LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT
FÜR DATENBANKSYSTEME

Masterarbeit
in Informatik

Sentence Embeddings in Various Supervision Settings

Johanna Reiml

Aufgabensteller: Prof. Dr. Thomas Seidl
Betreuer: Dr. Michael Fromm
Abgabedatum: 13.02.2023

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

Munich, 13.02.2023

.....
Johanna Reiml

Abstract

Sentence embeddings have become an important component in various Natural Language Processing tasks, as they provide a compact representation of the meaning of sentences. However, current methods for training sentence embeddings such as TSDAE and Augmented SBERT are not optimal and can be improved. This thesis presents a comprehensive investigation of sentence embedding training in various supervision settings with the aim of improving current methods. A novel modification, the Cross-Bi-Encoder (CBE), is introduced for Cross-Encoders, a specialized architecture for sentence-pair tasks, to improve the distillation performance of Augmented SBERT. Ablation studies are conducted to gain insights into the learning process of TSDAE and a new objective, Noise Matching (NM), is introduced to incorporate contrastive learning for training noise-invariant embeddings. Additionally, the impact of incorporating labeled topic information in a supervised setting is investigated, with the goal of training topic-sensitive sentence embeddings. Experiments are conducted on established datasets and evaluated with SentEval and the Unsupervised Sentence Embedding Benchmark. Results are compared with state-of-the-art methods, demonstrating the proposed improvements. The code and results will be made publicly available on GitHub[1] for easy reproducibility.

[1] https://github.com/jreiml/master-thesis-sentence-embeddings

Contents

Chapter 1

Introduction

1.1 Motivation

Representation learning [55] is a crucial aspect of Natural Language Processing (NLP), as it enables the modeling of language meaning in a compact and comprehensive way. Sentence embeddings in particular provide a dense vector representation of sentence meaning and offer a more comprehensive representation compared to traditional approaches such as averaging or concatenating word embeddings. One of the significant benefits of sentence embeddings is their reusability - they only need to be computed once per sentence, making them resource-saving. This is crucial given the resource-intensive nature of modern NLP models, which often require expensive hardware for efficient use [7].

This thesis aims to explore sentence embeddings in various supervision settings in order to gain a deeper understanding of their role in NLP. In the context of semantic textual similarity, sentence embeddings are used to measure the similarity between two sentences [12]. In semantic search, sentence embeddings are used to retrieve relevant sentences based on a user's query [69]. Clustering is another application where sentence embeddings are used to group similar sentences together [69]. Paraphrase mining involves finding paraphrased sentences in a corpus, and sentence embeddings provide a way to measure the similarity between sentences for this task [58]. Translated sentence mining extends this idea to a multilingual setting, by matching translated sentences with the original sentences in a corpus [27].

Recent advances in deep learning, in particular the Transformer architecture, have led to significant improvements in the quality of sentence embeddings. The Transformer architecture, first introduced in "Attention Is All You

Need" [76], leverages the self-attention mechanism to analyze the relationships between tokens in a sentence. This led to the development of various Transformer models such as BERT [22], RoBERTa [54], or BART [48], which have achieved state-of-the-art results in NLP tasks. One of the first successful applications of BERT to sentence embeddings was Sentence-BERT [69], which proposed the use of a Siamese architecture for BERT, now commonly referred to as a Bi-Encoder [74] or Sentence Transformer[1].

However, while the quality of sentence embeddings has improved, the training of these embeddings remains an open research question [25, 37, 77, 74]. There are three main supervision settings for the training of sentence embeddings: unsupervised, semi-supervised, and supervised. In the unsupervised setting, sentence embeddings are trained on large amounts of unlabeled text data. In the semi-supervised setting, sentence embeddings are trained on a combination of labeled and unlabeled data. In the supervised setting, sentence embeddings are trained on labeled data for a specific task.

The different supervision settings each have their own unique advantages and difficulties. Unsupervised learning poses the challenge of evaluating the accuracy of embeddings and identifying relevant information in the absence of labeled data. While the use of limited labeled data in semi-supervised learning can improve the quality of the embeddings, finding the most efficient way to incorporate the unlabeled data remains a challenge. On the other hand, the quality of embeddings in supervised learning is largely dependent on the availability and quality of labeled data for the specific task.

In conclusion, sentence embeddings are a powerful tool for NLP tasks and have seen great advances with the introduction of Transformer models. However, the training of sentence embeddings remains a challenge, as different supervision settings present unique difficulties and advantages.

The objective of this thesis is to advance the study of sentence embeddings by exploring ways to improve existing methods, with a particular focus on low-resource settings. We focus on methods that do not rely on large-scale training on large datasets, which is not always feasible for various use cases, and explore ways to learn effectively from smaller datasets.

[1]The use of the term derives from the name of the most popular library for this architecture, the SentenceTransformers library.

1.2 Thesis Objectives

The focus of this thesis is to explore the training of sentence embeddings in different supervision settings with the aim of improving current methods such as the Augmented SBERT [74] and TSDAE [77]. To achieve this, we propose novel modifications to Cross-Encoders, a specialized architecture for sentence pair tasks. We refer to our proposed model as the Cross-Bi-Encoder (CBE). Our approach builds on the idea of the Augmented SBERT, which leverages the high performance of Cross-Encoders for distilling Bi-Encoders. Our goal is to improve the distillation process from Cross-Encoders to Bi-Encoders, thus producing better sentence embeddings. Additionally, we will perform ablation studies to gain insights into the learning process of TSDAE and introduce a new objective, Noise Matching (NM), which uses contrastive learning in order to train noise-invariant embeddings. Moreover, we will examine the effect of incorporating labeled topic information into sentence embeddings in a supervised setting. The aim is to train topic-sensitive sentence embeddings that result in a clear separation of topics in the embedding space while preserving semantic similarity between sentences. These topic-sensitive embeddings could be valuable for data analysis, where uncovering subgroups within known groups in a dataset may be of interest.

The study will be evaluated using well-established datasets such as the STS Benchmark [12], Spanish STS Corpus [74], UKP Sentential Argument Mining Corpus [71], BWS Argument Similarity Corpus [74], and 20 Newsgroups Dataset [46]. The evaluation of sentence embeddings will be performed using both SentEval [18] and the Unsupervised Sentence Embedding Benchmark [77]. The results of the experiments will be thoroughly analyzed and compared with state-of-the-art methods to show the effectiveness of the proposed improvements. In conclusion, this thesis aims to contribute to the field of sentence embeddings by introducing a novel architecture for Cross-Encoders, providing new insights into the learning process of the TSDAE, and investigating the impact of incorporating labeled topic information in a supervised setting.

1.3 Thesis Structure

The structure of this thesis is organized to cover the entire process of exploring the effectiveness of training sentence embeddings in unsupervised, semi-supervised, and supervised settings.

- Chapter 2 “Related Work” provides a comprehensive review of the existing literature in the field of sentence embeddings. It lays the foundation

for the research presented in this thesis by providing a background of the current state-of-the-art models.

- Chapter 3 "Datasets" describes the datasets selected for the experiments in this thesis. It explains the characteristics of each dataset and how they were selected to ensure a fair evaluation of the proposed methods.
- Chapter 4 "Methods" provides an in-depth understanding of the methods used, including the objectives and design decisions.
- Chapter 5 "Experiments" describes the experimental setup, including the evaluation metrics and our hyperparameters. It provides the readers with a comprehensive understanding of the experiments performed and how the results were obtained.
- Chapter 6 "Evaluation" presents and analyzes the results of the experiments, comparing the performance of our proposed improvements with existing methods. The chapter provides a detailed evaluation of the results, highlighting the strengths and weaknesses of our proposed methods.
- Chapter 7 "Conclusions and Future Work" summarizes the findings of this study, providing insights into the contribution made to the field of sentence embeddings. It concludes with suggestions for future research in the field, identifying areas for improvement and potential avenues for exploration.

Chapter 2

Related Work

Chapter 2 of this thesis focuses on the related work in the field of sentence embeddings.

- The first section of the chapter, 2.1 Supervision Settings, describes the different types of supervision that are used to train sentence embeddings, including supervised, semi-supervised, and unsupervised learning.
- Section 2.2 Transformer Architecture provides an overview of the Transformer architecture, which is widely used in the field of NLP. This section also introduces the self-attention mechanism and multi-head self-attention, which are key components of the Transformer architecture.
- Section 2.3 Transformer Descendants discusses several popular Transformer-based models that are used for sentence embeddings, such as BERT or RoBERTa. For each model, this section provides an overview of the model, its pre-training objectives, and its advantages and disadvantages.
- Section 2.4 Transformer Models for Sentence Embeddings describes several state-of-the-art methods for training sentence embeddings, including Universal Sentence Encoder, Sentence-BERT, SimCSE, TSDAE, and Augmented SBERT. This section provides an overview of each method, along with its advantages and disadvantages.

2.1 Supervision Settings

In machine learning, supervision refers to the process of providing labeled data to a model so that it can learn. There are three main forms of supervision in deep learning: supervised learning, semi-supervised learning, and unsupervised learning. Each plays an important role in the development and training of deep

learning models. In the following sections, we will provide a formal definition for each approach and explore its characteristics and applications, with a focus on single-label classification tasks, which are simple to describe and implement.

2.1.1 Supervised Learning

Supervised learning is a machine learning paradigm in which a model is trained on labeled data. In this setting, the model receives input and generates predictions that are then compared to the labeled data to refine the model's performance. The accuracy of the model can be easily assessed in a supervised setting as the predictions can be compared to the provided labels [68]. In the following sections, we will provide a formal definition and illustration of supervised learning, and also discuss its strengths and weaknesses.

2.1.1.1 Definition and Overview

Let $X = \{x_i\}_{i=1}^n$ be a dataset of n input instances and $Y = \{y_i\}_{i=1}^n$ be the corresponding list of labels. In supervised learning, the goal is to learn a mapping function m_θ that maps input instances to their target output, where θ are the parameters of the model. This is achieved by minimizing a loss function $\mathcal{L}_s$ with respect to θ. Formally, the optimization problem in supervised learning can be expressed as,

$$\min_\theta \sum_{i=1}^{n} \mathcal{L}_s(x_i, y_i, \theta) \tag{2.1}$$

The most common types of problems in this category are regression and classification problems. In a regression problem, the labels y_i are continuous values and the loss function $\mathcal{L}_s$ is typically the mean squared error. In a classification problem, the labels are discrete class labels $y_i \in \{c_i\}_{i=1}^k$, where k is the number of classes. The loss function $\mathcal{L}_s$ is usually the cross-entropy loss in this case.

2.1.1.2 Advantages and Disadvantages

Supervised learning is considered to be the simplest of the three supervision settings. It achieves high performance on tasks with high-quality labeled data, allowing the model to accurately predict scores or discrete classes of arbitrary data. The trained model can be evaluated on a test split, which ensures that the model has learned a generalized solution for the task and not just memorized the training data. However, acquiring high-quality labeled data can be challenging, as it may require the use of costly human annotators who may introduce biases or make errors in annotating the data. There is also a special type of supervised learning called "weakly supervised learning". This

framework recognizes the limitations of perfect labels and develops methods to robustly handle imperfect labels [88].

2.1.1.3 Example

Let $X = \{(x_1, x_2)_i\}_{i=1}^n$ be a set of n input sentence pairs, and let $Y = \{y_i\}_{i=1}^n$ be a set of corresponding target similarity scores. The supervised learning task is to learn a function m_θ that maps an input sentence pair (x_1, x_2) to its predicted similarity score $p \in [0, 1]$ by minimizing the mean squared error (MSE) between the predicted similarity score $p = m_\theta((x_1, x_2))$ and the target similarity score y:

$$\mathcal{L}_{\text{MSE}}((x_1, x_2), y, \theta) = (y - m_\theta((x_1, x_2)))^2 \tag{2.2}$$

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{MSE}}((x_1, x_2)_i, y_i, \theta) \tag{2.3}$$

The parameters θ of the function m_θ can be learned by minimizing the loss function over the training set X with optimization algorithms such as gradient descent. The optimization process adjusts the parameters of the function m_θ until the loss function is minimized, resulting in the best possible mapping between sentence pairs and their similarity scores.

2.1.2 Semi-Supervised Learning

Semi-supervised learning (SSL) extends the supervised learning framework by incorporating unlabeled data into the learning process, thereby improving data efficiency and performance. The collection of large-scale labeled datasets can require significant resources, time, and effort, making SSL a viable alternative [65]. In this section, we will formally define SSL, provide an example, and examine its advantages and disadvantages. We will also outline the conditions necessary for SSL to be effective.

2.1.2.1 Definition and Overview

Let $X = \{x_i\}_{i=1}^n$ be a dataset of n input instances, with a labeled subset $X_L = \{x_i\}_{i=1}^{n_L}$ and corresponding labels $Y_L = \{y_i\}_{i=1}^{n_L}$, and an unlabeled subset $X_U = \{x_i\}_{i=1}^{n_U}$, where it is assumed that $n_L \ll n_U$. The objective in SSL is to learn a mapping function m_θ that maps input instances to their target output, where θ represents the parameters of the model. Formally, this is achieved by optimizing the following equation,

$$\min_\theta \sum_{x \in X_L, y \in Y_L} \mathcal{L}_s(x, y, \theta) + \alpha \sum_{x \in X_U} \mathcal{L}_u(x, \theta) + \beta \sum_{x \in X} \mathcal{R}(x, \theta) \tag{2.4}$$

where $\mathcal{L}_s$ is the per-example supervised loss, such as cross-entropy for classification or mean squared error for regression, $\mathcal{L}_u$ is the per-example unsupervised loss, and $\mathcal{R}$ is a regularization term that enforces desired properties on the model m_θ, including prior knowledge, smoothness, or consistency constraints between the labeled and unlabeled data. α and β are positive real numbers representing the tradeoff between the supervised loss, the unsupervised loss, and the regularization term [84].

2.1.2.2 Assumptions

SSL relies on certain assumptions in order to effectively improve prediction accuracy by integrating unlabeled data. These assumptions include the self-training assumption, the co-training assumption, the generative model assumption, the cluster assumption, the low-density separation assumption, and the manifold assumption [84].

The self-training assumption postulates that predictions with high confidence tend to be accurate, especially when classes form well-separated clusters. The co-training assumption states that instances have two conditionally independent views, and each view is sufficient for a classification task. The generative model assumption stipulates that data is generated from a mixture of distributions, and a valid link can be made between the distribution of unlabeled data and category labels if the generative model is correct.

The cluster assumption holds that data in the same class tend to form clusters, and the decision boundary should not cross high-density areas. The low-density separation assumption asserts that the decision boundary should be in a low-density region. Finally, the manifold assumption states that points in a local neighborhood of the low-dimensional manifold have similar class labels.

It is important to note that the validity of these assumptions must be met for SSL to be effective, otherwise, performance may decrease. For example, using unlabeled data from scientific papers to train a spam classifier for email would not be beneficial as the datasets are vastly dissimilar.

2.1.2.3 Advantages and Disadvantages

SSL has both advantages and disadvantages over traditional supervised methods. On one hand, SSL allows for better performance with fewer labeled data points and can extract additional information from the unlabeled data to further improve results. On the other hand, SSL methods tend to be more

complex and require specialized knowledge and computational resources, as well as data augmentation techniques which can be challenging to implement in NLP tasks.

2.1.2.4 Example

In this example, we outline an SSL method that uses a combination of labeled and unlabeled data. We first train a teacher model on a labeled dataset consisting of n_L sentence pairs $X_L = \{(x_1, x_2)_i\}_{i=1}^{n_L}$ and their corresponding similarity scores $Y_L = \{y_i\}_{i=1}^{n_L}$. The teacher model is represented by a function m_{θ_t} that maps a sentence pair (x_1, x_2) to its predicted similarity score $p_t \in [0, 1]$. The goal of the teacher model is to learn the optimal parameters θ_t that minimize the mean squared error (MSE) loss function between the predicted outputs p_t and the ground truth scores $y \in Y_L$:

$$\min_{\theta_t} \frac{1}{n_L} \sum_{(x_1,x_2)\in X_L, y\in Y_L} \mathcal{L}_{\text{MSE}}((x_1, x_2), y, \theta_t) \tag{2.5}$$

The parameters θ_t of the function m_{θ_t} can be learned by minimizing the supervised loss function over the training set X_L with optimization algorithms such as gradient descent.

Next, we use the trained teacher model to generate predicted similarity scores for an unlabeled dataset $X_U = \{(x_1, x_2)_i\}_{i=1}^{n_U}$, resulting in the silver labels $Y_U = \{m_{\theta_t}((x_1, x_2))\}_{(x_1,x_2)\in X_U}$. The combined dataset $X = X_L \cup X_U$ is then used to train a student model, represented by a function m_{θ_s} that maps a sentence pair (x_1, x_2) to its predicted similarity score $p_s \in [0, 1]$. The student model aims to minimize the MSE loss functions between the predicted outputs p_s and the corresponding ground truth scores y, where y is either a score from the labeled dataset or a score generated by the teacher model.

$$\min_{\theta_s} \frac{1}{n_L} \sum_{(x_1,x_2)\in X_L, y\in Y_L} \mathcal{L}_{\text{MSE}}((x_1, x_2), y, \theta_s) + \frac{\alpha}{n_U} \sum_{(x_1,x_2)\in X_U, y\in Y_U} \mathcal{L}_{\text{MSE}}((x_1, x_2), y, \theta_s) \tag{2.6}$$

In this equation, α is the positive real-valued importance term for the unsupervised loss. The parameters θ_s of the function m_{θ_s} can be learned by minimizing the loss function over the training set X with optimization algorithms such as gradient descent.

In this semi-supervised learning procedure, the teacher model provides ad-

ditional labeled data for the student model to learn from, allowing the student model to learn from all available data [74].

2.1.3 Unsupervised Learning

Unsupervised learning is the most challenging of the three supervision settings in deep learning, because it requires the model to extract patterns and relationships within the data without explicit labels. This presents a significant challenge in terms of evaluation and increases the risk of the model learning irrelevant patterns [68]. In the following sections, we will provide a clear definition of unsupervised learning and discuss its advantages and disadvantages. We will also define the concept of a domain and give a brief overview of Self-Supervised Learning, a special type of unsupervised learning.

2.1.3.1 Definition and Overview

Let $X = \{x_i\}_{i=1}^{n}$ be a dataset of n input instances without any corresponding labels. The goal of unsupervised learning is to uncover the underlying structure and patterns in the data by learning a mapping function m_θ that maps input instances to their target output, where θ are the parameters of the model. This is achieved by minimizing a loss function $\mathcal{L}_u$ with respect to θ. Formally, the optimization problem in unsupervised learning can be expressed as:

$$\min_{\theta} \sum_{i=1}^{n} \mathcal{L}_u(x_i, \theta)$$

In unsupervised learning, there is no direct supervision signal in the form of labeled examples, and the learned representation must implicitly capture the underlying structure and patterns of the input data. Unsupervised learning techniques have various applications, including auto-encoders, clustering, and dimensionality reduction [9].

This thesis focuses primarily on the use of unsupervised learning for pre-training and domain adaptation through self-supervised learning.

2.1.3.2 Self-Supervised Learning

Self-supervised learning is a form of unsupervised machine learning that trains a model by using the data itself as supervision, as opposed to traditional supervised learning that relies on manually annotated data. In self-supervised learning, the primary objective is not to achieve optimal performance on the pre-training task, but rather to extract rich and transferable features that can

be used for downstream tasks [65].

The reason for our interest in self-supervised learning is due to its ability to learn general strong language models. By continuing the pre-training objectives of the original self-supervised language model on the target domain, we aim to obtain transferable features that can improve the performance of supervised learning algorithms. This approach, known as *domain-adaptive pre-training*, has been shown to lead to improved performance in both high and low resource settings [28].

2.1.3.3 Domain

In machine learning, a domain is defined as $\mathcal{D} = \{\mathcal{X}, P(\mathcal{X})\}$, where $\mathcal{X}$ represents the feature space and $P(\mathcal{X})$ is the marginal probability distribution over the feature space. The task at hand, such as semantic textual similarity, is defined as $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, where $\mathcal{Y}$ is the label space and $P(Y|X)$ is the likelihood. The goal of learning algorithms is to estimate the prior distribution $P(Y)$ and the likelihood $P(Y|X)$ from the training data $\{(x_i, y_i)\}_{i=1}^{n}$ [68].

When the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ have different distributions, $P_S(X) \neq P_T(X)$, the domain adaptation problem arises. In NLP, the term "domain" is often used to refer to a type of corpus determined by a given dataset. However, Plank (2016) proposed the concept of "variety space" to better capture the underlying linguistic differences in the data samples used in NLP [67].

In the variety space, a corpus is considered as a subspace or sample from the high-dimensional variety space, where dimensions or latent factors are related to linguistic aspects such as genre, subdomain, and socio-demographic factors, among others. Neglecting these underlying factors can lead to overfitting on overrepresented domains and lack of robustness in models. Thus, understanding the variety space and its impact on the data is crucial for developing more robust and generalizable models in NLP [68].

As mentioned earlier, self-supervised learning can help mitigate the domain adaptation problem by allowing models to extract transferable features from the data itself, before fine-tuning them on their target task [28].

2.1.3.4 Advantages and Disadvantages

Unsupervised learning is a powerful approach in machine learning, especially in NLP, where large amounts of unlabeled data are often available. One of its

key advantages is its ability to learn from raw input data, which can lead to the development of robust models through self-supervised learning.

However, unsupervised learning also presents several challenges. One of the most significant is evaluation, as it can be difficult to determine the performance of a trained model in real-world settings without the presence of labels. This often requires manual evaluation, which can be time-consuming and subjective. Additionally, even if we know that there are certain patterns in the data, it can be difficult to formulate tasks that learn the desired patterns without the help of labels.

Despite these challenges, unsupervised learning remains an important area of research in NLP because of its potential to extract valuable insights from large amounts of data. By harnessing the power of deep learning algorithms, unsupervised learning can help us better understand the relationships and patterns within data, leading to the development of more sophisticated and effective NLP models.

2.1.3.5 Example

Unlike supervised and semi-supervised learning, unsupervised learning presents a unique challenge in that it is not always easy to provide a clear and straightforward example. Due to the fact that unsupervised learning requires the model to learn patterns and relationships within the data without explicit labels, it can be difficult to formulate the exact objective to be learned.

While it may be challenging to provide a clear example of unsupervised learning, the success of self-supervised learning in NLP highlights the potential of unsupervised learning for training robust models. In fact, many state-of-the-art methods for training sentence embeddings also rely on unsupervised learning [77] [74] [25]. The following sections of the related work will further demonstrate the effectiveness and significance of unsupervised learning in the field of NLP.

2.2 Transformer Architecture

The Transformer architecture, first introduced by Vaswani et al. [76], has revolutionized the field of Natural Language Processing (NLP). The model's innovative approach to sequence-to-sequence modeling [73] has inspired numerous spin-off models, including BERT [22], which is based on the encoder mechanism of the Transformer. In the following sections, we will give a brief overview

of the Transformer architecture, and discuss two of its key components: the Self-Attention Mechanism and Multi-Head Self-Attention.

2.2.1 Overview

The Transformer consists of an encoder and a decoder, each consisting of stacks of identical blocks. Each encoder block is composed of a multi-head self-attention module and a position-wise feed-forward network (FFN), with residual connections [31] and layer normalization [7] applied around each sub-layer module. The decoder block has a similar architecture, but with the addition of cross-attention modules between the multi-head self-attention modules and position-wise FFNs. Additionally, the self-attention layer in the decoder has been modified to prevent tokens from attending to subsequent tokens, using a causal attention mask represented by a lower triangular matrix.

The Transformer architecture has made a significant impact in the field of deep learning, including NLP, largely due to its innovative use of the attention mechanism [8]. This mechanism allows the model to attend to different parts of the input sequence and weigh their importance.

Given the importance of the self-attention mechanism, it is valuable to dive deeper into its workings. The next subsection, "Self-Attention Mechanism", will provide a comprehensive examination of this critical component of the Transformer architecture.

2.2.2 Self-Attention Mechanism

Unlike traditional LSTMs, which are limited to reading text either left-to-right or right-to-left [59], the Transformer's self-attention mechanism allows the model to attend to each input token simultaneously in a bidirectional manner. This mechanism is implemented through a Query-Key-Value (QKV) model, where for each token's input vector, a query, key, and value vector are created by multiplying the input vector with the weights contained in the W^Q, W^K, and W^V matrices, respectively. These weight matrices are learned during the training process.

The self-attention layer in the Transformer architecture receives as input a token embedding matrix of the original input sequence, formally defined as $X = (x_i)_{i=1}^n$. The output of the embedding layer is represented as $E \in \mathbb{R}^{|X| \times d}$, where d is the embedding dimension. The encoder or decoder, each consisting of l blocks, processes the input matrix of token embeddings $H_{l-1} \in \mathbb{R}^{|X| \times d}$

from the previous layer and outputs a new matrix $H_l \in \mathbb{R}^{|X| \times d}$ for each block $\mathcal{F}$.

For each self-attention layer, the Transformer generates three matrix representations of queries $Q \in \mathbb{R}^{|X| \times d_k}$, keys $K \in \mathbb{R}^{|X| \times d_k}$, and values $V \in \mathbb{R}^{|X| \times d_v}$. The dimensions of the queries and keys are represented by d_k, while the dimension of the values is represented by d_v. The output matrix after each self-attention layer is formally denoted as:

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

where the resulting matrix is referred to as the attention matrix $\mathcal{A}$. The attention mechanism computes the dot product between the query and key matrices, which are then divided by $\sqrt{d_k}$ and passed through a softmax function to obtain the weights on the values. This mechanism is referred to as Scaled Dot-Product Attention. The original Transformer paper proposed the hyperparameters $d = 512$ for the model hidden dimension and $l = 6$ for the number of layers in the encoder and decoder [76].

2.2.3 Multi-Head Self-Attention

The Transformer architecture utilizes the concept of multi-head self-attention in order to improve the performance of its attention mechanism. Instead of applying a single attention function, the Transformer applies multiple attention functions in parallel, each of which is referred to as a "head". The multi-head self-attention layer is defined as the concatenation of all heads, allowing the model to jointly attend to information from different representation subspaces at different positions.

Mathematically, the multi-head self-attention layer can be represented as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{and } W_i^Q \in \mathbb{R}^{d \times d_k}, W_i^K \in \mathbb{R}^{d \times d_k}, W_i^V \in \mathbb{R}^{d \times d_v}, W^O \in \mathbb{R}^{hd_v \times d}$$

In this equation, Q, K, and V represent the queries, keys, and values, respectively, and W_i^Q, W_i^K, and W_i^V are the weight matrices for each head. Additionally, W^O is the weight matrix for the output, and h is the number of

attention heads.

In the original Transformer architecture, $h = 8$ attention heads were used. To accommodate for the multiple attention heads, the dimension of the weights was scaled down by a factor of h, resulting in $d = d_k = d_v = 512/h = 64$ [76].

2.3 Transformer Descendants

In this section, we will examine the advances made by three specific Transformer descendants that have leveraged the Transformer's innovative architecture and combined it with self-supervised learning objectives. The models to be discussed are BERT [22], RoBERTa [54], and BART [48].

2.3.1 BERT

BERT, an acronym for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, introduced the innovative approach of incorporating a self-supervised learning objective into the Transformer architecture. In the following sections, we will delve into the intricacies of BERT's input representation and its training methodology.

2.3.1.1 Overview

BERT has achieved great success in NLP and established itself as a new baseline, outperforming eleven previous baseline models [22]. BERT popularized the use of Masked Language Modeling (MLM), which is based on the idea behind the *Cloze* task in literature. As a result, BERT is considered one of the most influential models in the field of NLP.

2.3.1.2 Input Representations

BERT represents each word-piece in the corpus using token embeddings [82]. The model processes raw text into a sequence of token ids using a word-piece tokenizer. To enhance the input representation, BERT also incorporates segment and position embeddings. The former distinguishes between text pairs, while the latter indicates the absolute position of each token within the sequence. The final input representation for a token is obtained by summing its token, segment, and position embeddings, as shown in Figure 2.1.

BERT uses two special tokens to further enhance the input representation, the [CLS] and [SEP] tokens. The [CLS] token is added at the beginning of

each sequence and its hidden state represents the entire sequence, serving as the basis for sentence-level tasks. The [SEP] token marks the end of a sequence. If there is only one sequence in the input, the [SEP] token is added at the end. For pairwise tasks, it also marks the end of the first sequence and the beginning of the second.

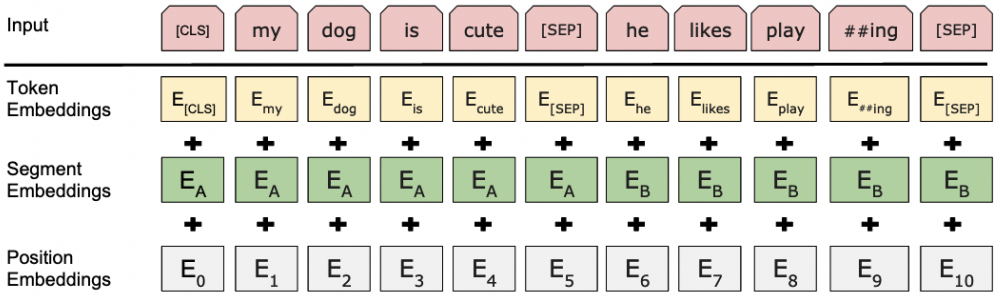


Figure 2.1: BERT input representation. Adopted from Devlin et al. [22], page 4.

2.3.1.3 Pre-training Objectives

BERT introduced two self-supervised learning objectives to learn strong contextualized token embeddings and a robust model. These objectives are Masked Language Modeling and Next Sentence Prediction.

Masked Language Modeling

Masked Language Modeling (MLM) is a self-supervised pre-training objective that aims to predict the original tokens of a partially masked input sequence. In MLM, a portion of the input tokens are replaced by the [MASK] token, with the goal of reconstructing the original tokens.

BERT selects 15% of the input tokens, replacing 80% with [MASK], replacing 10% with a random token, and leaving the remaining 10% unchanged. The masked input sequence is then fed into the Transformer encoder. The hidden representations of the input tokens from the last layer of the encoder are then passed through a decoder head, $D \in \mathbb{R}^{d \times |V|}$, where $|V|$ represents the size of the vocabulary. The decoder head projects the hidden representation of each input token to the size of the vocabulary and assigns a score to each candidate token.

The model is trained to reconstruct the original token for the 80% of tokens that were replaced by [MASK], to recover the token for the 10% of tokens that were replaced by a random token, and to predict the original token for the 10% of tokens that were left unchanged. The loss used for this objective is the cross-entropy loss.

It is important to note that the 80-10-10 strategy used by BERT is arbitrary. The authors of BERT were concerned that if the model only made predictions on the [MASK] token, it would suffer from fine-tuning mismatch, as the [MASK] token is only part of the pre-training process. However, subsequent research has shown that there is no need for the 80-10-10 strategy [79]. We discuss this in more detail in section 2.3.2.3.

Next Sentence Prediction

BERT also introduced Next Sentence Prediction as a secondary self-supervised pre-training objective, aimed at teaching the model the concept of sentence pair tasks. This objective could be beneficial for downstream tasks such as Question Answering [89]. The objective is to determine whether the sentence x_2 is the immediate next sentence after the sentence x_1 in a sentence pair (x_1, x_2).

To create the training data for this task, half of the samples were real sentence pairs, while the other half consisted of the first sentence followed by a random sentence sampled from another document in the corpus. However, recent research has shown that this task may not be necessary and may even be detrimental to performance [54]. Instead, it has been theorized that instead of the intended Next Sentence Prediction objective, the model instead learns a relatively easy topic modeling objective, such as "Are Sentence A and Sentence B about the same topic?" [45].

2.3.1.4 Fine-Tuning

BERT can be fine-tuned in a number of ways, depending on the downstream task. For instance, when fine-tuning BERT on a simple classification or regression task, a task-specific head is added to the pre-trained model, and all parameters are jointly fine-tuned on the downstream task. This approach is described in detail in the original paper [22]. Alternatively, the loss can be applied directly to the output embeddings. This approach can be used in sentence embedding models, where contrastive learning methods are commonly used to learn sentence representations [25, 37].

2.3.2 RoBERTa

RoBERTa, or A **R**obustly **O**ptimized **BERT** Pretraining **A**pproach, was introduced in 2019 to optimize the original BERT model. The authors of RoBERTa aimed to rigorously evaluate each design choice made in the BERT architecture, including hyperparameters, training data, training objectives, and data processing procedures [54]. In the following sections, we will provide a comprehensive overview of the RoBERTa model, including an examination of the modifications made to the original BERT architecture and a discussion of potential future enhancements.

2.3.2.1 Overview

Despite BERT's impressive performance in the field of NLP, researchers continued to strive for even higher scores on benchmarks. While other models, such as XLNet [85] and XLM [43], proposed modifications to the BERT architecture and achieved better results, they often resulted in more complex models than BERT. In contrast, the authors of RoBERTa took a different approach. By conducting a thorough examination of BERT's training procedures and design choices, they identified several areas for improvement. Based on this analysis, RoBERTa was developed as a more robust and optimized version of BERT.

One of the key findings of the RoBERTa authors was that the original BERT model was under-trained, which led to several optimizations in the training procedures. These optimizations resulted in improved performance, surpassing the results of every model published since the inception of BERT on four out of nine GLUE tasks and matching the state-of-the-art results on the SQuAD and RACE benchmarks.

The results of the RoBERTa study demonstrate the effectiveness of the simple method behind BERT and show that the simple BERT recipe, with thorough optimization, can yield highly competitive models. RoBERTa has established itself as a new baseline model alongside BERT due to its impressive performance.

2.3.2.2 Modifications and Advantages over BERT

RoBERTa made several modifications to the BERT architecture in order to improve its performance. One of the most significant modifications was to increase the size of the training dataset. While BERT was trained on only 13 GB of data from the *bookcorpus* and *emph* datasets, RoBERTa was trained on a much larger dataset of 160 GB. The additional data consisted largely of web-

crawled data and included a novel dataset called *CC-News*, which accounted for 76 GB of the total data. This larger training corpus exposed RoBERTa to a wider range of data examples.

Another modification made to the model was to increase the batch size during training. The original BERT model was trained with a batch size of 256, while RoBERTa was trained with a batch size of 2048. This allowed each weight update to consider more data examples at once.

RoBERTa also made modifications to the Masked Language Modeling (MLM) objective. In BERT, the tokens to be masked were determined at the beginning of training and remained fixed throughout the entire process, which could lead to learning subtle biases. RoBERTa, on the other hand, selected the tokens to be masked dynamically and randomly, avoiding these potential biases.

Another change made by RoBERTa was the removal of the Next Sentence Prediction objective. The authors of RoBERTa found that this objective was not necessary for the model to learn relationships between sentences, and in fact degraded performance in many tasks. As a result, they removed this objective, simplifying the training procedure. This also led to the effective removal of segment embeddings. They were still included for compatibility reasons, but there is now only one type of segment embedding now.

Finally, unlike BERT, RoBERTa did not randomly inject short sequences and trained its model with full-length ($T = 512$) sequences for the entire duration of training, further simplifying the training process.

These modifications, combined with the larger training corpus, resulted in a more robust model with improved performance [54].

2.3.2.3 Remaining Issues

Despite the impressive performance of RoBERTa, there are still areas for improvement that can further enhance the model's capabilities. One such area is the Masked Language Modeling (MLM) objective used in RoBERTa, which has not yet been fully analyzed.

One aspect of interest is the 80-10-10 rule, where 80% of the tokens in the input sequence are masked, 10% are randomly replaced with a different token, and 10% are left unchanged. However, recent research has shown that a simpler strategy, such as masking all chosen tokens, can perform just as well or

even better [79].

Additionally, the authors of RoBERTa did not examine the impact of the masking rate on performance, which has been shown to be significant. Wetting et al. (2022) found that pre-training with a masking rate of up to 40% can outperform the 15% baseline, and even a masking rate of 80% can preserve most of the performance as measured by fine-tuning on downstream tasks. They suggest that increasing the masking rate has two effects: 1) more of the input tokens are corrupted, which reduces the context size and creates a more challenging task, and 2) the models make more predictions, which benefits training. In their study, they tried to disentangle the effects of increased corruption and predictions and concluded that more predictions help and more corruption hurts. They suggested that increasing the batch size might have a similar effect as having more predictions. However, it must be kept in mind that having a very low corruption rate for MLM could significantly increase model training time, which represents a tradeoff [79].

Another recent study, Vega V2, which at the time of writing holds the top position on the SuperGLUE benchmark leaderboard, proposed a post-training optimization procedure for pre-trained language models. The authors introduced a "self-evolution learning mechanism", where the model first tries to identify the most challenging examples and then tries to learn from them, masking only the tokens that the model finds the most difficult to encode [87]. This approach incorporates a similar idea to the corruption rate minimization proposed by Wetting et al. (2022), as the authors effectively try to mask only those tokens where they expect the model to benefit from learning.

Additionally, studies have shown that the BERT architecture can still be improved. In their model DeBERTa, He et al. (2021) empirically demonstrated the necessity of computing self-attention with disentangled matrices based on their contents and relative positions, a mechanism they call disentangled attention [32]. DeBERTa also integrated an enhanced mask decoder, which is used to incorporate absolute positions in the decoding layer to predict the masked tokens in model pre-training. However, interestingly, Zhong et al. (2022) found in their preliminary studies for Vega V2 that the enhanced mask decoder did not actually improve the results.

Therefore, there is still room for improvement in both the BERT architecture and the MLM training procedure, and further research in this area is necessary to further advance the field of pre-trained language models.

2.3.3 BART

BART, or **B**idirectional and **A**uto-**R**egressive **T**ransformers, was the first to apply the combination of the Transformer architecture and self-supervised learning to sequence-to-sequence models. It leverages the power of bidirectional representations in the encoder and autoregressive generation in the decoder. In the following sections, we will give a brief overview of BART, discuss its pre-training objective and discuss its advantages and disadvantages.

2.3.3.1 Overview

BART is a Transformer-based sequence-to-sequence model that has been developed to address the limitations of BERT in handling generation tasks such as text summarization[24] and machine translation[76]. The model consists of two primary components: a bidirectional encoder, and a left-to-right autoregressive decoder.

The encoder takes in a corrupted text, generated by an arbitrary noising function, and computes hidden representations for each token in the sequence. The decoder, on the other hand, takes in the corrupted input sequence and the hidden representations from the encoder. Its objective is to denoise the text provided to the encoder and to reconstruct the original text [48]. An illustration of this architecture can be seen in Figure 2.2.

It is important to note that we will not be working directly with BART in this thesis. However, later in the thesis, we will discuss the TSDAE, an unsupervised method inspired by BART, which is used for learning sentence embeddings.

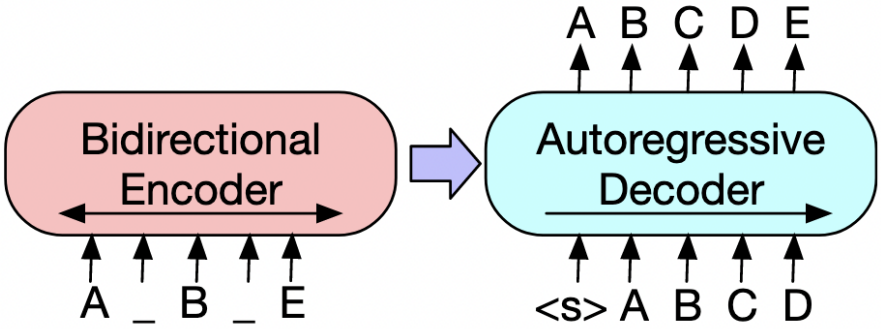


Figure 2.2: Overview model architecture for BART. Adopted from Lewis et al. [48], page 2.

2.3.3.2 Advantages and Disadvantages

One of the major advantages of BART is its flexibility in the choice of pre-training objectives. This is due to the fact that BART's objective is not tied to a specific noising function, which allows for the definition of various objectives. This flexibility enables researchers to experiment with and evaluate the performance of different objectives, thereby enabling a wider range of self-supervised tasks than what would be possible with encoder-only models.

In particular, the decoder component of BART allows the model to effectively handle generation tasks that require the production of output sequences, such as text summarization. Furthermore, when a classification task requires a more complex format than a simple numeric label, the use of a decoder can be more beneficial.

It is important to note, however, that encoder-decoder models, including BART, can be computationally expensive, particularly when the target output sequence to be generated is long. This can limit the scalability of the model and must be carefully considered when deciding whether to use BART for a specific task.

2.3.3.3 Denoising Approaches

BART is pre-trained by optimizing a reconstruction loss, which is the cross-entropy between the output of the decoder and the original document. Unlike BERT, which focuses on a specific token masking strategy, BART experiments with various transformations, including:

- **Token Masking:** This approach is similar to BERT's masking strategy, where tokens are randomly masked.
- **Token Deletion:** The model must determine the positions of randomly deleted tokens from the input.
- **Text Infilling (novel):** A number of text spans are sampled, with span lengths drawn from a Poisson distribution ($\lambda = 3$). Each span is replaced by a single [MASK] token. Spans of length 0 correspond to the insertion of [MASK] tokens. This approach teaches the model to predict the number of missing tokens in a span. It was inspired by SpanBERT, which also masks spans instead of individual tokens [38].
- **Sentence Permutation:** The document is divided into sentences based on full stops, and these sentences are shuffled in random order.

- **Document Rotation:** A token is randomly chosen and the document is rotated so that it begins with that token. This task trains the model to identify the start of the document.

All approaches are visualized in Figure 2.3. BART achieved the best performance by combining random shuffling of the original sentences and using the novel text infilling scheme, where text spans are replaced with a single mask token [48].

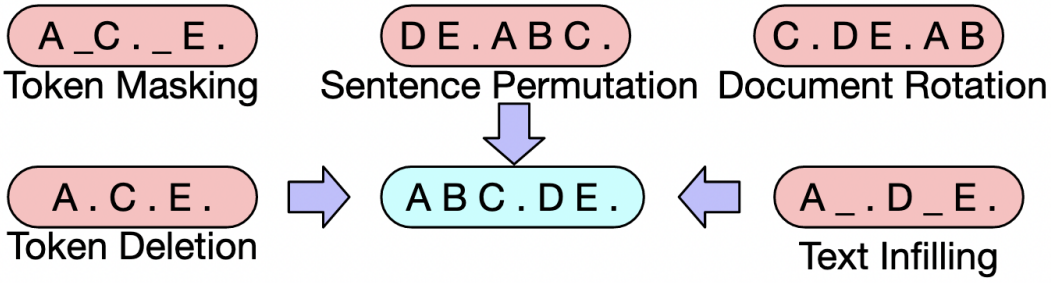


Figure 2.3: Overview denoising approaches for BART. Adopted from Lewis et al. [48], page 3.

2.4 Transformer Models for Sentence Embeddings

The Transformer architecture has demonstrated its effectiveness across a wide range of NLP tasks and has inspired its application to sentence embedding training. The Transformer's self-attention mechanism provides contextualized representations, contributing to its success in this area.

This section provides a brief overview of the history of sentence embeddings and focuses on the use of Transformer-based models for this purpose. Key methods in the field are examined in detail, including Universal Sentence Encoder [13], Sentence-BERT [69], SimCSE [25], TSDAE [77], and Augmented SBERT [74].

2.4.1 Sentence Embeddings

In this section, we give a brief history of sentence embeddings and a brief overview of how they are constructed in Transformer-based models.

2.4.1.1 History

The term "embedding" in machine learning refers to dense high-dimensional vector representations used to encode input data. In NLP, word and sentence embeddings represent words and sentences, respectively. The history of word embeddings includes traditional methods such as word2vec [61] and GloVe embeddings [66].

Obtaining sentence embeddings has proven to be more challenging than word embeddings. Two main approaches have emerged: 1) averaging pre-trained word embeddings, or 2) training a sentence embedding model directly. The first approach, averaging pre-trained word embeddings, has outperformed more complex supervised methods such as RNNs and LSTMs [6]. Some methods have also achieved strong supervised learning results on Natural Language Inference (NLI) datasets [19, 13, 69].

Traditional sentence embedding methods are limited by the static and non-contextual representation of the underlying pre-trained word embeddings. Words like "bank" have different meanings in different contexts. Naively applying pooling strategies to pre-trained Transformer models that have not been trained for sentence embedding tasks, results in poor performance [69]. Therefore, fine-tuning Transformer models is considered necessary before applying them to sentence embedding tasks.

2.4.1.2 Pooling Strategies

Transformer-based models, such as BERT and its variants, are well-known for their ability to generate token-level representations, which are effective in solving token-level tasks such as Named Entity Recognition (NER) [51]. To generate sentence-level representations, it is necessary to reduce these token-level representations using pooling strategies.

The most widely studied pooling strategies in NLP include CLS, MEAN, and MAX pooling. The CLS pooling strategy leverages the CLS token, which BERT inserts at the start of each sentence, to extract the sentence embedding from the hidden state of the first token. In some methods, the CLS token embeddings are further processed by a Multi-Layer Perceptron (MLP) linear

layer [25]. MEAN pooling computes the average of all token embeddings in the final hidden layer, while MAX pooling computes the maximum activation of each feature in the final hidden layer.

MEAN pooling has proven to be the most robust method [69, 77]. However, the CLS pooling strategy is also widely used to evaluate the performance of sentence embedding methods[25, 77]. Max pooling is rarely used in modern Transformer models and is mainly used for research purposes.

2.4.2 Model Architectures

This section provides an overview of the different model architectures used for the sentence embeddings and sentence pair tasks. The focus is on describing the general architectures that have been applied to train sentence embeddings, as well as highlighting the use of other types of Transformer architectures for sentence pair tasks.

2.4.2.1 Common Architectures

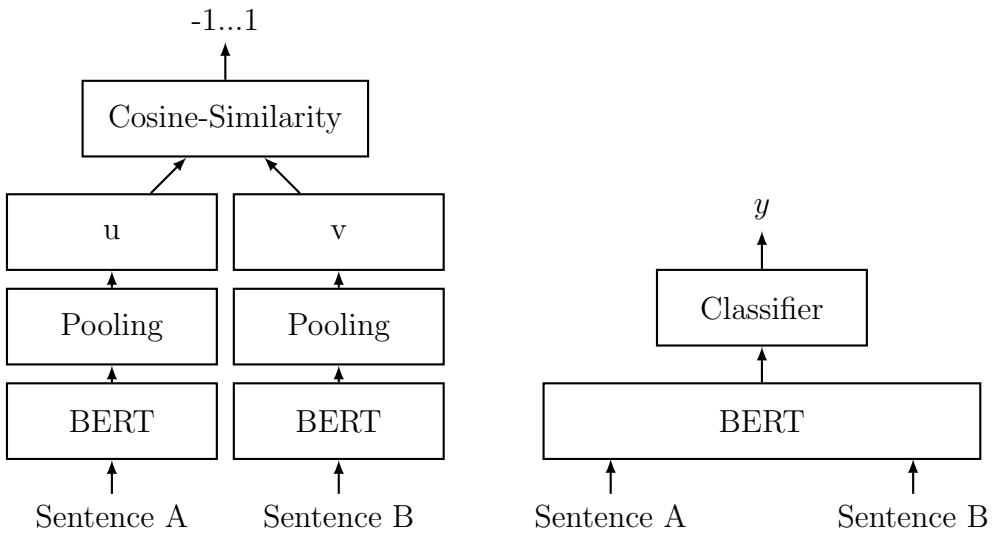
Sentence embedding tasks often use the Bi-Encoder architecture, which uses a Siamese network to encode pairs of sentences separately. The goal is to train the model to encode each sentence independently, without considering their relationship during encoding. A visual representation of the Bi-Encoder can be seen in Figure 2.4.

Transfer learning is frequently applied to Bi-Encoders to improve sentence embeddings. For example, encoders can be trained on a supervised task such as NLI by adding an interaction-based classification head to the Bi-Encoder, forcing the model to produce robust sentence embeddings. The classification head is then discarded during inference, and the encoder is used to generate only embeddings [19, 13, 69].

Another method for training sentence embeddings is contrastive learning. Unlike other methods, where sentence embeddings are projected through a linear layer, contrastive learning trains the embeddings in a more direct way [25].

In the general case of sentence pair tasks, the Cross-Encoder architecture provides a more powerful solution, but at a slower speed. This architecture finetunes the model on a sentence pair task, where both sentences are encoded within the same input sequence and separated by a special boundary token. The Cross-Encoder's self-attention mechanism allows it to consider the rela-

tionship between the sentences and make predictions via a simple classification or regression head. The illustration of the Cross-Encoder can also be found in Figure 2.4.



(a) Bi-Encoder **(b)** Cross-Encoder

Figure 2.4: Illustrations of the basic architectures.

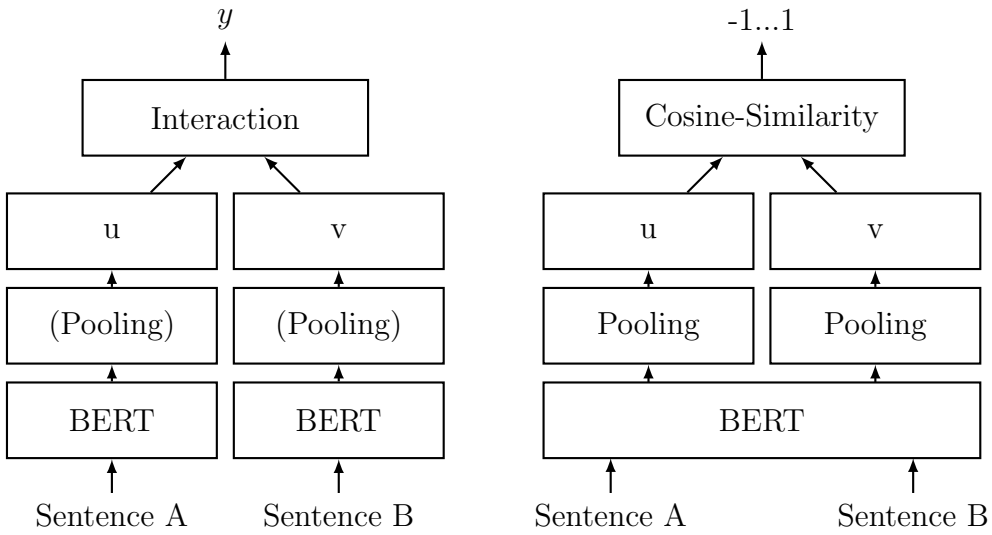
2.4.2.2 Extended Architectures

In general, for sentence pair interaction based models, three main approaches have been discussed in the literature: representation-based models, interaction-based models, and representation-interaction models [89]. Representation-based models, also known as Bi-Encoders, generate independent representations for each sentence in a pair. Interaction-based models, on the other hand, use Cross-Encoders to process the representations of both sentences in a pair in an interaction-based manner. Representation-interaction models, as the name suggests, are a combination of both approaches. These models first generate independent representations for each sentence and then use a late-interaction mechanism to compute a label based on the hidden states of each sentence.

While some literature refers to this type of encoder as a Poly-Encoder [36], we propose the term Bi-Cross-Encoder for these representation-interaction models. An example of such an architecture is ColBERT. It computes two separate embeddings for a query and a document, and then uses a "Contextualized Late Interaction" mechanism to derive a single scalar score from the two embeddings

using a maximum similarity operation between the token embeddings [39].

Another novel architecture we propose is the Cross-Bi-Encoder. This model is similar to a Cross-Encoder, with an interaction-based approach, but instead of adding a classification head on top of the hidden states, it uses a pooling approach similar to the Bi-Encoder. Two separate poolings are computed from the token embeddings and compared using cosine similarity.



(a) Bi-Cross-Encoder **(b)** Cross-Bi-Encoder

Figure 2.5: Illustrations of the extended architectures. For the Bi-Cross-Encoder, pooling is optional. The interaction-based mechanism can be done on either token-level or sentence-level embeddings.

2.4.2.3 Advantages and Disadvantages

Model architectures play a crucial role in the success of sentence pair tasks. Two main architectures that have been proposed are the Bi-Encoder and the Cross-Encoder, both of which have their own advantages and disadvantages. The choice of architecture should be based on the specific task requirements.

The Cross-Encoder architecture is considered to be more powerful and efficient than the Bi-Encoder architecture. This is because Cross-Encoders use self-attention mechanisms to directly compare both sentences and make a prediction for them, resulting in higher performance with less data and better generalization capabilities. This makes Cross-Encoders a popular choice for

various pair classification or regression tasks, especially in resource-constrained environments.

However, the main drawback of the Cross-Encoder architecture is its computational complexity during inference. To compute the similarity score between all sentences, a forward pass must be performed for each pair, resulting in $\mathcal{O}(n^2)$ complexity. This makes Cross-Encoders unsuitable for tasks that require the comparison of many sentences, as the computational requirements become infeasible.

In contrast, Bi-Encoders may not be as powerful as Cross-Encoders, but they have a simpler $\mathcal{O}(n)$ complexity, making them more suitable for certain tasks such as clustering. The Bi-Encoder computes n embeddings and calculates the cosine-similarity between all embeddings, which takes only a few milliseconds. The embeddings can also be precomputed and stored for future use. This can be useful in document retrieval tasks, where all document embeddings are computed once and only the query needs to be projected into the embedding space for real-time comparison [74].

The Bi-Cross-Encoder architecture offers a compromise between Cross-Encoders and Bi-Encoders. It generally has $\mathcal{O}(n)$ complexity and provides slightly improved performance over Bi-Encoders without significantly sacrificing performance, provided the interaction module is not overly complex, and the dataset is not too large. The Bi-Cross-Encoder is slightly more complex than the Bi-Encoder, but it offers a good balance between performance and computational complexity by allowing for the pre-computation of embeddings.

One limitation to note is that only the score function of the Bi-Encoder is guaranteed to be symmetric, making the other architectures unsuitable for tasks that require a symmetric similarity function. This property of asymmetry is only useful for certain applications [74].

In summary, the choice of architecture should be based on the specific task requirements and the trade-off between performance and computational complexity. Both Bi-Encoders and Cross-Encoders have their own advantages and disadvantages, and it is important to carefully consider these factors to make the best choice for a particular task.

2.4.3 Universal Sentence Encoder

The Universal Sentence Encoder (USE) is a robust sentence embedding model introduced by Cer et al. [13]. To the best of our knowledge, it was the first use of the Transformer architecture for training sentence embeddings and set a new benchmark for evaluating other sentence embeddings. The authors of the paper used a combination of unsupervised and supervised learning methods, training the Transformer on the Stanford Natural Language Inference (SNLI) dataset[10].

The authors used a Transformer to learn robust sentence representations for the NLI task through a Siamese architecture. The architecture projected a concatenated representation of two sentence embeddings and their interactions $(u, v, |u - v|, u * v)$ through a linear layer, which produced scores for the relevant labels of contradiction, entailment, and neutral. The network was optimized using cross-entropy loss, with the SNLI dataset providing the supervised training data.

The paper provides only a high-level overview of the method, without specifying the exact objectives used to obtain the sentence embedding model. In particular, the exact unsupervised tasks that were trained were not specified in detail. However, the architecture of the USE during training can be seen as a variant of the Bi-Cross-Encoder, as shown in Figure 2.6. During inference, the model is used as a Bi-Encoder.

The authors also tested the use of a Deep Averaging Network (DAN), but the Transformer-based approach yielded better results. Today, DANs are not commonly used in NLP research, so they will not be discussed further in this thesis.

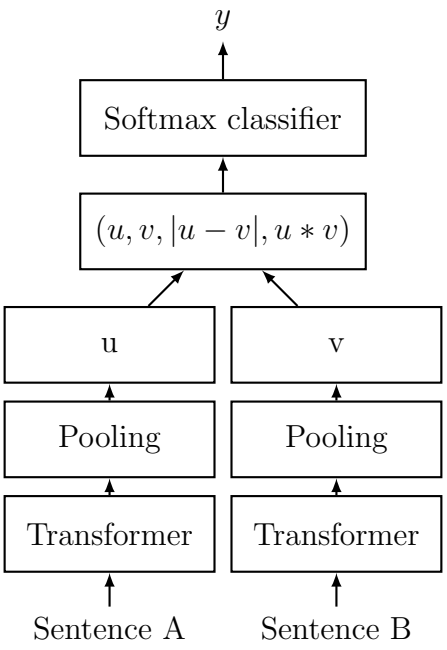


Figure 2.6: USE architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two Transformer networks have tied weights (Siamese network structure) [69].

2.4.4 Sentence-BERT

The Sentence-BERT paper [69] is a seminal paper in the field of sentence embeddings, published in 2019. The authors aimed to improve upon previous methods, such as InferSent [19] and USE [13], by leveraging the powerful representation capabilities of BERT to obtain sentence embeddings. This approach resulted in state-of-the-art performance, outperforming previous methods such as InferSent and USE.

The authors used a supervised training architecture similar to USE, but made two key modifications. First, they used a pre-trained BERT model as the starting point of their training process, rather than training a Transformer model from scratch as in USE. This allowed them to leverage BERT's pre-existing knowledge and achieve better downstream task performance. Second, they incorporated a larger NLI dataset by including the Multi-NLI dataset [80] in addition to the SNLI dataset in their training process.

To optimize the use of BERT for sentence embeddings, the authors conducted

extensive analysis to determine the best methods. They found that using simple pre-trained BERT models with naive CLS or MEAN pooling strategies performed poorly, even compared to older methods such as averaging GloVe embeddings. They compared different pooling methods and found that MEAN pooling gave the best results. Additionally, they found that using $(u, v, |u-v|)$ instead of $(u, v, |u-v|, u*v)$ for the concatenation of representations and their interactions in the training process of the NLI task resulted in improved performance, as shown in Table 2.1.

The authors evaluated their method using the SentEval [18] benchmark and achieved a new state-of-the-art performance. Due to its strong performance and thorough justification of the design choices, Sentence-BERT quickly established itself as a new baseline model in the field of sentence embeddings.

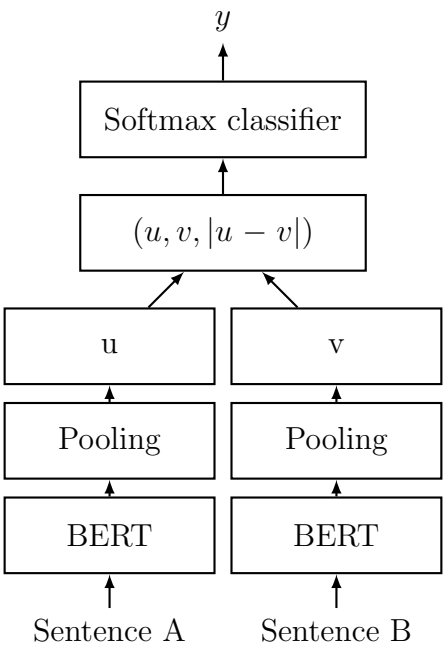


Figure 2.7: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (Siamese network structure). Adopted from Reimers et al. [69], page 3.

	NLI	**STSb**
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(u, v)	66.04	-
$(\|u - v\|)$	69.78	-
$(u * v)$	70.54	-
$(\|u - v\|, u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v, \|u - v\|)$	**80.78**	-
$(u, v, \|u - v\|, u * v)$	80.44	-

Table 2.1: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman's rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy. Adopted from Reimers et al. [69], page 7.

2.4.5 SimCSE

So far, we have only looked at supervised approaches for training sentence embeddings. In this section, we will take a look at SimCSE, short for **Sim**ple **C**ontrastive Learning of **S**entence **E**mbeddings, a state-of-the-art unsupervised method for learning sentence embeddings. We begin with a brief explanation of contrastive learning and describe how data augmentation can be used in this setting. The results of SimCSE are then discussed. Finally, we briefly summarize other closely related methods.

2.4.5.1 Contrastive Learning

Contrastive learning is a popular unsupervised method for learning sentence embeddings, and is used in the SimCSE framework. The objective of contrastive learning is to learn discriminative representations by minimizing the distance between semantically close neighbors and maximizing the distance between non-neighbors [30].

Given a set of paired examples $D = \{(x_i, x_i^+)\}_{i=1}^m$, where x_i and x_i^+ are se-

mantically related, the SimCSE method uses a cross-entropy objective with in-batch negatives [15, 33]. Let h_i and h_i^+ be the representations of x_i and x_i^+, respectively. The training objective for (x_i, x_i^+) in a mini-batch of N pairs is defined as:

$$\mathcal{L}_i = -\log \frac{\exp(\text{sim}(h_i, h_i^+))/\tau}{\sum_{j=1}^{N} \exp(\text{sim}(h_i, h_j^+))/\tau} \tag{2.7}$$

where τ is a temperature hyperparameter and the cosine similarity is defined as:

$$\text{sim}(h_1, h_2) = \frac{h_1^T h_2}{\|h_1\| \, \|h_2\|} \tag{2.8}$$

The input sentences can be encoded via a pre-trained language model such as BERT or RoBERTa as $h = m_\theta(x)$ and then fine-tuned using the contrastive learning objective above [25].

To evaluate their contrastive learning method, the authors of SimCSE propose to use the metrics of Alignment and Uniformity. These metrics evaluate, respectively, that positive instances remain close and that embeddings for random instances are scattered on the hypersphere [25].

In SimCSE, a collection of sentences $\{x_i\}_{i=1}^m$ is taken, with $x_i^+ = x_i$ and minimal augmentation in the form of dropout noise. The method can also be adapted to a supervised setting, e.g. using NLI data. Sentences with entailment labels are treated as positive pairs, while contradictions are treated as hard negative pairs. Furthermore, in-batch negative sampling generates additional negative pairs [25].

2.4.5.2 Data Augmentation for Contrastive Learning

In the field of contrastive learning, the construction of positive example pairs (x_i, x_i^+) is crucial. In computer vision, strong data augmentation methods, such as adding Gaussian noise or rotations to the input x_i, have been widely adopted [14]. However, defining appropriate data augmentation strategies for NLP is challenging. In NLP, we work with sentences, which are discrete types of data sequences. We can easily modify them, but augmentations can lead to significant changes in meaning even with small modifications.

To address this problem, SimCSE proposes a minimal form of data augmentation for NLP. The authors argue that traditional data augmentation methods, such as adding noise to sentences, can significantly damage their structure and meaning. Instead, they propose using a simple form of noise that involves

running the same sentence through two separate forward passes of a model, resulting in two different dropout masks. The positive example pair in this case is still the same sentence, with the difference between the embeddings being only slight variations in values caused by the dropout masks. This approach is considered minimal because it preserves the input sentence without applying any augmentation that changes the meaning of the sentence. [25]

2.4.5.3 Findings

The authors of SimCSE evaluated various parameters in their study, including the use of shared vs unshared weights in the Bi-Encoder and the effect of different forms of data augmentation. Regarding the construction of positive pairs, the authors compared their proposed method of using the same sentence as a positive pair with other options, such as using contextually relevant text like the next sentence or the next three sentences.

The results showed that using the same sentence as a positive pair with different dropout masks as minimal data augmentation was the best approach. The authors found that using the same dropout mask for both representations or removing it altogether led to a representation collapse, where the model overfits on the training data and becomes unable to generalize. The results also showed that other forms of augmentation that change the text lead to worse performance on the STS task, as shown in Table 2.2. Furthermore, shared weights in the Bi-Encoder performed better than unshared weights, and more sophisticated contrastive learning schemes, such as using the next sentence or the next three sentences, did not perform as well on the STS task, as shown in Table 2.3.

The SimCSE method achieved state-of-the-art results on the STS task in both supervised and unsupervised settings. The authors found that their method performed best in terms of the alignment/uniformity metrics they proposed, with the alignment remaining roughly the same while the uniformity improved throughout training. Additionally, the SimCSE method showed comparable or better transfer task performance compared to existing work, and the addition of an auxiliary MLM objective with a weight of $\lambda = 0.1$ further improved performance.

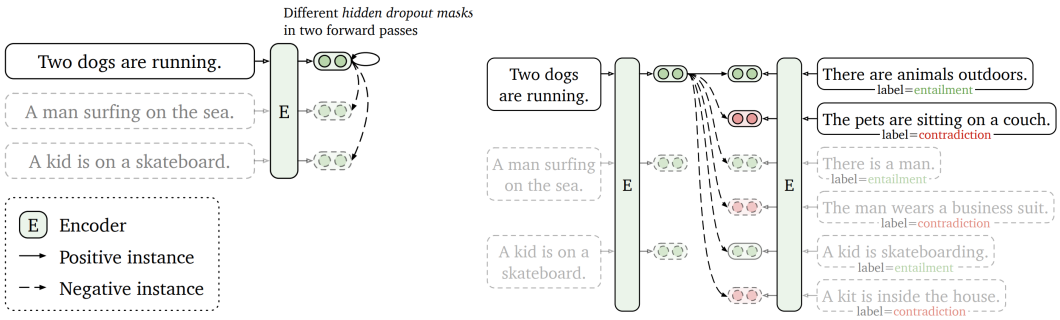
Overall, the SimCSE method demonstrated a robust theoretical background and inspired many other methods that rely on contrastive learning for learning sentence embeddings.

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

Table 2.2: Comparison of data augmentations on STS-B development set (Spearman's correlation). *Crop k%*: keep 100-*k%* of the length; *word deletion k%:* delete k% words; *Synonym replacement*: use nlpaug [56] to randomly replace one word with its synonym; *MLM k%*: use $\text{BERT}_{\text{base}}$ to replace k% of words. Adopted from Gao et al. [25], page 3.

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

Table 2.3: Comparison of different unsupervised objectives (STS-B development set, Spearman's correlation). The two columns denote whether we use one encoder or two independent encoders. *Next 3 sentences*: randomly sample one from the next 3 sentences. *Delete one word*: delete one word randomly (see Table 2.2). Adopted from Gao et al. [25], page 3.



(a) Unsupervised SimCSE **(b)** Supervised SimCSE

Figure 2.8: (a) Unsupervised SimCSE predicts the input sentence itself from in-batch negatives, with different hidden dropout masks applied. (b) Supervised SimCSE leverages the NLI datasets and takes the entailment (premise-hypothesis) pairs as positives, and contradiction pairs as well as other in-batch instances as negatives. Adopted from Gao et al. [25], page 2.

2.4.5.4 Variations

In addition to SimCSE, several other unsupervised methods for learning sentence embeddings have been proposed in the literature. For example, BERT flow [50] and BERT whitening [72] are earlier methods for creating sentence embeddings for BERT, but these methods do not rely on contrastive learning and do not produce strong results.

The supervised version of SimCSE has also been applied as a pre-training method for strong pre-trained sentence embedding models, which are available on the SentenceTransformers website[1]. These models are trained on large scale datasets, totaling over a billion sentence pairs, with the largest dataset consisting of 700 million reddit comments. In this training setup, they integrate a triplet objective, where a data example consists of an anchor, a positive example, and a negative example. The negative examples serve as hard negative examples, in addition to in-batch negative sampling.

Contrastive Tension (CT) is a closely related method to SimCSE. The main differences are that CT uses unshared weights for the Bi-Encoder and maximizes the dot product between identical sentences, instead of the cosine similarity. CT does not motivate its method as well as SimCSE and does not analyze any forms of data augmentation. They also did not study dropout noise, a crucial

[1] `https://sbert.net`

element behind the success of these methods, instead focusing on studying pre-trained models in a layer-wise fashion and their performance on STS tasks [37].

Difference-based Contrastive Learning for Sentence Embeddings (DiffCSE) is another method inspired by SimCSE. DiffCSE learns sentence embeddings that are sensitive to the difference between the original sentence and an edited sentence obtained by stochastically masking out the original sentence and then sampling from a masked language model. DiffCSE is an instance of *equivariant* contrastive learning, which generalizes contrastive learning and learns representations that are insensitive to certain types of augmentation and sensitive to other harmful types of augmentation [20]. DiffCSE outperforms vanilla SimCSE in benchmarks and achieves very close, though slightly better performance compared to SimCSE with the auxiliary MLM objective [16]. Many subsequent studies have also tried to incorporate the idea of a contrastive learning objective for learning sentence embeddings [60, 78, 81].

2.4.6 TSDAE

TSDAE (**T**ransformer-based **S**equential **D**enoising **A**uto-**E**ncoder for Unsupervised Sentence Embedding Learning) is an unsupervised training objective that focuses on learning embeddings for domain-specific tasks in addition to STS tasks. The following sections will provide an overview of the method, present their findings, and discuss some shortcomings of the study.

2.4.6.1 Overview

TSDAE is a novel unsupervised training objective for sentence embeddings, that aims to capture domain-specific information beyond the conventional Semantic Textual Similarity (STS) task. Unlike previous approaches, the authors argue that solving the STS task requires only general knowledge, and that the pre-trained knowledge obtained from training on general datasets is sufficient. However, the authors argue that this may lead to shortcomings in capturing domain-specific information.

The TSDAE architecture is based on the BART model and introduces a bottleneck in the encoder to learn sentence embeddings in a transfer-learning setting. The encoder compresses the information of a sentence into a single sentence embedding, which is then used by the decoder to reconstruct of the sentence through conditional language modeling. After training, only the encoder is used and the decoder is discarded. The authors conducted extensive studies on various hyperparameters, such as the type of data augmentation, the per-

centage of tokens to be augmented in a sentence, the type of pooling used, and whether the weights of the encoder and decoder should be tied or not, to determine the optimal configuration of hyperparameters for the TSDAE model.

The authors evaluate and compare the performance of TSDAE with state-of-the-art methods for sentence embedding representation on four datasets from heterogeneous domains. They call this benchmark USEB (Unsupervised Sentence Embedding Benchmark) and make it publicly available. The results demonstrate the effectiveness of the proposed method in learning sentence embeddings that capture domain-specific information. [77].

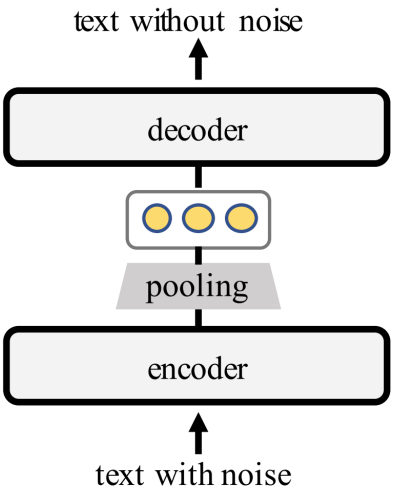


Figure 2.9: Architecture of TSDAE. Adopted from Wang et al. [77], page 2.

2.4.6.2 Findings

The authors of TSDAE found that the optimal hyperparameters for their model were deleting 60% of the tokens in the input sentence, CLS pooling, and tying the weights of the encoder and decoder. The latter serves as a form of regularization that reduces the number of parameters. The authors chose CLS pooling over MEAN pooling, even though the latter achieved higher scores, because according to them, CLS pooling "maintains position information" while MEAN pooling does not. However, both CLS and MEAN pooling allow the model to learn relevant positional information in its final embedding, since the model is trained to capture the relevant information in the pooled sentence embedding. The authors likely chose CLS pooling because it is commonly used in related methods they compared themselves to.

The authors compared the performance of TSDAE with MLM, CT, SimCSE, and Bert-flow and found that their method outperformed these state-of-the-art methods in capturing domain-specific information. The authors achieved further improvement by pre-training on the target domain and fine-tuning on the NLI and STSb datasets, which is related to the concept of domain-adaptive pre-training.

2.4.6.3 Limitations

The TSDAE approach has been shown to be effective in learning sentence embeddings that capture domain-specific information, outperforming other state-of-the-art methods. However, there is a lack of a clear motivation and theoretical foundation for what exactly TSDAE learns. This is a potential limitation in the design of the method, as it leads to an unfair comparison with other methods.

The TSDAE performs two tasks simultaneously: it adapts to the domain by self-supervised training on the denoising objective, and it learns sentence embeddings by conditioning the decoder generation on its encoder-decoder structure. This is in contrast to other methods, such as MLM, SimCSE, CT, and Bert-flow, which use only one encoder and focus on only one task, such as learning domain knowledge via a decoder head (MLM) or training sentence embeddings directly (SimCSE, CT, Bert-flow). The inclusion of an auxiliary MLM goal has been shown to further improve the performance of sentence embeddings in downstream tasks in SimCSE [25].

A fairer comparison could have been made by including a setting of MLM plus SimCSE in this study, and we will explore this setting of MLM plus SimCSE, along with measuring the impact of deletion noise in contrastive learning of sentence embeddings, when we propose our methods later.

2.4.7 Augmented SBERT

The Augmented Sentence-BERT (SBERT) is another approach that proposes both semi-supervised and unsupervised methods for sentence pair tasks. It leverages the power of the Cross-Encoder to annotate and distill unlabeled data into a Bi-Encoder, with the goal of improving its performance. In the following sections, we discuss the overview, results, and limitations of this method.

2.4.7.1 Overview

Cross-Encoders perform full-attention over input pairs, but their computational complexity makes them impractical for many use cases. Bi-Encoders, on the other hand, map input pairs independently to a dense vector space, but require a large annotated training dataset and fine-tuning over the target task to achieve competitive performance.

The Augmented SBERT combines the advantages of Cross- and Bi-Encoders by leveraging the power of the Cross-Encoder to annotate and distill unlabeled data into a Bi-Encoder. This approach can be applied in two scenarios: semi-supervised with a limited annotated dataset and unsupervised in the absence of annotated data. In the semi-supervised scenario, the method involves training a Cross-Encoder (BERT) on a small annotated gold dataset, weakly labeling new pairs with the Cross-Encoder to create a silver dataset, and training a Bi-Encoder (SBERT) on the extended dataset of gold and silver pairs. In the unsupervised scenario, the method involves training a Cross-Encoder (BERT) on a source dataset with annotations, using this Cross-Encoder to label the target dataset, and finally training a Bi-Encoder (SBERT) on the labeled target dataset [74].

This approach can also be seen as a self-training strategy [23, 49] that leverages the principle of knowledge distillation [86]. The general idea is to transfer the knowledge from a computationally expensive teacher model (the Cross-Encoder) to a computationally efficient student model (the Bi-Encoder). This idea is taken to an extreme in the Trans-Encoder, which uses unsupervised sentence pair modeling through self-and mutual-distillations between Cross-Encoders and Bi-Encoders [52].

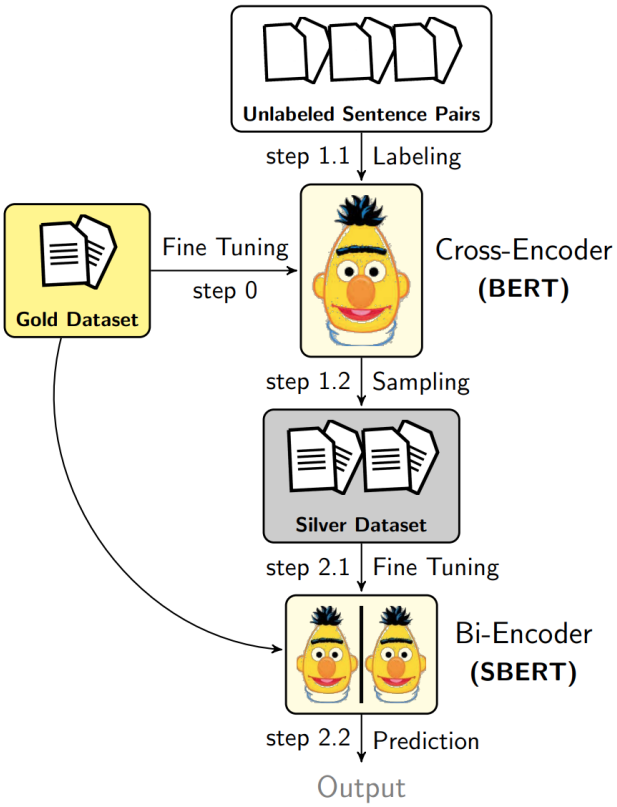


Figure 2.10: Overview of the Semi-Supervised method of the Augmented SBERT. Adopted from Thakur et al. [74], page 3.

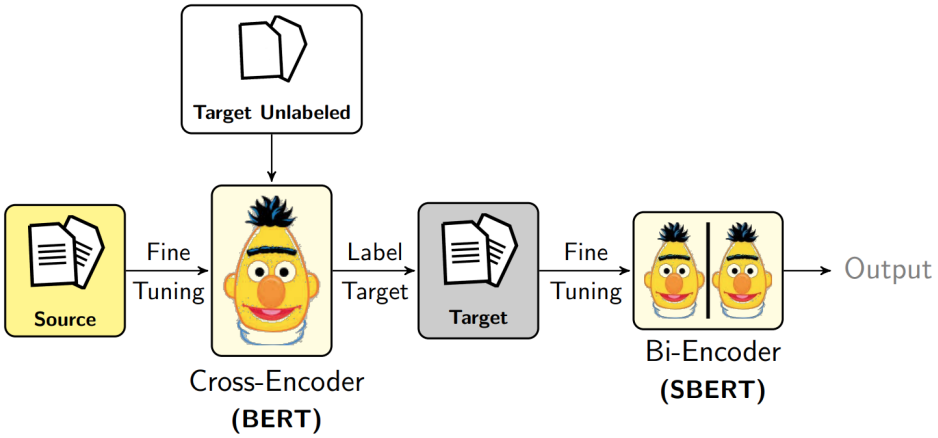


Figure 2.11: Overview of the Unsupervised method of the Augmented SBERT. Adopted from Thakur et al. [74], page 4.

2.4.7.2 Findings

The Augmented SBERT method was evaluated on four sentence pair tasks: argument similarity, semantic textual similarity, duplicate question detection, and news paraphrase identification. In the semi-supervised setting, the method showed consistent performance improvement over state-of-the-art methods that fine-tune a Bi-Encoder in a supervised manner, with improvements ranging from 1 to 6 percentage points.

In the domain adaptation scenario, the undistilled Bi-Encoder often struggled to map the new domain into a meaningful vector space, resulting in a significant drop in performance. In contrast, the Augmented SBERT method showed significant improvement, with gains of up to 37 percentage points. It should be noted, however, that the comparison was limited to an undistilled Bi-Encoder and a Bi-LSTM baseline.

The authors conducted an analysis of various sampling strategies for creating new sentence pairs from unlabeled data, highlighting the importance of the sampling strategy for the success of the method. The methods examined included random sampling, Okapi BM25, Kernel Density Estimation (KDE), Semantic Search, and a combination of BM25 and Semantic Search.

Random sampling generated predominantly negative pairs, while BM25 sampling generated mostly positive pairs. The KDE strategy aimed to match the label distribution of the silver dataset to that of the gold training set. Seman-

tic search used a trained Bi-Encoder to collect positive pairs. However, the authors were unable to reach a clear conclusion on the best sampling strategy, as no single method emerged as the clear superior choice [74].

2.4.7.3 Limitations

The Augmented SBERT approach has shown strong performance in sentence pair tasks, but there are limitations to consider. One of the limitations is the mismatch in value ranges between the Cross-Encoder and the Bi-Encoder. The Cross-Encoder projects the CLS token embedding to a score through a linear layer and a sigmoid activation function, mapping the score to the range $(0, 1)$. The Bi-Encoder, on the other hand, uses cosine similarity, which is defined in the range [-1, 1]. We argue that this difference can cause problems during distillation, as we will show later. To mitigate this, we propose to use our previously introduced Cross-Bi-Encoder architecture the standard Cross-Encoder, since it also uses cosine similarity to compute its scores, similar to the Bi-Encoder.

Another limitation is the "Kernel Density Estimation" sampling strategy used to obtain new sentence pairs from unlabeled data. This strategy involves randomly sampling a large number of pairs and then using a trained Cross-Encoder to annotate them. The method then uses kernel density estimation to estimate the continuous density functions for the scores and attempts to minimize the KL Divergence [42] between the distributions using a sampling function $Q(s)$.

$$Q(s) = \begin{cases} 1 & \text{if } F_{gold}(s) \geq F_{silver}(s) \\ \frac{F_{gold}(s)}{F_{silver}(s)} & \text{if } F_{gold}(s) < F_{silver}(s) \end{cases} \tag{2.9}$$

The sampling function defined by the equation 2.9 adjusts the probability of retaining a sample based on the comparison of the density functions of the gold and silver data sets. However, this approach can be computationally inefficient because it discards many samples. In addition, as can be seen in Figure 2.12, the KDE strategy clearly fails to approximate the target distribution. We argue that the reason for this is that the large number of negative examples vastly outnumbers the positive examples when simple random sampling is applied, rendering their reduced-probability sampling approach ineffective. To address this, we will later propose a modified iterative approach that achieves the goal of matching the target gold distribution.

Additionally, during our own experiments, we discovered a potential problem

with the implementation of the "SemEval Spanish STS" dataset described by the authors. They recommended normalizing the scores by 5 to get scores in the range $[0, 1]$. However, this normalization only holds true for the test split, which indeed had scores in the range $[0, 5]$. For the training split, the correct range is $[0, 4]$, which would require dividing the scores by 4 instead. We have taken care to correct this in our implementation.

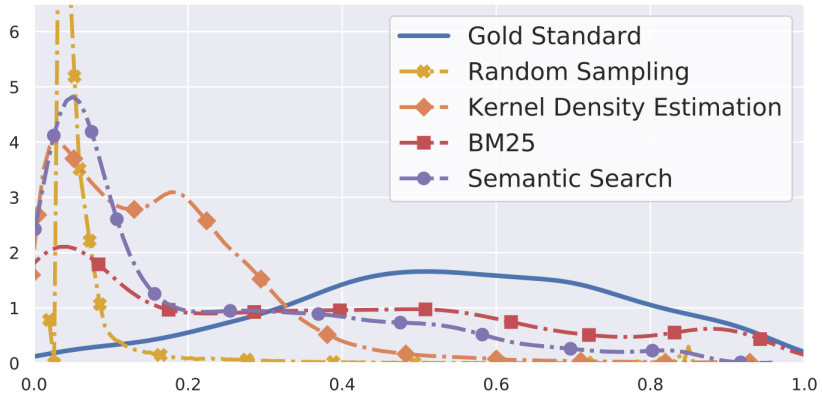


Figure 2.12: Comparison of the density distributions of gold standard with silver standard for various sampling techniques on Spanish-STS (in-domain) dataset. Adopted from Thakur et al. [74], page 8.

Chapter 3

Datasets

In this chapter, we look at the datasets we use to train all of our models. We introduce each dataset and explain how the data was collected and annotated.

3.1 SentEval

SentEval is a popular evaluation toolkit that provides various tasks to evaluate the performance of sentence embeddings, including classification, natural language inference, and semantic similarity tasks [18]. In this study, we evaluate our models on the semantic textual similarity (STS) tasks, which is consistent with comparable work in the field [69, 25]. Specifically, we use the STS tasks from the 2012 to 2016 SemEval workshops [4, 5, 2, 1, 3], the STS benchmark [12], and the SICK-relatedness dataset [57]. These datasets provide a finely annotated gold standard for semantic similarity, with scores ranging from 0 to 5 for each sentence pair, which we normalize to a range of $[0, 1]$ by dividing each score by 5.

Most of these datasets were published as part of the SemEval workshop, which is an international NLP research initiative aimed at advancing the state of the art in semantic analysis and producing high-quality annotated datasets. Each year's workshop consists of common tasks that evaluate and compare computational semantic analysis systems[1].

We provide details on the characteristics of the relevant datasets in tables 3.1, 3.2, and 3.3. The tables specify the size of each split for each dataset, and we provide further details on the composition of the STS-B dataset, as it will be used extensively in our experiments.

[1] `https://semeval.github.io/`

Dataset	#Train	#Dev	#Test	#Total
STS12	2234	-	3108	5342
STS13	-	-	1500	1500
STS14	-	-	3750	3750
STS15	-	-	3000	3000
STS16	-	-	1186	1186
STS-B	5749	1500	1379	8628
SICKR	-	-	9927	9927

Table 3.1: An overview of the STS datasets we use for our evaluation [64].

Genre	File	Years	#Train	#Dev	#Test
news	MSRpar	2012	1000	250	250
news	headlines	2013-16	1999	250	250
news	deft-news	2014	300	-	-
captions	MSRvid	2012	1000	250	250
captions	images	2014-15	1000	250	250
captions	track5.en-en	2017	-	125	125
forum	deft-forum	2014	450	-	-
forum	answers-forum	2015	-	375	-
forum	answer-answer	2016	-	-	254

Table 3.2: An overview of the STS-B dataset [12].

Genre	#Train	#Dev	#Test	#Total
news	3299	500	500	4299
caption	2000	625	625	3250
forum	450	375	254	1079
All	**5749**	**1500**	**1379**	**8628**

Table 3.3: An overview of the train/dev/test distribution of the STS-B dataset [12].

3.2 Unsupervised Sentence Embedding Benchmark

The Unsupervised Sentence Embedding Benchmark (USEB) is a small benchmark aimed at evaluating the performance of sentence embedding representations across four datasets from heterogeneous domains. The benchmark was

introduced in the TSDAE paper [77] to measure the ability of a method to learn sentence embeddings that capture domain-specific information. The tasks included in the USEB are Re-Ranking (RR), Information Retrieval (IR), and Paraphrase Identification (PI).

The datasets used in the USEB are AskUbuntu (RR task) [47], CQADupStack (IR task) [34], TwitterPara (PI task) [83, 44], and SciDocs (RR task) [17]. In detail, AskUbuntu is a collection of user posts from the AskUbuntu technical forum, where models are required to re-rank 20 candidate questions based on similarity to an input post. The evaluation metric used is Mean Average Precision (MAP). CQADupStack is a question retrieval dataset of forum posts from 12 different forums including Android, English, and Mathematica, among others. Models are required to retrieve duplicate questions from a large candidate pool and the metric used is MAP@100. TwitterPara consists of two similar datasets, the Twitter Paraphrase Corpus (PIT2015) [83] and the Twitter News URL Corpus (TURL) [44], which are collections of annotated tweet pairs with a score indicating whether they are paraphrases. The evaluation metric used is Average Precision (AP) over both the gold confidence scores and the similarity scores from the models. SciDocs is a benchmark consisting of multiple tasks related to scientific papers, and for the USEB, the tasks of Cite, Co-Cite, Co-Read, and Co-View are used. Models are required to identify relevant papers from up to 30 candidates given one query paper title, and the evaluation metric is MAP [77].

For evaluation, sentences are encoded into fixed-sized vectors using cosine similarity for sentence similarity. Only the titles from the AskUbuntu, CQADupStack, and SciDocs datasets are used in the USEB. For datasets with subdatasets or sub-tasks, such as CQADupStack, TwitterPara, and SciDocs, the final score is derived by averaging the scores from each sub-dataset or sub-task. For unsupervised training, the flat sentences from the training split without any labels are used [77]. The statistics for each dataset are shown in Table 3.4.

Dataset	Task	#queries	Avg. #relevant	Avg. #candidates	Avg. length	Size of unsupervised training set	Size of supervised training set
AskUbuntu	RR	200	5.9/5.4	20	9.2	165K	23K
CQADupStack	IR	3K	1.1/1.1	39K	8.6	44K	13K
SciDocs	RR	4K	5	30	12.5	312K	380K

Dataset	Task	#paraphrase	#non-paraphrase	Avg. length	Size of unsupervised training set	Size of supervised training set
TwitterPara	PI	–/2K	–/9K	13.9	53K	23K

Table 3.4: Dataset statistics. The slash symbol '/' separates the numbers for development and test. Multiple subdatasets are included in CQADupStack, SciDocs and TwitterPara. CQADupStack has one sub-dataset for each of the 12 forums. The avg. #relevant, avg. #candidates and avg. length are all general statistics without distinguishing the sub-datasets. Adopted from Wang et al. [77], page 5.

3.3 Spanish STS Corpus

We follow the approach of Thakur et al. [74] in creating a SemEval Spanish STS Corpus. The train and validation splits are composed of sentence pairs from the Spanish STS datasets provided by the SemEval 2014 [2] and SemEval 2015 [1] workshops. As no default validation split was provided, we randomly selected 220 sentence pairs for validation. These sentence pairs were collected from Spanish news articles and Wikipedia articles. Additionally, the STS14 dataset includes the Li65 trial dataset, which contains 65 sentence pairs annotated by three native Spanish speakers.

The goal of the STS task is to measure the similarity between two sentences and assign a score in the range of $[0, 4]$, where 4 represents a high degree of similarity (i.e., paraphrases) and 0 indicates no relation between the sentences. The scores are then normalized to the range of $[0, 1]$ by dividing each score by 4.

The test split consists of sentence pairs from the SemEval STS 2017 dataset, which were extracted from image caption pairs in the SNLI dataset [10]. The similarity scores for these sentence pairs were also normalized to the range of $[0, 1]$ by dividing each score by 5.

The use of the Spanish STS dataset in our study allows us to evaluate our

sentence embedding models on a language other than English, which allows for a comparison with the results presented in Thakur et al. [74]. An overview of the dataset and its components can be found in table 3.5.

Year	**Domain**	**Train**	**Dev**	**Test**	**Total**
2014	Speakers	65	-	-	65
2014	News	480	-	-	480
2014	Wikipedia	324	-	-	324
2015	Newswire	500	-	-	500
2015	Wikipedia	251	-	-	251
2017	Image caption	-	-	250	250
Combined	-	**1420**	**220**	**250**	**1870**

Table 3.5: An overview of the Spanish STS datasets we use for our experiments [64].

3.4 UKP Sentential Argument Mining Corpus

The UKP Sentential Argument Mining Corpus is a dataset released by the Ubiquitous Knowledge Processing Lab that contains 25,492 annotated sentences from across 8 topics. While it is not used as an argument mining dataset in this study, we use it for its topics, and it also serves as the basis for another dataset that we will introduce later in the chapter.

Other argument extraction approaches often rely on consistent text types, making it impractical to extract arguments from the web [21]. The UKP Sentential Argument Mining Corpus addresses this problem by restricting topics to those that can be implicitly expressed by keywords, and defining an argument as a span of text expressing evidence or reasoning that can be used to support or oppose a given topic. This restriction limits the complexity of arguments, but allows for a wider range of text types to be handled.

The corpus was created by selecting eight controversial topics and collecting the top 50 Google results for each topic, including a variety of text types such as news reports, editorials, blogs, debate forums, and encyclopedia articles. The collected documents underwent standard preprocessing using tools such as Apache Tika and Stanford CoreNLP [71]. The distribution of topics across their splits can be observed in the Table 3.6.

Topic	#Train	#Dev	#Test	#Total
Abortion	2827	787	315	3929
Death penalty	2627	731	293	3651
Nuclear energy	2573	717	286	3576
Gun control	2404	669	268	3341
Cloning	2187	609	243	3039
School uniforms	2165	602	241	3008
Marijuana legal.	1780	497	198	2475
Minimum wage	1778	497	198	2473
Combined	**18341**	**2042**	**5109**	**25492**

Table 3.6: An overview of the UKP Sentential Argument Mining dataset.

3.5 BWS Argument Similarity Corpus

The BWS Argument Similarity Corpus (BWS) is a dataset specifically designed for the evaluation of argument similarity. It was created by Thakur et al., as part of their study of the Augmented SBERT [74]. It extends the sentences collected for the UKP Sentential Argument Mining Corpus [71] with annotations for argument similarity. Unlike previous work on argument similarity [62, 70], the BWS dataset uses continuous annotations to provide a more nuanced and accurate representation of argument similarity.

The annotations were collected using the Best-Worst Scaling (BWS) method [40] and a comparative approach. For each topic, regardless of stance, all arguments were randomly paired and filtered using a distant supervision strategy [62] to ensure a certain proportion of similar arguments within pairs. The argument pairs were then sampled based on a desired similarity distribution and annotated through crowdsourcing on Amazon Mechanical Turk. The quality of the annotations was assessed using split-half reliability measures [11], yielding an average correlation of 0.66 across all topics, which reflects the difficulty of the task and is acceptable given their optimized annotation strategy.

In the experiments of the Augmented SBERT model, the BWS dataset is used with different splitting strategies. In cross-topic tasks, topics T1-T5 are used for training, T6 for development, and T7 and T8 for testing. In in-topic tasks, argument pairs are randomly sampled and split into training, development, and test sets with an equal number of pairs from each topic. The BWS dataset allows them to evaluate the performance of the models on a continuous scale and on unseen topics in cross-topic tasks. We will follow their approach

and use it for our experiments as well.

Topic T	**Score**	**Topic** T	**Score**
Cloning	0.84	Nuclear energy	0.64
Abortion	0.79	Death penalty	0.58
Minimum wage	0.50	Gun control	0.59
Marijuana legal.	0.57	School uniforms	0.64
Whole dataset = 0.66			

Table 3.7: Mean split-half reliability estimate is calculated using Spearman's rank correlation ρ per topic T and over the whole BWS Argument Similarity dataset. Adopted from Thakur et al. [74], page 6.

3.6 20 Newsgroups Dataset

The 20 Newsgroups dataset is a widely-used collection of approximately 20,000 newsgroup documents, evenly distributed across 20 different newsgroups. The dataset was originally collected by Ken Lang for his paper "Newsweeder: Learning to filter netnews", although this is not explicitly stated[46][2]. The 20 Newsgroups collection has become a popular benchmark for evaluating text classification and text clustering algorithms in machine learning.

The 20 different newsgroups correspond to various topics, ranging from closely related subjects (e.g., comp.sys.ibm.pc.hardware / comp.sys.mac.hardware) to highly unrelated subjects (e.g., misc.forsale / soc.religion.christian). Additionally, six hierarchical labels were extracted and used in addition to the fine-grained labels. We use the original test split, but use a test split that is about 10% of the total dataset size, which is taken from the original train split. A detailed overview of all labels for the 20 Newsgroups can be seen in Tables 3.8 and 3.9. We will use this dataset in our experiments to measure how well we can integrate topic information into sentence embeddings.

To maintain the generalizability of the results, it is crucial to remove headers, signature blocks, and quotations from each news article, following the recommended practice outlined by the Scikit-learn library[3]. Without this preprocessing step, it is possible for a classifier to overfit on specific elements that appear in the 20 Newsgroups dataset, such as newsgroup headers, leading to

[2] `http://qwone.com/~jason/20Newsgroups/`

[3] `https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html#filtering-text-for-more-realistic-training`

results that do not generalize to other documents.

Topic	**#Train**	**#Dev**	**#Test**	**#Total**
alt.atheism	382	98	319	799
comp.graphics	488	96	389	973
comp.os.ms-windows.misc	494	97	394	985
comp.sys.ibm.pc.hardware	486	104	392	982
comp.sys.mac.hardware	472	106	385	963
comp.windows.x	496	97	395	988
misc.forsale	494	91	390	975
rec.autos	499	95	396	990
rec.motorcycles	503	95	398	996
rec.sport.baseball	491	106	397	994
rec.sport.hockey	493	107	399	999
sci.crypt	511	84	396	991
sci.electronics	506	85	393	984
sci.med	496	98	396	990
sci.space	503	90	394	987
soc.religion.christian	509	90	398	997
talk.politics.guns	436	110	364	910
talk.politics.mideast	468	96	376	940
talk.politics.misc	389	76	310	775
talk.religion.misc	312	65	251	628
all	9428	1886	7532	18846

Table 3.8: All topics contained in the 20 Newsgroups corpus, with their amount of examples per split.

Computer	**Recreational**	**Science**
comp.graphics	rec.autos	sci.crypt
comp.os.ms-windows.misc	rec.motorcycles	sci.electronics
comp.sys.ibm.pc.hardware	rec.sport.baseball	sci.med
comp.sys.mac.hardware	rec.sport.hockey	sci.space
comp.windows.x		
For Sale	**Politics**	**Religion**
misc.forsale	talk.politics.misc	talk.religion.misc
	talk.politics.guns	alt.atheism
	talk.politics.mideast	soc.religion.christian

Table 3.9: All topics contained in the 20 Newsgroups corpus, with their according to overarching topics.

Chapter 4

Methods

In this chapter, we present our innovative approaches to improving sentence embeddings. First, we introduce the Cross-Bi-Encoder architecture, which addresses the shortcomings of existing Cross-Encoder models in distillation settings. We then review the TSDAE method, perform an analysis of its learning objectives, and present a new training method called Noise Matching (NM). Finally, we propose a new approach for incorporating topic information into sentence embeddings. Our methods aim to address the challenges faced by current approaches and provide a deeper understanding of how they work.

4.1 Cross-Bi-Encoder

In this section, we present our proposed Cross-Bi-Encoder architecture. Our aim is to address the limitations of current Cross-Encoder models in distillation settings, such as those used in the Augmented SBERT method discussed in section 4.3. To motivate this, we first provide a mathematical background of the relevant concepts, followed by an overview of the Cross-Encoder, Bi-Encoder and Cross-Bi-Encoder architectures. Finally, we will discuss the proposed advantages of our Cross-Bi-Encoder over current Cross-Encoder models.

4.1.1 Mathematical Background

In this section, we present the mathematical background relevant to the sentence pair architectures. To better understand the predictions made by these architectures, we first introduce the cosine similarity, sigmoid activation function, and linear layer formulas.
The cosine similarity between two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as:

$$sim(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}|\,|\mathbf{v}|} \tag{4.1}$$

where $\cdot$ is the dot product and $|\cdot|$ is the magnitude (or Euclidean length) of the vector. The cosine similarity ranges from -1 to 1, with values close to 1 indicating high similarity and values close to -1 indicating high dissimilarity.
The sigmoid activation function is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

where e is the base of the natural logarithm and x is the input to the activation function. The sigmoid function maps any input value to the range (0, 1), making it useful for modeling binary outcomes or probabilities.
A linear layer in a neural network performs a simple linear transformation on its input. Given an input vector $\mathbf{x}$ of size d and a weight matrix $\mathbf{W}$ of size $d \times m$, where m is the number of output units in the layer, a linear layer computes an output vector $\mathbf{y}$ of size m such that:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{4.3}$$

where $\mathbf{b}$ is a bias vector of size m. The weight matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$ are learnable parameters of the layer that are updated during training to minimize a loss function.
With this theoretical background, we can try to identify potential issues with the underlying computations and predictions made by the different architectures.

4.1.2 Score Predictions

We now describe in detail how each architecture obtains its predictions. We will focus on a simple regression setting.

Cross-Encoder

The Cross-Encoder model, denoted as m_{CE}, maps an input sentence pair (x_1, x_2) to a similarity score $s \in (0, 1)$ due to its sigmoid activation function. It consists of the following steps:

1. **Tokenization**: The input sentence pair (x_1, x_2) is tokenized into the sequence $t = [\text{CLS}](...)_{x_1}[\text{SEP}](...)_{x_2}[\text{SEP}]$.

2. **BERT Encoding**: The tokenized representation t is passed through the BERT model m_{BERT}, which maps each token to a hidden state h_i in the last hidden layer.

3. **Pooling**: A pooling operation is applied to derive a representation for the sentence. In the case of a Cross-Encoder, this is simply the [CLS] token embedding $\mathbf{h_0} \in \mathbb{R}^d$, where d is the hidden size of the model.

4. **Classifier**: The classifier function c_{CE} is applied, which projects the token embedding $\mathbf{h_0}$ to a score $s \in \mathbb{R}$ using a linear layer as in Equation 4.3.

5. **Sigmoid Activation**: The score s is then passed through a sigmoid activation layer as in Equation 4.2.

Bi-Encoder

The Bi-Encoder model, denoted as m_{BE}, maps an input sentence pair (x_1, x_2) to a cosine similarity score $s \in [-1, 1]$. It consists of the following steps:

1. **Tokenization**: The input sentence pair (x_1, x_2) is tokenized into two separate sequences $t_u = \text{[CLS]}(...)_{x_1}\text{[SEP]}$ and $t_v = \text{[CLS]}(...)_{x_2}\text{[SEP]}$.

2. **BERT Encoding**: Each tokenized representation t is passed separately through the BERT model m_{BERT}, which maps each token to a hidden state h_i^t in the last hidden layer.

3. **Pooling**: In both cases, a pooling operation is applied. In the case of the Bi-Encoder, the most popular choice is to apply a MEAN operation over all hidden states in $\mathbf{h^{t_u}}$ and $\mathbf{h^{t_v}}$, resulting in two embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$.

4. **Cosine Similarity**: A cosine similarity function is applied to $\mathbf{u}$ and $\mathbf{v}$ as defined in Equation 4.1.

Cross-Bi-Encoder

The Cross-Bi-Encoder model, denoted as m_{CBE}, maps an input sentence pair (x_1, x_2) to a cosine similarity score $s \in [-1, 1]$. It consists of the following steps:

1. **Tokenization**: The input sentence pair (x_1, x_2) is tokenized into the sequence $t = \text{[CLS]}(...)_{x_1}\text{[SEP][CLS]}(...)_{x_2}\text{[SEP]}$. Note that there is an extra [CLS] token.

2. **BERT Encoding**: The tokenized representation t is passed through the BERT model m_{BERT}, which maps each token to a hidden state $\mathbf{h_i}$ in the last hidden layer.

3. **Pooling**: A pooling operation is applied to derive two representations of the sentences. We suggest applying MEAN pooling from the first [CLS] token to the first [SEP] token, and from the second [CLS] token to the second [SEP] token. This mimics the pooling procedure of the Bi-Encoder while allowing for interaction between both sentences, resulting in two embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$.

4. **Cosine Similarity**: A cosine similarity function is applied to $\mathbf{u}$ and $\mathbf{v}$ as defined in Equation 4.1.

In conclusion, the Cross-Encoder, Bi-Encoder, and Cross-Bi-Encoder models each have a unique procedure for obtaining similarity scores.

4.1.3 Proposed Advantages

The proposed advantages of the Cross-Bi-Encoder over the Cross-Encoder are as follows:

1. **Consistent Scoring Metric**: Both the Cross-Bi-Encoder and the Bi-Encoder use cosine similarity as their scoring metric, which is defined in the range $[-1, 1]$. In contrast, the standard Cross-Encoder uses a sigmoid function, which produces scores in the range $(0, 1)$. This is particularly problematic in situations where negative sentence pairs are annotated, which were not part of the training data. Typically, negative pairs are trained to have scores close to 0, resulting in orthogonal vectors. However, during inference, the embedding space in the Bi-Encoder can sometimes yield slightly negative cosine similarities for negative pairs. Due to the limited value range of the sigmoid function, the standard Cross-Encoder may not be able to express the full range of negative sentence pair scores, resulting in a mismatch with the scoring function of the Bi-Encoder. The Cross-Bi-Encoder, on the other hand, is expected to have similar performance to the Cross-Encoder, but with better alignment to the scoring function of the Bi-Encoder. This could be beneficial in a teacher-student setup, such as the Augmented SBERT.

2. **Equal Feature Weighting**: Both the Cross-Bi-Encoder and Bi-Encoder weight each feature equally for their predictions, as cosine similarity only

considers the orientation of the vectors and not their magnitude. In contrast, the Cross-Encoder's linear projection uses a weight matrix, resulting in a score derived from a weighted sum. This may not be ideal for transferring knowledge across tasks or domains, as different features in the output embeddings may be more relevant for different tasks. The weighted linear projection of the Cross-Encoder may result in a distorted, biased view. The equal weighting of all dimensions in the cosine similarity allows the Cross-Bi-Encoder to produce better annotation scores in domain-adaptation settings.

4.2 TSDAE Analysis

In this section, we delve into a deeper analysis of the TSDAE method, which was introduced in Section 2.4.6. Our goal is to gain a comprehensive understanding of the method's learning objectives. To this end, we first examine the deletion noise in the TSDAE method, and then discuss its training objectives. Finally, we propose a new training method called Noise Matching (NM), which aims to achieve similar objectives to the TSDAE.

4.2.1 Deletion Noise

The TSDAE method, as previously discussed in Section 2.4.6, uses a denoising approach to learn sentence embeddings by making the task more difficult. This approach, however, may have introduced an unintended side effect: invariance towards deletion noise.

Invariance is a mathematical property that describes the unchanged behavior of a system under certain transformations. In mathematical terms, if a function or a set of equations describes a system and remains unchanged after a transformation, the function or the set of equations is considered invariant under that transformation. Mathematically, this can be expressed as $f(T(x)) = f(x)$, where $T(x)$ is the noising function (i.e., deletion of 60% of the tokens in x) and f describes our model. This property would mean that a sentence and its variant (after deletion noise has been applied) would have equal embeddings.

However, the model may not necessarily learn strict invariance, but it may assign too high of a similarity for the deleted variants of sentences. This can be expressed as

$$sim(f(T(x)), f(x)) > sim(f(\tilde{x}), f(x)) > sim(f(T(x)), f(\tilde{x})) \tag{4.4}$$

where $\tilde{x}$ is a sentence with identical meaning but with synonyms instead. In an ideal sentence embedding model, we would want the order

$$sim(f(\tilde{x}), f(x)) > sim(f(T(x)), f(x)) > sim(f(T(x)), f(\tilde{x})) \tag{4.5}$$

To illustrate this, we present an example where we encode variations of the same sentence from the train dataset of a TSDAE model[1]. The base sentence is "problems with keyboard shortcuts in gnome 3.10", with a synonym variation of "issues with keyboard hotkeys in gnome 3.10." and a deletion variation of "problems with in gnome 3.". As shown in Table 4.1, the ordering follows equation 4.4, not equation 4.5. This observation is not a rigorous proof, but we will further investigate the property of deletion noise invariance in another architecture and compare the evaluation scores with the TSDAE.

Sentence	**Base**	**Synonyms**	**Deletion**
Base	100%	77.08%	77.61%
Synonyms	77.08%	100%	63.64%
Deletion	77.61%	63.64%	100%

Table 4.1: An example, indicating the TSDAE's learning objective might make it insensitive towards deletion noise. Similarity scores are cosine similarities.

4.2.2 Objective Breakdown

The TSDAE method has two primary learning objectives that enable it to learn sentence embeddings. First, it learns a language modeling objective through its conditional language modeling head in the decoder. This teaches the model to reconstruct the entire sentence in an autoregressive manner, making it a more challenging task compared to other methods such as SimCSE or CT, which only learn sentence embeddings directly. Second, the TSDAE passes the pooled sentence embedding to the decoder through cross-attention, which forces the sentence embeddings to effectively capture the information of the entire sequence. This is a crucial difference from other methods that use only an encoder and focus solely on either learning domain knowledge via a decoder head (e.g., MLM) or training sentence embeddings directly. The added difficulty in the TSDAE's language modeling objective could result in stronger generalization capabilities, especially for smaller datasets where it is easy to overfit on the training data.

[1] We used the publicly available kwang2049/TSDAE-askubuntu model from HuggingFace.

To make a fair comparison with the TSDAE, it would be beneficial to consider alternative architectures with similar goals. For example, the addition of an auxiliary MLM objective has been shown to improve the performance of sentence embeddings in downstream tasks in SimCSE [25]. In the next section, we will outline potential alternative architectures and introduce a new objective, Noise Matching (NM), that could match the performance of the TSDAE.

4.2.3 Noise Matching

In this section, we introduce a new training method, Noise Matching (NM), which aims to achieve similar objectives as the TSDAE method. The goal of this method is to replicate the aspect of the TSDAE method where 60% of the input sentence is noised, but with a different approach inspired by the SimCSE method introduced in Section 2.4.5.

Naively applying the SimCSE method and defining $x^+ = T(x)$, where $T(x)$ is the deletion noise, would not result in a strong objective. This is because $T(x)$ and x are not semantically related, and it is not desirable to think of them as positive pairs. The high amount of deletion noise in $T(x)$ makes it a subsequence of x, not a semantically related sequence.

To clarify, a subsequence is a sequence obtained from another sequence by deleting some or no elements without changing the order of the remaining elements. In other words, let $X = (x_i)_{i=1}^n$ be a sequence of n elements. A sequence $Y = (y_j)_{j=1}^m$ is a subsequence of X if there exists a strictly increasing sequence of indices $(i_j)_{j=1}^m$ from X such that for all $j \in 1, ..., m$, $y_j = x_{i_j}$ [29].

Given this, the model should learn that $x^+ = T(x)$ is similar to x, since all tokens contained in $T(x)$ are also contained in x. However, it should not learn that x is similar to $T(x)$, since x is neither equal to nor is it a subsequence of $T(x)$.

To achieve this objective, we implement a stop-gradient operator (sg) on the hidden representation h_i of the sentence x_i, allowing only the gradient to flow backwards through the original sentence. This way, the model will only learn to be similar to the original sentence, not the noised one. An illustration of this approach can be seen in Figure 4.1.

Finally, we will evaluate the new objective, NM, by itself, in combination with

MLM, and in combination with the TSDAE objective. The goal is to compare the performance of the NM objective with that of the TSDAE method and to gain a comprehensive understanding of its learning objectives.

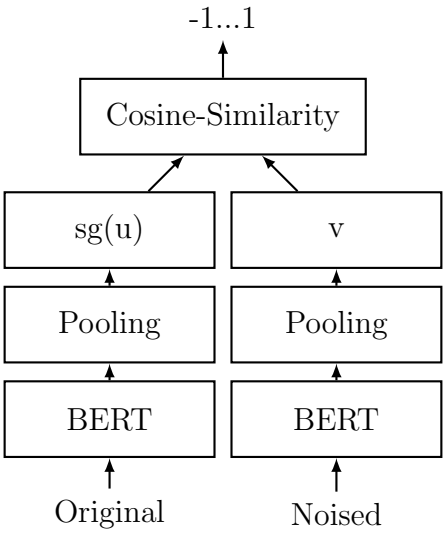


Figure 4.1: Bi-Encoder architecture for the NM objective, where sg is the stop gradient operator.

4.3 Improved Augmented SBERT

In this section, we propose further improvements to the Augmented SBERT method that go beyond the introduction of the Cross-Bi-Encoder 4.1. Our goal is to improve its performance and address some of the limitations of the approach. To this end, we explore how domain adaptation can be used to improve the unsupervised method in unfamiliar domains. In addition, we present an improved iterative algorithm for sampling sentence pairs using the KDE sampling strategy in the semi-supervised setting.

4.3.1 Domain Adaptation

In the unsupervised setting, the Cross-Encoder is trained on a closely related gold dataset to annotate the target dataset. However, this approach may be insufficient if the Cross-Encoder is not familiar with the target domain. To address this issue, we propose to apply the idea of domain-adaptive pre-training we introduced earlier in Section 2.1.3.2, which involves continuing the original pre-training objective on the target domain prior to fine-tuning.

In this thesis, we use Masked Language Modeling (MLM) pre-training to adapt the Cross-Encoders and extract domain knowledge. To ensure a smooth transition from pre-training to the downstream task, we format the input as pairwise tokenization:

$$[\text{CLS}](...)x_1[\text{SEP}][\text{CLS}](...)x_2[\text{SEP}]$$

where $(...)x_1$ and $(...)x_2$ represent the two tokenized sentences.

Another crucial aspect is that the sentence pairs annotated by the Cross-Bi Encoder are the same as those used for MLM pre-training, which allows the model to acquire relevant knowledge more easily. Before fine-tuning the model in the downstream task, we freeze the embeddings to avoid moving too far away from what we learned in pre-training. However, it is important to consider the trade-off between allowing the model to adapt to its new task and retaining information from its pre-training domain. This trade-off must be carefully weighed when deciding which layers to freeze, as freezing multiple layers can mitigate the risk of catastrophic forgetting [41], but may also affect performance on the target task.

4.3.2 Iterative KDE Algorithm

The Augmented SBERT method [74], as discussed in the previous Section 2.10, has limitations in its kernel density estimation (KDE) algorithm. The method uses KDE to estimate the continuous density functions of the scores and minimizes the KL Divergence [42] between the distributions using a reduced-probability sampling approach. However, this can be computationally inefficient and fail to accurately approximate the target distribution due to the large number of negative examples that vastly outnumber the positive examples.

To overcome these limitations, we propose the iterative KDE algorithm. Given a set of gold scores and a set of silver scores, this algorithm returns a list of indices corresponding to the silver scores. The algorithm first computes the gold KDE using the gold scores and then shuffles the silver scores and selects the first two indices as the initial silver dataset. At each iteration, the algorithm compares the density of each remaining (index, score) pair in the silver scores to both the gold KDE and the silver KDE. If the gold density is greater than the silver density, the index is added to the silver dataset, the score is removed from the silver scores, and the silver KDE is recomputed using the updated silver dataset. The loop continues until convergence, defined by some

stopping criterion. The algorithm returns the final silver dataset.

The proposed iterative KDE algorithm is more computationally efficient because it retains only those samples that are most likely to be positive examples. It is also more accurate because it iteratively updates the silver KDE based on the comparison to the gold KDE, resulting in a better approximation of the target distribution with a hard cutoff instead of a sampling function. This improved approach offers advantages over the original KDE procedure described in the Augmented SBERT method [74], resulting in a more reliable and efficient density estimation method.

Algorithm 1 Iterative KDE

Require: Gold scores **gold_scores**, Silver scores **silver_scores**
Ensure: List of indices **silver_dataset**

```
function ITERATIVE KDE
    Calculate gold KDE on gold_scores
    Shuffle silver scores
    Select first two indices of silver scores as silver_dataset
    Remove first two scores from silver_scores
    Calculate silver KDE on silver_dataset
    while not converged do
        for each remaining silver (index, score) pair do
            Compare its density to both gold and silver KDEs
            if gold density is greater than silver density then
                Add the index to silver_dataset
                Remove the score from silver_scores
                Recalculate silver KDE on silver_dataset
            end if
        end for
    end while
    return silver_dataset
end function
```

4.4 Topic Information in Embeddings

We will now present our final method, a supervised approach that aims to study the effect of integrating topic information into sentence embeddings. We will first outline the aim of our study, before specifying how we will use our topic information in training, and before presenting our approach to effectively learn sentence embeddings.

4.4.1 Goal

The study of supervised sentence embeddings often involves the fine-tuning of a Bi-Encoder model on a sentence pair dataset. However, this alone is not a novel research question, as the process of fine-tuning encoder models is well established. An interesting area of research would be to experiment with building generalized models using supervised data, but this would require large datasets and significant computational resources, making it infeasible for this work.

Given these limitations, we propose a variant of Augmented SBERT for learning sentence embedding, guided by labelled topic data. We use a flat dataset, where each example is annotated with a topic label, in addition to a small publicly available sentence similarity dataset. Our goal is to train a sentence embedding model in a self-supervised, domain-adaptive setting that incorporates the desired topic information while providing meaningful sentence similarity scores. This approach could be useful in situations where a dataset with known topics has unknown subtopics. Generating sentence embeddings that incorporate topic information to achieve inter-topic embedding separation while maintaining good sentence similarity scores within clusters could be an interesting use case.

To the best of our knowledge, there is limited prior work in this area. Our approach attempts to incorporate topic information in an unsupervised manner to produce topic-sensitive sentence embeddings. Although there is some vaguely related work [26], the main related work is simply related to prompts in general [53].

4.4.2 Prompted Topic Information

In our proposed approach, we aim to incorporate the desired topic information into sentence embeddings by means of a "topic prompt". For a given sentence x, which is typically tokenized as "[CLS](...)$_x$[SEP]", we add a label text t_y and a delimiter token $[T]$, which marks the end of the topic label text t_y. The resulting input format is "[CLS](...)t_y[T](...)$_x$[SEP]". This allows the model to learn from the labeled data in a self-supervised manner, rather than simply incorporating the topics through a standard supervised loss function, where the topic label is usually learned through a simple classification head and cross-entropy loss. The use of a delimiter token ensures that the model always knows where the topic starts and ends. In this thesis, we use $[T] = ¥$ as the topic delimiter, since this token is not used in our datasets. By using a topic prompt,

our approach tries to implicitly guide the process of generating topic-sensitive sentence embeddings.

4.4.3 Distilling Topic Information

In this section, we present our approach to incorporating topic information into sentence embeddings through a modification of the unsupervised Augmented SBERT pipeline. Our goal is to pre-train the model using masked language modeling, allowing it to learn from the topic information prompt and the sentence pair input format, before fine-tuning it on a sentence similarity downstream task.

We propose a specific tokenization format for our pre-training objective, consisting of a topic information prompt and a sentence pair input format. The tokenization format is constructed as follows:

$$[\text{CLS}](...)t_{y_1}[\text{T}](...)_{x_1}[\text{SEP}][\text{CLS}](...)t_{y_2}[\text{T}](...)_{x_2}[\text{SEP}]$$

where t_y represents the topic information prompt, $(...)x_1$ and $(...)x_2$ represent the two sentences, and [T] is the delimiter token that marks the end of the topic label text.

In the pre-training stage, we use masked language modeling, but carefully avoid masking the topic information t_y in our MLM noising function. This allows the model to learn from the topic information prompt. After adapting the model to the new domain and integrating domain knowledge, we apply the unsupervised Augmented SBERT pipeline to further improve the performance of the model, as outlined in Section 4.3.1. The results of this approach are discussed in detail in the Experiments section.

Chapter 5

Experiments

In this chapter, we describe our experimental setups to evaluate the performance of our proposed methods. First, we present the hyperparameters that we use for most of our experiments, then we present the studies that motivate our Cross-Bi-Encoder architecture. This is followed by a detailed examination of all sampling strategies used for the Augmented SBERT [74], including our newly proposed KDE strategy. We then outline our unsupervised learning experiments, which focus on the newly proposed TSDAE objectives. We then turn our attention to the semi-supervised learning case, where all of our proposed improvements to Augmented SBERT are applied. Finally, we examine the supervised setting, which aims to integrate topic information into sentence embeddings.

5.1 Hyperparameters

Fine-tuning pre-trained language models, such as BERT, on a downstream task can result in varying performance scores across different random seeds, as reported in previous studies [35, 63]. This variability is particularly pronounced when working with larger models or smaller datasets [22]. To address this issue and ensure stability in our experiments, we carefully select the hyperparameters for fine-tuning.

Following the recommendations of the BERT authors, we set our learning rate to 2e-5 and include a warm-up period of 10% of the total training time, followed by a linear decay. The AdamW optimizer is used with a weight decay of 0.01, and our batch size is set to 16, unless otherwise specified. The BERT model refers to bert-base-uncased, and the RoBERTa model refers to roberta-base. All other hyperparameters are set to their default values as provided

by the Huggingface library[1]. To further reduce variability in training, we apply seed optimization in certain settings and average our results over multiple seeds.

To evaluate our models, we use the Spearman metric to measure the correlation for regression problems and the F1 macro score for classification problems. After each epoch, we evaluate our models and take the model with the best Spearman or F1 score at the end. Both metrics are multiplied by 100 to obtain a human-readable score in the range [0, 100].

In summary, our careful selection of hyperparameters and the use of seed optimization and averaging of results contribute to the stability and robustness of our experimental results.

5.2 Cross-Bi-Encoder

In this section, we evaluate the performance of our proposed Cross-Bi-Encoder architecture through a series of experiments. Our experiments are conducted on either the BWS Argument Similarity corpus, the STS Benchmark dataset, or SentEval, all of which were introduced in Chapter 3. All experiments use the hyperparameters specified in the previous section. We first present basic experiments to introduce each architecture, followed by experiments using distillation.

5.2.1 Basic Experiments

In the first experiment, we evaluate the performance of the Bi-Encoder and Cross-Encoder architectures on the STS-B test split using the BERT model. Both models are fine-tuned for three epochs on the STS-B train split.

To address the limitations of the Cross-Encoder's output range, we experiment with different activation functions, including *tanh* with a range of $(-1, 1)$ and no activation function with an unlimited range. The performance is measured on the STS-B test split and the distribution of scores is analyzed to determine the best-performing model. All models are trained for three epochs.

Subsequently, we evaluate the performance of our proposed Cross-Bi-Encoder on the STS-B test split, comparing it to the Bi-Encoder and Cross-Encoder.

[1] https://huggingface.co/docs/transformers/master/en/main_classes/trainer

The correlation between the scores produced by the Cross-Encoder, Cross-Bi-Encoder, and Bi-Encoder is also studied. The experiments are conducted using both the BERT and RoBERTa models, which are fine-tuned for three epochs each.

5.2.2 Distillation Experiments

In this set of experiments, we investigate the impact of distillation on the performance of Bi-Encoder models. We train our models on both BERT and RoBERTa. We first conduct a distillation experiment on the STS-B dataset by training a Cross-Encoder and Cross-Bi-Encoder model, and then using their predictions to re-annotate the STS-B train split. The re-annotated datasets are distilled into Bi-Encoders, which acts as a form of label regularization [86]. A baseline Bi-Encoder is also trained on the STS-B dataset for comparison. All models are trained for three epochs.

The evaluation is extended to the BWS dataset by performing a similar distillation experiment. The Cross-Encoder and Cross-Bi-Encoder models are trained on the BWS dataset and used to re-annotate the scores, which are distilled into Bi-Encoders. A baseline Bi-Encoder is trained on the original BWS train dataset. To evaluate the ability of the Bi-Encoders to encode topic information, the encoder weights are frozen, and a classification head is added to the sentence embedding. The classifier is trained on the topics of the BWS argument similarity dataset for ten epochs with an increased learning rate of 1e-3, and the F1 macro score is determined on the test split. Additionally, the trained Bi-Encoders make predictions on the SentEval STS tasks. Our metrics are computed by taking the Spearman correlation between the predictions and the original labels. The models are trained for five epochs each.

Finally, we replicate the previous distillation experiment and simulate an unsupervised domain adaptation task by training the Cross-Encoder and Cross-Bi-Encoder models on the STS-B dataset and using their predictions to annotate the similarity scores for the BWS dataset, which are distilled into Bi-Encoders. A baseline Bi-Encoder is also trained on the original BWS train dataset. The same evaluation metrics are used as in the previous experiment. All models are trained for five epochs.

Note that there is a small detail in our implementation here, as we directly predicted and compared the scores on the individual STS splits, instead of using the recommended SentEval toolkit. The subsequent sections are not affected by this, as we use SentEval for those.

In this set of experiments, we explore the use of distillation to regularize the training of Bi-Encoder models. We first conduct a distillation experiment on the STS-B dataset by training a Cross-Encoder and Cross-Bi-Encoder model, and then using them to re-annotate the scores of the STS-B train split. The re-annotated scores are then distilled into a Bi-Encoder, which acts as a form of label regularization [86], while a baseline Bi-Encoder is also trained on the STS-B dataset. All models are trained for three epochs.

5.3 Sentence Pair Sampling

In this section, we evaluate the performance of different sentence pair sampling strategies used in the Augmented SBERT model. Our experiments extend the analysis from the original paper, which evaluated only the Spanish STS dataset, to the STS-B dataset. We use bert-base-multilingual-cased and BERT, respectively. Additionally, we annotate sentence pairs using the Cross-Bi-Encoder architecture.

We compare three sentence pair sampling methods that generate similar sentence pairs: BM25, Semantic Search using a pre-trained model, and Semantic Search using a fine-tuned model. These experiments provide insights into the effectiveness of each sampling strategy.

We then evaluate the impact of candidate sets on the performance of our proposed iterative KDE algorithm by conducting experiments with various configurations. The candidate set can be generated using random pairs, pairs obtained through Semantic Search, or a combination of both.

Finally, we summarize the best sentence pair sampling methods for the Augmented SBERT model, which will inform the design of our subsequent unsupervised and semi-supervised learning experiments.

5.4 TSDAE Experiments

In this section, we present our TSDAE experiments. Our experiments include a replication of the original experiments, our novel contrastive learning objectives, and an evaluation of our improved Augmented SBERT. We also compare our results with the best pre-trained models listed on the SentenceTransform-

ers library website.[2]

5.4.1 Replication Study

We present a replication study of the original TSDAE setup as described in the original paper. The experiments in the TSDAE paper were performed on the USEB dataset, which was introduced in Chapter 3. The authors used 100K training steps with a batch size of 8, the AdamW optimizer with a constant learning rate of 3e-5, no warm-up period, and a weight decay of 0.0. The CLS pooling method was used, the token deletion rate was set to 60%, and the encoder and decoder weights were tied. The base model was BERT.

Our goal is to verify the results of the original study by replicating the experimental setup described by the authors. Unfortunately, only some parts of the TSDAE have been released to the public, and the original experiments conducted as part of the study were not included.

5.4.2 Contrastive Experiments

In this section, we investigate the performance of different unsupervised learning methods in the context of sentence embeddings through contrastive experiments. We compare the settings of SimCSE, MLM + SimCSE, Noise Matching (NM), MLM + NM, and TSDAE + NM, using the MEAN and CLS pooling methods.

To ensure a fair comparison, all methods use BERT and are trained for 12.5K steps with a batch size of 64 and a learning rate of 3e-5. The remaining default hyperparameters described in Section 5.1. Our approach combines SimCSE and NM as low-loss auxiliary objectives that guide the direction of learning when combined with MLM or TSDAE. This is to avoid making too many changes compared to the original TSDAE objective, so we can better understand the performance and learning objectives of these methods.

We find that a batch size of 8, used in the original TSDAE experiments, made the contrastive learning objectives too easy. Hence, we use a batch size of 64 to maintain sufficient task difficulty for the contrastive objectives.

Through these experiments, our goal is to gain a comprehensive understanding of the learning objectives and performance of these methods.

[2] `https://sbert.net/docs/pretrained_models.html`

5.4.3 Augmented SBERT

Since the previous architectures were not built with the goal of outperforming TSDAE, we also want to test our improved Augmented SBERT pipeline in this unsupervised learning setting. Our approach aims to improve performance by incorporating the cross-bi encoder and domain adaptation through MLM pre-training. We use the base model RoBERTa.

To perform domain adaptation, we first perform 12.5K steps of MLM pre-training with a batch size of 64 and a learning rate of 3e-5 on sentence pairs sampled from our training dataset, consisting of both positive and negative pairs. After MLM pre-training, we freeze the embedding weights and fine-tune our Cross-Bi-Encoder on the STS-B dataset for 5 epochs. The sentence pairs from the original MLM pre-training are then annotated using the fine-tuned Cross-Bi-Encoder, resulting in the annotated silver dataset. This dataset is then used to fine-tune our Bi-Encoder for an additional 12.5K steps, with a batch size of 64.

To further analyze the impact of our approach, we perform an ablation study. We train a total of three Bi-Encoders: one starting from the original RoBERTa checkpoint (DomainSBERT), one starting from the checkpoint after the extra MLM pre-training (FullDomainSBERT), and one with a Cross-Bi-Encoder that omits the MLM objective (AugSBERT).

Finally, we evaluate all trained Bi-Encoders on the corresponding test split of the USEB to assess their performance.

5.5 Semi-Supervised Augmented SBERT

In this section, we evaluate the performance of our proposed improvements to the semi-supervised Augmented SBERT method [74]. Our aim is to demonstrate that the use of our Cross-Bi-Encoder and iterative KDE algorithm lead to improved performance compared to the original method.

We conduct experiments on three datasets: STS Spanish, BWS Argument Similarity (both in-topic and cross-topic settings), using the RoBERTa model. The dataset details are summarized in Table 5.1.

Our Cross-Encoder and Cross-Bi-Encoder models are used as teacher models in a data augmentation setting. The train datasets are reannotated using our trained Cross-Encoders as a form of label regularization, following the ap-

proach of Yuan et al. [86]. The goal is to construct a silver dataset from new sentence pairs sampled from a gold dataset, which are then annotated by the teacher model.

We evaluate five different sampling strategies: no sampling, random sampling, Okapi BM25, Kernel Density Estimation (KDE), and Semantic Search. For the KDE strategy, we use our iterative KDE algorithm to sample its sentence pairs from a mix of positive and negative examples.

We will later demonstrate the effectiveness of our proposed improvements in improving the performance of the semi-supervised Augmented SBERT method.

Dataset	**Spanish-STS**	**BWS (cross-topic)**	**BWS (in-topic)**
#train	1400	2125	2471
#dev	220	425	478
#test	250	850	451
#total	**1870**	**3400**	**3400**

Table 5.1: Summary of all datasets being used for diverse in-domain sentence pair tasks in this paper

5.6 Distilling Topic Information

In this section, we aim to evaluate the effectiveness of incorporating topic information into sentence embeddings for the purpose of learning embeddings that are sensitive to specific topics while retaining good performance on sentence similarity tasks. To achieve this, we present our training setup and describe in detail the use and evaluation of each dataset. Our experiments are designed to provide insights into the impact of topic information on sentence embeddings.

5.6.1 Training Setup

In this section, we describe the training setup for evaluating the effectiveness of incorporating topic information into sentence embeddings. As outlined in Section 4.4, our approach leverages the combination of self-supervised training and the Augmented SBERT pipeline to extract sentence embeddings that are sensitive to specific topics while retaining good performance on sentence similarity tasks. To evaluate this, we use the RoBERTa model and two datasets: the UKP Sentential Argument Mining dataset and the 20 Newsgroup dataset.

From the standard UKP Sentential Argument Mining dataset, we created a

pair dataset containing 73364 sentence pairs, each with 2 positive and 2 negative sentence pairs. The sentence pairs were obtained from semantic search and random search, and one of the 8 argument topics was used as a label for each sentence in the pair. For the 20 Newsgroups dataset, we created a pair dataset containing 75424 sentence pairs, each with 4 positive and 4 negative sentence pairs, obtained from semantic search and random search. The parent groups of the 20 Newsgroups dataset were used as the topic labels for each sentence in the pair, with the idea of investigating the effect of this on learning information about the actual groups in the dataset.

Our experiments compare two settings: one where each sentence pair is trained without any topic information, and another where each sentence in the pair is prepended with an additional topic label.

First, we establish a baseline method using a simple classification objective, where a classifier head is used to classify the topic label.

Next, we use the Augmented SBERT pipeline to distill multiple versions of Bi-Encoders. The distillation process starts with MLM training on RoBERTa using all the approximately 75K sentence pairs for 5 epochs with a batch size of 64 and a learning rate of 3e-5. Then, we train on the STS-B dataset for 5 epochs to obtain our Cross-Bi-Encoders. For the Bi-Encoders, we then use either the MLM pre-training (FullDomainSBERT) or the RoBERTa model (DomainSBERT) as a starting point.

In the setting without the topic information prompt, the distillation process is straightforward. A score s is obtained for each sentence pair (x_1, x_2), resulting in (x_1, x_2, s), which is then used to distill the Bi-Encoder. In the topic information prompt setting, the Cross-Bi-Encoder produces a score s for each pair $((t_{y_1}, x_1), (t_{y_2}, x_2))$, resulting in $(t_{y_1}, x_1, t_{y_2}, x_2, s)$, which is used to train the first Bi-Encoder. The second Bi-Encoder is trained on (x_1, x_2, s), which represents the hidden prompt setting. Both Bi-Encoders are trained for 5 epochs with a batch size of 64.

The results of these experiments are discussed in detail in the Experiments section.

5.6.2 Evaluation

For the 20 Newsgroups dataset, we evaluate the effectiveness of our training procedures through two experiments:

1. A frozen-encoder classification task, where we train a classifier on the original 20 topics and evaluate its performance using the F1 macro metric. In this task, we evaluate both topic information prompt settings.

2. A SentEval benchmark evaluation, which measures the overall performance of the sentence embeddings on sentence similarity tasks.

For the UKP Sentential Argument Mining dataset, we run three evaluations:

1. A frozen-encoder classification task, where we train a classifier on the 8 topics and evaluate its performance using the F1 macro metric. In this task, we evaluate both topic information prompt settings.

2. A SentEval benchmark evaluation, which measures the overall performance of the sentence embeddings on sentence similarity tasks.

3. A BWS Argument Similarity dataset evaluation, which measures how well our approach indirectly distills the argument similarity measure in each setting.

The frozen-encoder classification task assesses the ability of the sentence embeddings to capture the topic information. The F1 macro score provides a balanced measure of the classifier's performance across all topics. The SentEval evaluation measures the overall performance of the sentence embeddings on sentence similarity tasks. The BWS Argument Similarity dataset evaluation evaluates the ability of our approach to distill the measure of argument similarity in each setting.

Chapter 6

Evaluation

In this chapter, we list and discuss all of our results for the experiments described in chapter 5. In the following sections, we will use the abbreviations CBE for Cross-Bi-Encoder, CE for Cross-Encoder, and BE for Bi-Encoder. For regression tasks, we report Spearman's rank correlation as $\rho \times 100$ unless otherwise noted, and for classification tasks, we report the F1 macro score.

6.1 Cross-Bi-Encoder Results

In this section, we will look at simple performance scores on the STS-B test dataset for all our most essential architectures.

6.1.1 Basic Experiments

Method	**STS-B**
Bi-Encoder	83.77 ± 0.23
Cross-Encoder	85.33 ± 0.79

Table 6.1: Spearman correlation of Bi-Encoder and normal Cross-Encoder on STS-B.

In Table 6.1 we can see the performance of the two basic architectures, BE and CE, on the STS-B test split after finetuning on the STS-B train split. The CE delivers slightly better scores than the BE.

Method	**STS-B (spearman)**	**STS-B (pearson)**
Cross-Encoder No Activation	85.33 ± 0.79	83.91 ± 1.00
Cross-Encoder Sigmoid	85.33 ± 0.79	86.52 ± 0.65
Cross-Encoder Tanh	85.33 ± 0.79	85.04 ± 0.79

Table 6.2: Spearman/Pearson correlation of Cross-Encoder with different activations on STS-B.

Next in Table 6.2, we wanted to analyze whether using the sigmoid activation function for the CE made sense, as we suggested that it might be a source of potential discrepancies when distilling into the BE. However, the sigmoid achieves the best performance on the Pearson similarity, while the other options struggle to map their predictions to a reasonable range. The tanh range maps too many predictions into a negative range, as tanh is essentially a rescaled logistic sigmoid function. The CE without activation function, on the other hand, has a range of values far beyond what was in the dataset, making it unsuitable for predictions. An illustration of the activation functions on the test splits, can be seen in Figure 6.1.

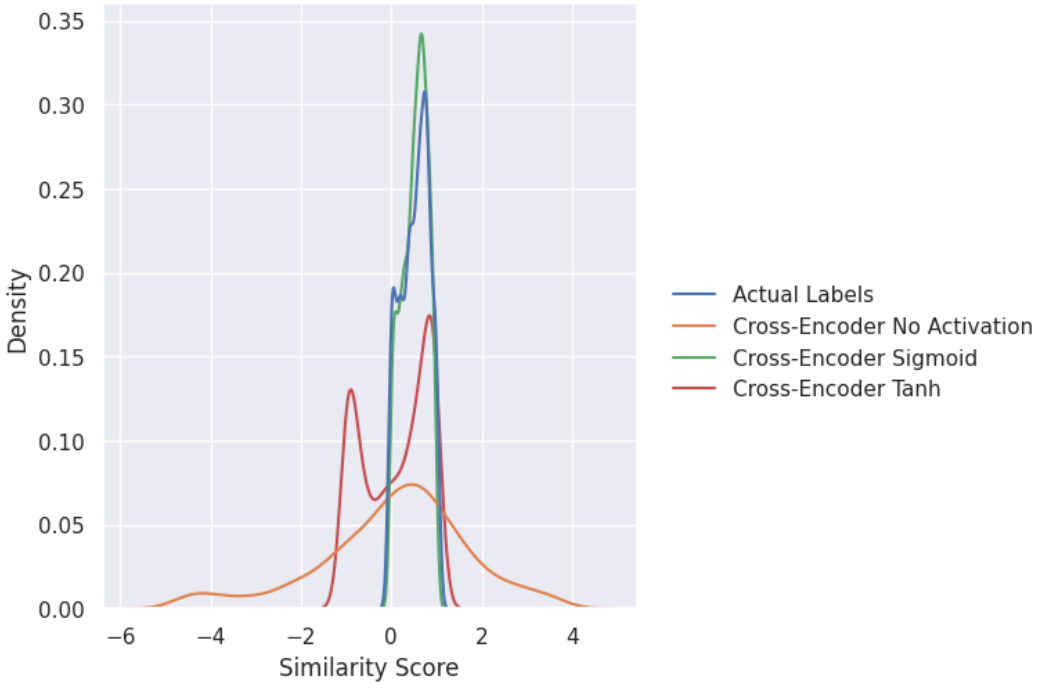


Figure 6.1: Kernel density estimation of the different activation functions.

Method	STS-B (Original)	STS-B (Bi-Encoder)
BERT Bi-Encoder	83.77 ± 0.23	100.00 ± 0.00
BERT Cross-Bi-Encoder	85.43 ± 0.35	93.91 ± 0.10
BERT Cross-Encoder	85.33 ± 0.79	92.25 ± 0.22
RoBERTa Bi-Encoder	85.60 ± 0.28	100.00 ± 0.00
RoBERTa Cross-Bi-Encoder	88.83 ± 0.24	92.76 ± 0.04
RoBERTa Cross-Encoder	88.66 ± 0.23	90.90 ± 0.57

Table 6.3: Spearman correlation of all models on STS-B test with BERT.

As a last basic experiment, we also present the CBE in Table 6.3, along with performance comparisons with the RoBERTa model. Here we can see that the RoBERTa model generally seems to outperform the BERT model, at least on the STS-B task. The CBE and the CE seem to have quite similar scores. However, the CBE has a higher correlation with the BE scores, which is probably due to their common scoring function, the cosine similarity.

6.1.2 Distillation Experiments

In the following, we will describe our distillation experiments on the STS-B and BWS datasets.

Method	STS-B
BERT Base	83.77 ± 0.23
BERT Distill CBE	83.91 ± 0.25
BERT Distill CE	83.77 ± 0.11
RoBERTa Base	85.60 ± 0.28
RoBERTa Distill CBE	85.62 ± 0.30
RoBERTa Distill CE	85.49 ± 0.14

Table 6.4: Spearman correlation of all distilled models on STS-B test with BERT/RoBERTa.

In the Table 6.4 we can see a simple distillation example where we let the CE and CBE architecture re-annotate their train splits after training on them. Essentially, we are performing a self-distillation process to the BE. This is typically used as a form of label regularization, as described by Yuan et al. [86]. We do not observe much difference in score between the architectures. RoBERTa again outperforms BERT.

Method	BWS Similarity (ρ)	BWS Topic (F1)
BERT Base	64.27 ± 0.61	93.95 ± 0.08
BERT Distill CBE	65.49 ± 1.14	93.82 ± 0.04
BERT Distill CE	64.93 ± 0.61	94.00 ± 0.25
RoBERTa Base	64.66 ± 0.98	93.58 ± 0.54
RoBERTa Distill CBE	64.40 ± 0.73	93.59 ± 0.36
RoBERTa Distill CE	65.16 ± 0.35	93.32 ± 0.24

Table 6.5: Spearman correlation of all BWS to BWS distilled models on BWS test with BERT/RoBERTa.

Next, instead of looking at the STS-B dataset, Table 6.5 we look at the BWS dataset again. As a reminder, it contains argument similarities. Here we see

mixed results in the distilled BEs, but no conclusions can be drawn. Additionally, we do not see any outliers in terms of BWS Topic F1 score.

Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
BERT Base	43.37	67.17	55.37	66.78	70.30	59.09	64.21	60.9
BERT Distill CBE	42.84	67.95	55.10	67.47	71.12	59.52	63.35	61.05
BERT Distill CE	44.63	66.45	55.08	67.56	70.30	58.94	63.70	60.95
RoBERTa Base	53.07	72.37	62.43	72.44	73.48	71.58	67.33	67.53
RoBERTa Distill CBE	53.25	71.97	62.66	72.53	73.23	71.85	67.28	67.54
RoBERTa Distill CE	54.82	72.21	63.11	71.97	73.96	72.28	68.21	68.08

Table 6.6: Spearman correlation of all BWS to BWS distilled models on SentEval benchmarks with BERT/RoBERTa.

Looking at the STS performance of the BEs trained on the BWS datasets in table 6.6, we can see that the BE (distill from CE) seems to have an advantage of about 1% points over the other architectures. However, we believe that this is an anomaly, as we will see in the next table.

Method	BWS Similarity (ρ)	BWS Topic (F1)
BERT Base	64.27 ± 0.61	93.95 ± 0.08
BERT Distill CBE	61.12 ± 0.77	93.39 ± 0.06
BERT Distill CE	57.03 ± 0.78	93.99 ± 0.13
RoBERTa Base	64.66 ± 0.98	93.58 ± 0.54
RoBERTa Distill CBE	61.93 ± 1.02	93.94 ± 0.37
RoBERTa Distill CE	56.37 ± 4.43	94.03 ± 0.64

Table 6.7: Spearman correlation of all STS-B to BWS distilled models on BWS test with BERT/RoBERTa.

In our next setup, as can be seen in Table 6.7, we instead trained our CE and CBE again on the STS-B dataset, and then used them to generate scores for the BWS training dataset. We essentially simulate a domain transfer here, as we train on one task, semantic similarity, but instead annotate for argument similarity. As a baseline for an upper performance limit, we trained the regular BE on the default BWS dataset. Nevertheless, a comparison of the BE (distilled from CE) and the BE (distilled from CBE) clearly shows that the

CBE outperforms the CE by up to 5% points. However, again, we do not see any outliers in terms of BWS Topic F1 score.

Method	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**STS-B**	**SICK-R**	**Avg.**
BERT Base	43.37	67.17	55.37	66.78	70.30	59.09	64.21	60.9
BERT Distill CBE	46.93	69.85	58.80	69.99	70.52	63.54	64.80	63.49
BERT Distill CE	46.39	68.15	57.30	68.46	70.46	63.52	63.72	62.57
RoBERTa Base	53.07	72.37	62.43	72.44	73.48	71.58	67.33	67.53
RoBERTa Distill CBE	55.25	75.28	65.83	74.87	75.21	72.80	70.04	69.9
RoBERTa Distill CE	51.05	70.74	62.52	72.68	74.12	70.94	68.88	67.28

Table 6.8: Spearman correlation of all STS-B to BWS distilled models on SentEval benchmarks with BERT/RoBERTa.

This reasoning also applies to the performance on the STS tasks in Table 6.8. For RoBERTa, the BE (distilled from CBE) achieves STS scores, which are roughly 2.5 points higher than the rest. We believe that these results alongside the results in Table 6.7 show the potential of the CBE for achieving improvements in distillation settings.

6.2 Sentence Pair Sampling Results

In this section, we will analyze some distributions of pair samplings strategies across different datasets and model architectures.

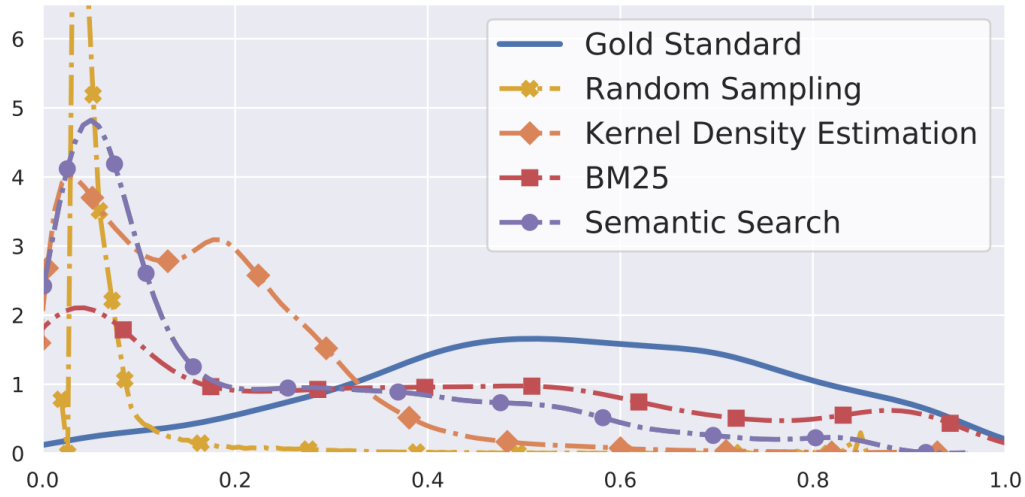
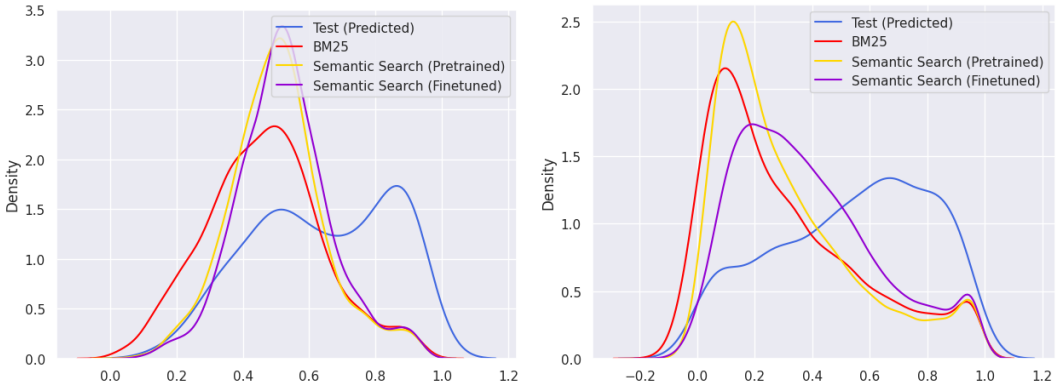


Figure 6.2: Original comparison of the density distributions of gold standard with silver standard for various sampling techniques on Spanish-STS (in-domain) dataset by Thakur et al. [74], page 8.

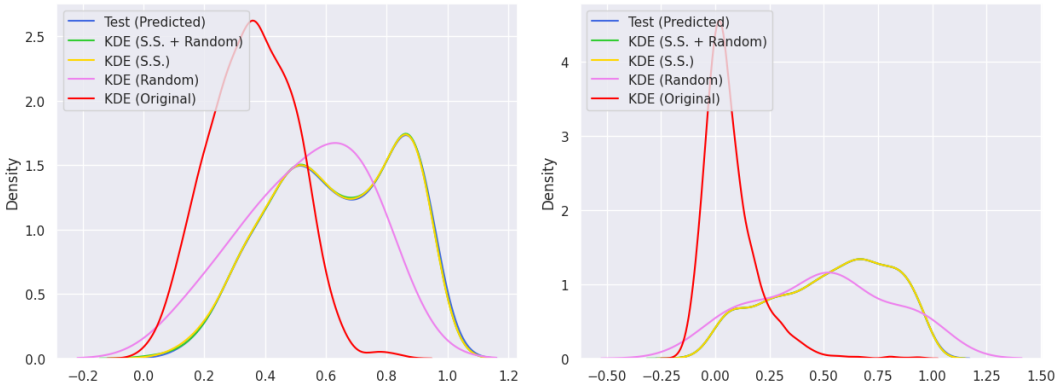
The original distributions obtained by Thakur et al. in Figure 6.2 seem to have some problems. We are not sure how they arrived at their results, but their proposed semantic search method seems to perform worse than BM25. So far, in all of our experiments with similar datasets, we have usually seen better results with semantic search than with BM25. In addition, as mentioned earlier when discussing their work, their KDE sampling function fails to approximate the target distribution.



(a) Spanish STS **(b)** STS-B

Figure 6.3: Comparison of the density distributions of strategies to get similar pairs in the different datasets.

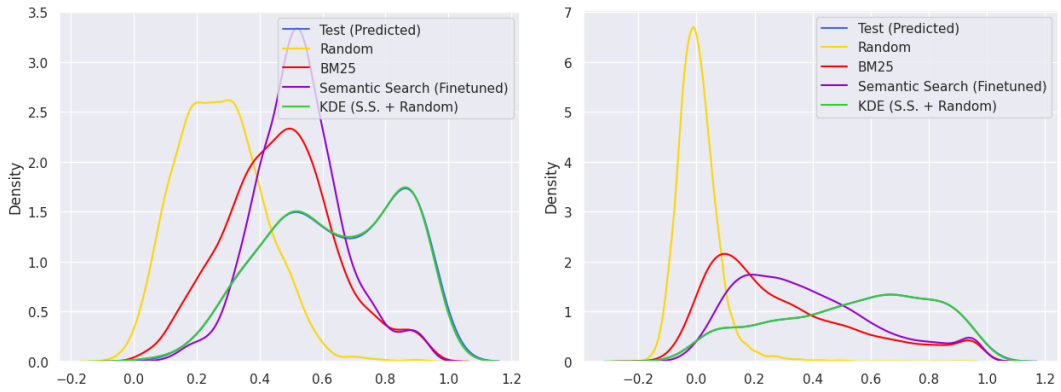
For our datasets, we tried to use the strategies to obtain similar sentence pairs. Namely, we tried BM25, Semantic Search (Pretrained) and Semantic Search (Finetuned). As you can see in the figure 6.3, the best results are usually obtained by the Finetuned Semantic Search sampling function.



(a) Spanish STS **(b)** STS-B

Figure 6.4: Comparison of the density distributions of strategies to apply KDE sampling in the different datasets.

Next, as for the KDE-approximated distributions in Figure 6.4, we find that our iterative approach, which we introduced in Section 4.3.2, does a good job of matching the target distribution. It also seems essential to sample candidates using semantic search rather than just randomly, because otherwise there are not enough positive candidates to do a good approximation. For our subsequent studies, we use the combined approach of semantic and random sampling, as it ensures that the dataset has enough negative and positive samples to draw from.



(a) Spanish STS **(b)** STS-B

Figure 6.5: Comparison of the density distributions of our proposed strategies to build sentence pairs.

You can find an illustration of all of our combined improved sampling strategies in Figure 6.5.

6.3 TSDAE Experiments Results

In this section, we will look at the training results of the original TSDAE and our improvements.

Method	**AskU.**	**CQADup.**	**Twitter**			**SciDocs**					**Avg.**
Subtask			**TURL**	**PIT**	**Avg.**	**Cite**	**CC**	**CR**	**CV**	**Avg.**	
Unsupervised learning based on BERT-base											
TSDAE	59.4	14.5	76.8	69.2	73.0	71.4	73.9	75.0	75.6	74.0	55.2
MLM	54.3	11.7	71.9	69.7	70.8	71.2	75.8	75.1	76.2	74.6	52.9
CT	56.3	13.3	74.6	70.4	72.5	63.4	67.1	70.1	69.7	67.6	52.4
SimCSE	55.9	12.4	74.5	62.5	68.5	62.5	65.1	67.7	67.6	65.7	50.6
BERT-flow	53.7	9.2	72.8	65.7	69.3	61.3	62.8	66.7	67.1	64.5	49.2
Domain adaptation: target task → NLI+STS											
TSDAE	59.4	14.4	75.8	**73.1**	**74.5**	**75.6**	**78.6**	**78.1**	**78.2**	**77.6**	**56.5**
MLM	**60.6**	14.3	75.0	68.6	71.8	74.7	78.2	77.0	77.6	76.9	55.9
CT	56.4	13.4	75.9	68.9	72.4	66.5	69.6	70.6	72.2	69.7	53.0
SimCSE	56.2	13.1	75.5	67.3	71.4	65.5	68.5	70.0	71.4	68.9	52.4
BERT-flow	58.2	13.9	76.5	67.4	72.0	62.2	64.8	68.1	68.0	65.8	52.5
Out-of-the-box supervised pre-trained models											
SBERT-base-nli-v2	53.4	11.8	75.4	69.9	72.7	66.8	70.0	70.7	72.8	70.1	52.0
SBERT-base-nli-stsb-v2	54.5	12.9	75.9	68.5	72.2	66.2	69.2	69.9	72.3	69.4	52.3
USE-large	(59.3)	***(15.9)***	**77.1**	69.8	73.5	67.1	69.5	71.4	72.6	70.2	54.7
In-domain supervised training (upper bound)											
SBERT-supervised	63.8	16.3	81.6	75.8	78.7	90.4	91.2	86.2	83.6	87.9	61.6

Table 6.9: Original evaluation using average precision. Results are averaged over 5 random seeds. The best results excluding the upper bound are bold. USE-large was trained with in-domain training data for AskUbuntu and CQADupStack (scores in italic). Adopted from Wang et al. [77], page 7.

First, in the original results of the TSDAE in table 6.9, the TSDAE is the best pre-training strategy, achieving a strong score of 55.2 using only unsupervised learning, beating even supervised sentence embedding models on these domain-specific tasks. Its performance is further improved in the setting where the models are fine-tuned on NLI + STS data after pre-training, achieving a strong average score of 56.5. Next, we look at our results for the experimental setup here.

Method	**AskU.**	**CQADup.**	**Twitter**			**SciDocs**					**Avg.**
Subtask			**TURL**	**PIT**	**Avg.**	**Cite**	**CC**	**CR**	**CV**	**Avg.**	
Augmented SBERT learning based on RoBERTa-base											
FullDomainSBERT	62.6	**15.0**	74.6	70.6	72.6	**75.6**	**77.2**	**76.5**	**77.7**	**76.8**	**56.7**
DomainSBERT	**63.2**	14.7	74.8	70.2	72.5	75.0	76.9	76.3	77.5	76.4	56.7
AugSBERT	60.8	14.8	74.8	**72.4**	**73.6**	70.3	73.0	73.4	75.1	73.0	55.5
Unsupervised learning based on BERT-base (CLS pooling)											
TSDAE (repr.)	57.4	11.9	**79.4**	59.1	69.3	69.8	73.5	74.2	75.4	73.2	52.9
SimCSE	56.9	12.7	70.0	61.9	65.9	59.3	62.1	66.1	65.5	63.3	49.7
NM	57.1	12.7	73.4	62.9	68.1	67.0	69.8	72.0	72.0	70.2	52.0
MLM + SimCSE	59.2	13.2	76.5	67.0	71.7	70.5	73.0	73.7	74.3	72.9	54.3
MLM + NM	59.0	13.3	76.5	61.3	68.9	69.8	72.4	74.2	74.0	72.6	53.5
TSDAE + NM	57.4	13.0	77.6	56.3	67.0	68.8	71.3	72.9	72.8	71.5	52.2
Unsupervised learning based on BERT-base (mean pooling)											
SimCSE	55.4	12.6	69.0	61.8	65.4	58.3	61.4	65.5	64.9	62.5	49.0
NM	57.5	13.2	74.0	64.4	69.2	65.7	68.0	71.2	70.6	68.9	52.2
MLM + SimCSE	58.2	11.7	72.6	71.3	71.9	71.1	73.4	74.2	75.0	73.4	53.8
MLM + NM	59.5	14.2	75.1	61.0	68.1	67.6	70.3	72.7	72.7	70.8	53.1
TSDAE + NM	58.6	13.9	77.1	65.3	71.2	66.1	69.6	72.0	71.6	69.8	53.4
Pretrained Sentence Transformer (upper bound)											
Pretrained	68.9	25.8	77.7	81.3	79.5	92.1	90.4	86.1	84.2	88.2	65.6

Table 6.10: Our evaluation using average precision. Results are averaged over 3 random seeds. The best results excluding the upper bound are bold.

If we look at our results in Table 6.10, we see that the TSDAE falls behind quite a bit. Instead of its previous score of 55.2 in the unsupervised setting, it could only achieve 52.9. However, we are not sure what the reason for this is. Unfortunately, the authors of the TSDAE did not publish their experimental code, so we were unable to check if there was a difference between our implementations. A possible difference could be the random seeds chosen for the TSDAE, or maybe we have a problem in our implementation. We leave it to future work to verify the results of the TSDAE, because our experiment suggests that it's not as strong as other methods. For example, it was beaten by both MLM + NM and MLM + SimCSE, with MLM + SimCSE being the strongest of all the new unsupervised methods we proposed. Looking at our proposed NM objective, we see that it beats SimCSE quite confidently by more than 2 points, indicating that the objective actually works and learns something. However, when the goal is combined with MLM, the minimal noise solution of the SimCSE method seems to be superior. We believe that this shows quite clearly that the noise mechanism of the TSDAE is not a desirable property to have in sentence embedding learning. It seems consistent with

the findings by Wettig et al. [79], that too much noise can hurt training. In this specific case, it might cause problems with making sentences that are subsequences of another sentence more similar than they should be. As for the Augmented SBERT methods, we can see that they far outperform the other unsupervised methods, and even outperform the supervised TSDAE, which was fine-tuned on NLI + STS-B data, by some points. Nevertheless, we find that the Augmented SBERT approaches that incorporate MLM pre-training seem to perform about 1 percent point better than the Augmented SBERT that simply trains on STS-B. MLM training in the distilled Bi-Encoder did not seem to make much difference. We believe that these results show that the Augmented SBERT architecture is a strong method for domain adaptation, beating all other methods we proposed. However, the Pretrained Sentence Transformer model far outperforms all other methods, even beating the previous "upper bound" achieved with Super In-Domain Learning. This is because the Pretrained Transformer has already been pretrained on 1 billion sentence pairs, and has likely seen many pairs from similar domains as the datasets. We believe that a future area of research should try to focus on domain adaptation of these pretrained transformers, since they already serve as strong baseline sentence embedding models.

6.4 Semi-Supervised Augmented SBERT Results

In this section, we will discuss the results of the Semi-Supervised Augmented SBERT approach.

Modelset	(Seed Opt.)	Spanish-STS	BWS (cross-topic)	BWS (in-topic)
Baseline	-	30.27	5.53	6.98
USE (Yang et al., 2019)	-	86.86	53.43	57.23
BERT	×	77.50 ± 1.49	65.06 ± 1.06	65.91 ± 1.20
SBERT	×	68.36 ± 5.28	58.04 ± 1.46	61.20 ± 1.66
BERT (Upper-bound)	✓	77.74 ± 1.24	65.78 ± 0.78	66.54 ± 0.94
SBERT (Lower-bound)	✓	72.07 ± 2.05	60.54 ± 0.99	63.77 ± 2.29
SBERT-NLPAug	✓	74.11 ± 2.58	58.15 ± 1.66	61.15 ± 0.86
AugSBERT-R.S.	✓	62.05 ± 2.53	59.95 ± 0.70	64.54 ± 1.90
AugSBERT-KDE	✓	74.67 ± 1.01	**61.49 ± 0.71**	**69.76 ± 0.50**
AugSBERT-BM25	✓	75.08 ± 1.94	61.48 ± 0.73	68.63 ± 0.79
AugSBERT-S.S.	✓	74.99 ± 2.30	61.05 ± 1.02	68.06 ± 0.93
AugSBERT-BM25+S.S.	✓	**76.24 ± 1.42**	59.41 ± 0.98	63.30 ± 1.34

Table 6.11: Summary of all the original scores for the in-domain tasks in the original Augmented SBERT paper. Adopted from Thakur et al.[74], page 7.

In the results obtained by Thakur et al. in Table 6.11, the improvements made by their method are in the range of 1-6% points, which is quite a good result. What we find interesting is the random sampling in their results, as we find that the low scores could be related to the fact that the CE is more unpredictable when distilling negative pairs, as it has a different range of values from the BE. Before we look at our results for the CBE architecture, we next look at our reproduced results for the CE.

Model	Spanish-STS	BWS (cross-topic)	BWS (in-topic)
RoBERTA (Upper-bound)	80.51 ± 1.32	65.75 ± 0.47	66.72 ± 0.39
SRoBERTa (Lower-bound)	69.91 ± 2.95	62.64 ± 1.23	65.04 ± 0.60
AugSBERT-None	67.76 ± 0.49	**63.80 ± 1.07**	66.38 ± 0.70
AugSBERT-R.S.	75.93 ± 1.45	62.82 ± 1.38	**67.74 ± 0.76**
AugSBERT-KDE	74.70 ± 1.98	59.65 ± 0.13	64.50 ± 1.24
AugSBERT-BM25	77.17 ± 1.94	62.54 ± 1.76	66.84 ± 1.06
AugSBERT-S.S.	**77.09 ± 1.99**	62.07 ± 1.37	66.64 ± 1.97

Table 6.12: Results for AugSBERT semi with CE. Results are averaged over 3 random seeds.

Our results for the CE, in Table 6.12, look slightly different from those obtained by Thakur et al. An important difference to note is that we use RoBERTa for our experiments. An interesting observation we can make in our results is that for the BWS (cross-topic) setting, the Augmented SBERT *without* any augmentations performs the best, highlighting the potential advantages that CBE can bring, as we will see in the next table.

Model	**Spanish-STS**	**BWS (cross-topic)**	**BWS (in-topic)**
RoBERTA (Upper-bound)	75.56 ± 3.35	66.58 ± 1.17	69.97 ± 0.40
SRoBERTa (Lower-bound)	69.91 ± 2.95	62.64 ± 1.23	65.04 ± 0.60
AugSBERT-None	68.10 ± 4.80	63.39 ± 0.61	67.64 ± 0.64
AugSBERT-R.S.	74.68 ± 1.42	**64.88 ± 0.99**	**69.91 ± 0.90**
AugSBERT-KDE	77.40 ± 0.96	64.22 ± 0.89	69.63 ± 0.33
AugSBERT-BM25	77.76 ± 0.45	64.46 ± 0.63	69.85 ± 0.88
AugSBERT-S.S.	**79.59 ± 1.55**	64.11 ± 0.10	68.40 ± 0.70

Table 6.13: Results for AugSBERT semi with CBE Results are averaged over 3 random seeds.

The results for the semi-supervised experiments using the CBE are quite robust, as can be seen in the Table 6.13. We achieve state-of-the-art results within a few percentage points. We believe this demonstrates the huge advantage of the CBE architecture. One interesting observation we would like to make, however, is that our iterative KDE approach seems to be mediocre in terms of results. A possible reason for this could be that discarding samples leads to a smaller silver dataset, which may cause the lower results. We think this warrants further investigation, as intuitively we expected the KDE to perform the best.

6.5 Distilling Topic Information Results

We will now have a look at our last setting, where we attempt to distill topic information along with similarity.

Model	Prompted	Raw (F1)	Prompted (F1)	Avg.
Pretrained SBERT	Original	68.86	73.98	71.42
Classifier	Original	64.48	68.35	66.42
Pretrained RoBERTa	Original	63.06	69.17	66.12
FullDomainSBERT	Original	62.86	68.65	65.75
FullDomainSBERT	Hidden	63.12	68.3	65.71
FullDomainSBERT	Prompted	63.3	66.76	65.03
DomainSBERT	Hidden	60.9	67.41	64.16
DomainSBERT	Original	60.82	67.47	64.14
DomainSBERT	Prompted	61.23	65.02	63.13
Classifier	Prompted	50.67	29.31	39.99

Table 6.14: Topic results for the supervised setting with the 20 Newsgroups dataset.Results are averaged over 2 random seeds.

For the topic performance results in Table 6.14, we can see that our approach was not successful in the 20newsgroup dataset. Trying to distill or learn the actual groups from the labels of the parent groups did not work at all. In fact, all Augmented-SBERT models fall behind even the pre-trained RoBERTa model. Again, we can see that the pre-trained SentenceTransformer has robust results, with a 5 point lead in score, and even outperforms the supervised classifier.

Model	Prompted	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Pretrained SBERT	Original	70.78	77.88	77.76	83.29	83.16	82.87	83.18	79.84
FullDomainSBERT	Original	63.66	69.05	69.92	76.43	78.32	70.41	72.57	71.48
FullDomainSBERT	Prompted	63.72	69.1	69.18	75.83	76.26	69.48	72.19	70.82
DomainSBERT	Hidden	63.33	68.07	69.2	75.37	76.24	70.71	71.58	70.64
DomainSBERT	Prompted	62.77	68.07	68.97	75.03	76.33	69.84	71.98	70.43
DomainSBERT	Original	62.12	68.62	68.75	75.49	76.11	69.3	71.99	70.34
FullDomainSBERT	Hidden	63.21	67.77	68.46	74.98	76.13	68.45	71.09	70.01
Pretrained RoBERTa	Original	53.82	46.58	56.64	64.96	63.62	69.83	76.84	61.76
Classifier	Original	46.05	36.77	39.89	43.3	50.44	53.71	61.69	47.41
Classifier	Prompted	38.18	33.51	32.96	39.48	45.62	47.61	61.99	42.77

Table 6.15: SentEval results for the 20 Newsgroups dataset. Results are averaged over 2 random seeds.

However, as we can see in the Table 6.15, the Augmented SBERT was able to

distill the sentence similarity at least to some extent, achieving an improvement of up to 10% over the pretrained RoBERTa, but again falling far behind the pretrained SBERT. In conclusion, our proposed approach does not seem to work well, at least in the case of the 20 newsgroups. We cannot say for sure if the problem is related to our dataset or our methods, so we will look at the next dataset first.

Model	**Prompted**	**Raw BWS (ρ)**	**Prompted BWS (ρ)**	**Raw UKP (F1)**	**Prompted UKP (F1)**	**Avg.**
FullDomainSBERT	Prompted	62.74	56.88	74.45	99.78	73.46
DomainSBERT	Original	63.09	55.14	74.04	99.36	72.91
FullDomainSBERT	Original	64.38	52.49	74.04	99.14	72.51
DomainSBERT	Prompted	60.8	53.96	73.53	99.68	71.99
FullDomainSBERT	Hidden	58.44	48.11	75.39	98.99	70.23
Pretrained SBERT	Original	55.65	42.1	78.13	99.98	68.97
DomainSBERT	Hidden	55.47	41.5	75.06	98.81	67.71
Pretrained RoBERTa	Original	45.2	39.05	74.1	99.27	64.41
Classifier	Original	32.92	16.81	76.32	98.78	56.21
Classifier	Prompted	28.83	15.84	76.48	100	55.28

Table 6.16: Topic results for the supervised setting with the UKP Sentential Argument Mining dataset. Results are averaged over 3 random seeds.

Looking at the results in Table 6.16, we can see that almost all Augmented SBERT settings were able to outperform the pre-trained SBERT model in terms of the BWS similarity metric. However, again, the method seems to be mostly unsuccessful in terms of achieving meaningful improvements towards distilling topic sensitivity for the most part. The classifier model, which is the only model that beats the pre-trained RoBERTa, is very bad at the STS task, achieving only 56.21 points. Interestingly, however, the "Hidden" prompt settings were able to achieve a 1 percent improvement in topics over the pre-trained RoBERTa. However, they are still 3 points behind the Pretrained SBERT model.

Model	Prompted	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
Pretrained SBERT	Original	70.78	77.88	77.76	83.29	83.16	83.18	82.87	79.84
DomainSBERT	Prompted	67.46	74.08	74.29	79.75	80.3	77.03	75.38	75.47
FullDomainSBERT	Prompted	67.94	73.4	74.71	79.25	80.03	76.22	75.89	75.35
DomainSBERT	Original	67.29	74.41	74.16	79.21	79.61	76.63	74.54	75.12
FullDomainSBERT	Original	67.94	73.92	74.27	78.67	79.38	75.9	75.46	75.08
DomainSBERT	Hidden	66.43	71.04	72.4	77.25	79.18	77.12	74.72	74.02
FullDomainSBERT	Hidden	66.29	70.86	71.79	76.62	78.34	75.55	73.23	73.24
Pretrained RoBERTa	Original	53.82	46.58	56.64	64.96	63.62	76.84	69.83	61.76
Classifier	Prompted	52.26	54.05	47	54.79	61.05	67.01	58.84	56.43
Classifier	Original	45.38	45.93	47.62	52.98	57.84	62.38	51.28	51.92

Table 6.17: SentEval results for the supervised setting with the UKP Sentential Argument Mining dataset. Results are averaged over 3 random seeds.

In our last Table 6.17, we can see that on the UKP dataset, the augmented SBERT was able to achieve a more significant improvement in STS scores than the 20 Newsgroups dataset by about 4 points. Interestingly, the baseline classifier models are again very bad at the STS task. However, it is still not at all close to the pre-trained SBERT model, as it still beats all other models by 4 percentage points. In conclusion, we believe that our last proposed method was unsuccessful in achieving the desired goal of distilling topic information using the Augmented SBERT objective.

Chapter 7

Conclusion and Future Work

We conclude with providing a summary of our findings and an interpretation of the results. Finally, we end with touching upon some potential future research directions.

7.1 Conclusion

In this thesis, we conducted a comprehensive investigation of sentence embedding training in various supervision settings with the aim of improving current methods. We introduced a novel architecture, the Cross-Bi-Encoder, for Cross-Encoders, a specialized architecture for sentence pair tasks, and demonstrated its effectiveness in improving the distillation performance of Augmented SBERT. Through ablation studies, we gained insights into the learning process of TSDAE and introduced a new objective, Noise Matching (NM), that incorporates contrastive learning to train noise-invariant embeddings. The proposed modifications and techniques were evaluated on established datasets and compared with state-of-the-art methods, demonstrating improved performance.

Additionally, we investigated the impact of incorporating labeled topic information in a supervised setting. Unfortunately, we did not find any positive results.

In conclusion, our findings contribute to the field of sentence embeddings by introducing a novel architecture for Cross-Encoders and providing new insights into the learning process of TSDAE. Our work highlights the importance of considering the proper architecture, supervision setting and objectives when training sentence embeddings, and provides a foundation for future research in this area.

7.2 Limitations and Future Work

We have explored various aspects of sentence embedding training in unsupervised, semi-supervised, and supervised settings. However, there are still several limitations and avenues for future work that could be pursued to further improve the performance of sentence embeddings.

One limitation is the lack of exploration of larger models or models relying on large-scale pre-training, as we did not have abundant cheap GPU resources available to us during our experiments. Our focus was on training domain-specific models in settings where limited data is available. Additionally, we did not evaluate our methods on benchmarks such as MTEB [64] or BEIR [75].

In terms of future work, further investigation could be conducted on the sampling size for the semi-supervised learning procedure in the Augmented SBERT with the Cross-Bi-Encoder. It is possible that greater improvements could be achieved with a larger sampling size, which would further increase the size of the silver dataset.

Additionally, due to the large number of experiments performed, no hyperparameter search was performed, which could be explored in future work.

Another avenue for future work is to train and evaluate the Cross-Bi-Encoder architecture with larger datasets, which could result in a strong pre-trained Cross-Bi-Encoder architecture and better performance in more domain adaptation settings. It could also be worthwhile to investigate the impact of freezing layers for distillation in the Cross-Bi-Encoder architecture on downstream task performance.

Finally, future work could also focus on improving existing strong sentence embedding models, such as the Pretrained SentenceTransformer, rather than starting from BERT or RoBERTa each time. This could lead to even better performance in various NLP tasks. Note that these models will require either supervised or novel unsupervised methods to train, since they are already good at solving sentence-embedding tasks.

In conclusion, there are many exciting opportunities for future work in the field of sentence embeddings, and we hope that this thesis will inspire further research and advancements in this area.

Bibliography

[1] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263, 2015.

[2] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@ COLING*, pages 81–91, 2014.

[3] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez Agirre, Rada Mihalcea, German Rigau Claramunt, and Janyce Wiebe. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics), 2016.

[4] Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, 2012.

[5] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43, 2013.

[6] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

[7] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1(2665):2012, 2012.

[10] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[11] John C Callender and HG Osburn. An empirical comparison of coefficient alpha, guttman's lambda-2, and msplit maximized split-half reliability estimates. *Journal of Educational Measurement*, pages 89–99, 1979.

[12] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[15] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 767–776, 2017.

[16] Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*, 2022.

[17] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

[18] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[19] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[20] Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljačić. Equivariant contrastive learning. *arXiv preprint arXiv:2111.00899*, 2021.

[21] Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[23] Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*, 2020.

[24] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679, 2021.

[25] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[26] Yang Gao, Yue Xu, Heyan Huang, Qian Liu, Linjing Wei, and Luyang Liu. Jointly learning topics in sentence embedding for document summarization. *IEEE Transactions on Knowledge and Data Engineering*, 32(4):688–699, 2019.

[27] Mandy Guo, Qinlan Shen, Yinfei Yang, Heming Ge, Daniel Cer, Gustavo Hernandez Abrego, Keith Stevens, Noah Constant, Yun-Hsuan Sung, Brian Strope, et al. Effective parallel corpus mining using bilingual sentence embeddings. *arXiv preprint arXiv:1807.11906*, 2018.

[28] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[29] Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *Acm Sigact News*, 28(4):41–60, 1997.

[30] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[31] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[32] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[33] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

[34] Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian document computing symposium*, pages 1–8, 2015.

[35] Hang Hua, Xingjian Li, Dejing Dou, Cheng-Zhong Xu, and Jiebo Luo. Noise stability regularization for improving bert fine-tuning. *arXiv preprint arXiv:2107.04835*, 2021.

[36] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[37] Sverker Janson, Evangelina Gogoulou, Erik Ylipää, Amaru Cuba Gyllensten, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. In *International Conference on Learning Representations, 2021*, 2021.

[38] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[39] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[40] Svetlana Kiritchenko and Saif M Mohammad. Capturing reliable fine-grained sentiment associations by crowdsourcing and best-worst scaling. *arXiv preprint arXiv:1712.01741*, 2017.

[41] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[42] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[43] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[44] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391*, 2017.

[45] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[46] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[47] Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. Semi-supervised question retrieval with gated convolutions. *arXiv preprint arXiv:1512.05726*, 2015.

[48] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020.

[49] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. *AI Open*, 3:71–90, 2022.

[50] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*, 2020.

[51] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2020.

[52] Fangyu Liu, Yunlong Jiao, Jordan Massiah, Emine Yilmaz, and Serhii Havrylov. Trans-encoder: Unsupervised sentence-pair modelling through self-and mutual-distillations. *arXiv preprint arXiv:2109.13059*, 2021.

[53] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[54] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[55] Zhiyuan Liu, Yankai Lin, Maosong Sun, Zhiyuan Liu, Yankai Lin, and Maosong Sun. Representation learning and nlp. *Representation Learning for Natural Language Processing*, pages 1–11, 2020.

[56] Edward Ma. nlpaug: data augmentation for nlp, 2019.

[57] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216–223. Reykjavik, 2014.

[58] Louis Martin, Angela Fan, Éric De La Clergerie, Antoine Bordes, and Benoît Sagot. Muss: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*, 2020.

[59] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[60] Yu Meng, Chenyan Xiong, Payal Bajaj, Paul Bennett, Jiawei Han, Xia Song, et al. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *Advances in Neural Information Processing Systems*, 34:23102–23114, 2021.

[61] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[62] Amita Misra, Brian Ecker, and Marilyn A Walker. Measuring the similarity of sentential arguments in dialog. *arXiv preprint arXiv:1709.01887*, 2017.

[63] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines, 2021.

[64] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[65] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[66] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[67] Barbara Plank. What to do about non-standard (or non-canonical) language in nlp. *arXiv preprint arXiv:1608.07836*, 2016.

[68] Alan Ramponi and Barbara Plank. Neural unsupervised domain adaptation in nlp—a survey. *arXiv preprint arXiv:2006.00632*, 2020.

[69] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[70] Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*, 2019.

[71] Christian Stab, Tristan Miller, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources using attention-based neural networks, 2018.

[72] Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*, 2021.

[73] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[74] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 2020.

[75] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

[76] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[77] Kexin Wang, Nils Reimers, and Iryna Gurevych. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. *arXiv preprint arXiv:2104.06979*, 2021.

[78] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*, 2022.

[79] Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005*, 2022.

[80] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[81] Xing Wu, Chaochen Gao, Zijia Lin, Jizhong Han, Zhongyuan Wang, and Songlin Hu. Infocse: Information-aggregated contrastive learning of sentence embeddings. *arXiv preprint arXiv:2210.06432*, 2022.

[82] Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

[83] Wei Xu, Chris Callison-Burch, and William B Dolan. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 1–11, 2015.

[84] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[85] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[86] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[87] Qihuang Zhong, Liang Ding, Yibing Zhan, Yu Qiao, Yonggang Wen, Li Shen, Juhua Liu, Baosheng Yu, Bo Du, Yixin Chen, et al. Toward

efficient language model pretraining and downstream adaptation via self-evolution: A case study on superglue. *arXiv preprint arXiv:2212.01853*, 2022.

[88] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[89] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*, 2021.

Technological support used

In the process of writing this thesis, the following technological support was used:

- Web-based version of ChatGPT (Jan 30)
- Unofficial text-chat-davinci-002-20221122 model (for writing initial text for sections 2.1, 2.2, and 2.3)
- DeepL Write (for proofreading and rewriting large parts of the thesis)
- Grammarly (for proofreading)

Areas of help

The technological assistance mentioned above was used for the following purposes:

- Transforming key ideas from bullet points into flowing text
- Formulating text throughout the paper
- Debugging Latex code and tables
- Creating diagrams such as encoder architectures
- Generating Python code

Integration process

ChatGPT was used as a paraphrase model, and programming tool. DeepL Write rewrote and proofread almost the entire thesis.

What is your own contribution

All information and methodology in this thesis is our own work. We only used modern AI tools to help formulate our thoughts. All citations are properly placed. Any places where citations are missing, are considered common sense or our own thoughts.