; KOVALEVA, Olga; RUMSHISKY, Anna. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*. 2021, vol. 8, pp. 842–866.
15. WU, Yonghui; SCHUSTER, Mike; CHEN, Zhifeng; LE, Quoc V; NOROUZI, Mohammad; MACHEREY, Wolfgang; KRIKUN, Maxim; CAO, Yuan; GAO, Qin; MACHEREY, Klaus, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. 2016.
16. LIU, Yinhan; OTT, Myle; GOYAL, Naman; DU, Jingfei; JOSHI, Mandar; CHEN, Danqi; LEVY, Omer; LEWIS, Mike; ZETTLEMOYER, Luke; STOYANOV, Veselin. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 2019.
17. SENNRICH, Rico; HADDOW, Barry; BIRCH, Alexandra. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*. 2015.
18. RADFORD, Alec; WU, Jeffrey; CHILD, Rewon; LUAN, David; AMODEI, Dario; SUTSKEVER, Ilya, et al. Language models are unsupervised multitask learners. *OpenAI blog*. 2019, vol. 1, no. 8, p. 9.
19. CLARK, Kevin; LUONG, Minh-Thang; LE, Quoc V; MANNING, Christopher D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*. 2020.
20. WOLF, Thomas; DEBUT, Lysandre; SANH, Victor; CHAUMOND, Julien; DELANGUE, Clement; MOI, Anthony; CISTAC, Pierric; RAULT, Tim; LOUF, R'emi; FUNTOWICZ, Morgan, et al. Transformers: State-of-the-art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 38–45. Available also from: `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.
21. CONNEAU, Alexis; KHANDELWAL, Kartikay; GOYAL, Naman; CHAUDHARY, Vishrav; WENZEK, Guillaume; GUZMÁN, Francisco; GRAVE, Edouard; OTT, Myle; ZETTLEMOYER, Luke; STOYANOV, Veselin. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*. 2019.
22. REIMERS. *sbert.com website*. 2019. Available also from: `https://www.sbert.net/docs/pretrained_cross-encoders.html`.
23. *all-mpnet-base-v2 model* [`https://huggingface.co/sentence-transformers/all-mpnet-base-v2`]. [N.d.]. Accessed: May 10, 2023.
24. SONG, Kaitao; TAN, Xu; QIN, Tao; LU, Jianfeng; LIU, Tie-Yan. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*. 2020, vol. 33, pp. 16857–16867.
25. *all-MiniLM-L6-v2* [`https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`]. [N.d.]. Accessed: May 10, 2023.
26. WANG, Wenhui; WEI, Furu; DONG, Li; BAO, Hangbo; YANG, Nan; ZHOU, Ming. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*. 2020, vol. 33, pp. 5776–5788.

27. MUENNIGHOFF, Niklas; TAZI, Nouamane; MAGNE, Loïc; REIMERS, Nils. MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*. 2022.

28. STRAKA, Milan; NÁPLAVA, Jakub; STRAKOVÁ, Jana; SAMUEL, David. RobeCzech: Czech RoBERTa, a monolingual contextualized language representation model. In: *Text, Speech, and Dialogue: 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6–9, 2021, Proceedings 24*. Springer, 2021, pp. 197–209.

29. LEHEČKA, Jan; ŠVEC, Jan. Comparison of czech transformers on text classification tasks. In: *Statistical Language and Speech Processing: 9th International Conference, SLSP 2021, Cardiff, UK, November 23–25, 2021, Proceedings 9*. Springer, 2021, pp. 27–37.

30. SIDO, Jakub; PRAŽÁK, Ondřej; PŘIBÁŇ, Pavel; PAŠEK, Jan; SEJÁK, Michal; KONOPÍK, Miloslav. Czert–Czech BERT-like Model for Language Representation. *arXiv preprint arXiv:2103.13031*. 2021.

31. KOCIÁN, Matěj; NÁPLAVA, Jakub; ŠTANCL, Daniel; KADLEC, Vladimír. Siamese bert-based model for web search relevance ranking evaluated on a new czech dataset. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022, vol. 36, pp. 12369–12377. No. 11.

32. *BERT-Base, Multilingual Cased* [`https://github.com/google-research/bert/blob/master/multilingual.md`]. [N.d.]. Accessed: May 10, 2023.

33. FENG, Fangxiaoyu; YANG, Yinfei; CER, Daniel; ARIVAZHAGAN, Naveen; WANG, Wei. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*. 2020.

34. REIMERS, Nils; GUREVYCH, Iryna. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*. 2019.

35. REIMERS, Nils; BEYER, Philip; GUREVYCH, Iryna. Task-oriented intrinsic evaluation of semantic textual similarity. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 87–96.

36. BOWMAN, Samuel R; ANGELI, Gabor; POTTS, Christopher; MANNING, Christopher D. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*. 2015.

37. CONNEAU, Alexis; KIELA, Douwe. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*. 2018.

38. GAO, Tianyu; YAO, Xingcheng; CHEN, Danqi. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*. 2021.

39. CHEN, Ting; KORNBLITH, Simon; NOROUZI, Mohammad; HINTON, Geoffrey. A simple framework for contrastive learning of visual representations. In: *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

40. CHEN, Ting; SUN, Yizhou; SHI, Yue; HONG, Liangjie. On sampling strategies for neural network-based collaborative filtering. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 767–776.

41. CER, Daniel; DIAB, Mona; AGIRRE, Eneko; LOPEZ-GAZPIO, Inigo; SPECIA, Lucia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*. 2017.

42. WANG, Kexin; REIMERS, Nils; GUREVYCH, Iryna. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. *arXiv preprint arXiv:2104.06979*. 2021.

43. JANSON, Sverker; GOGOULOU, Evangelina; YLIPÄÄ, Erik; CUBA GYLLENSTEN, Amaru; SAHLGREN, Magnus. Semantic re-tuning with contrastive tension. In: *International Conference on Learning Representations, 2021*. 2021.

44. HINTON, Geoffrey; VINYALS, Oriol; DEAN, Jeff. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 2015.

45. LIU, Fangyu; JIAO, Yunlong; MASSIAH, Jordan; YILMAZ, Emine; HAVRYLOV, Serhii. Trans-Encoder: Unsupervised sentence-pair modelling through self-and mutual-distillations. *arXiv preprint arXiv:2109.13059*. 2021.

46. SHLENS, Jonathon. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*. 2014.

47. REIMERS, Nils; GUREVYCH, Iryna. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*. 2020.

48. SIDO, Jakub; SEJÁK, Michal; PRAŽÁK, Ondřej; KONOPÍK, Miloslav; MORAVEC, Václav. Czech news dataset for semantic textual similarity. *arXiv preprint arXiv:2108.08708*. 2021.

49. SVOBODA, Lukáš; BRYCHCÍN, Tomáš. Czech dataset for semantic textual similarity. In: *Text, Speech, and Dialogue: 21st International Conference, TSD 2018, Brno, Czech Republic, September 11-14, 2018, Proceedings 21*. Springer, 2018, pp. 213–221.

50. HABERNAL, Ivan; PTÁČEK, Tomáš; STEINBERGER, Josef. Sentiment analysis in czech social media using supervised machine learning. In: *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*. 2013, pp. 65–74.

51. KOCMI, Tom; POPEL, Martin; BOJAR, Ondrej. Announcing CzEng 2.0 Parallel Corpus with over 2 Gigawords. *arXiv preprint arXiv:2007.03006*. 2020.

52. PYTORCH. *PyTorch* [`https://github.com/pytorch/pytorch`]. GitHub, 2016. Accessed: May 10, 2023.

53. JOHNSON, Jeff; DOUZE, Matthijs; JÉGOU, Hervé. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*. 2019, vol. 7, no. 3, pp. 535–547.

54. GODBOLE, Varun; DAHL, George E.; GILMER, Justin; SHALLUE, Christopher J.; NADO, Zachary. *Deep Learning Tuning Playbook*. 2023. Available also from: `http://github.com/google/tuning_playbook`. Accessed: May 10, 2023.

55. OORD, Aaron van den; LI, Yazhe; VINYALS, Oriol. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*. 2018.

56. BIRD, Steven; LOPER, Edward; KLEIN, Ewan. *Natural Language Toolkit*. NLTK project, 2009–2021. Available also from: `https://www.nltk.org/`. Accessed: May 10, 2023.

57. LOSHCHILOV, Ilya; HUTTER, Frank. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. 2017.

58. WANG, Liang; YANG, Nan; HUANG, Xiaolong; JIAO, Binxing; YANG, Linjun; JIANG, Daxin; MAJUMDER, Rangan; WEI, Furu. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*. 2022.

59. LIU, Zheng; SHAO, Yingxia. Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder. *arXiv preprint arXiv:2205.12035*. 2022.