Masterarbeit im Fach Informatik
Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. H. Ney

Improving Unsupervised Contrastive Learning for Sentence Embeddings

14. June 2022

vorgelegt von:
Ruixiang Wang

Gutachter:
Prof. Dr.-Ing. Hermann Ney
Prof. Gerhard Lakemeyer, Ph. D.

Betreuer:
M. Sc. David Thulke
M. Sc. Leonard Dahlmann

Abstract

Learning sentence embeddings of high quality benefits a wide range of Natural Language Processing (NLP) tasks, like sentences similarity analysis, information retrieval, etc. Moreover, labeled data is rare and costly but unlabelled data can be easily acquired in many real-world scenarios. Thus, learning sentence embeddings in an unsupervised way plays a significant role in many NLP applications.

Recently, contrastive learning has been attracting much attention for learning unsupervised sentence embeddings. In this work, two state-of-the-art contrastive learning methods SimCSE and ConSERT are investigated, which are based on pre-trained language models. We also show contrastive learning methods do help to improve the quality of sentence embeddings on downstream tasks.

Furthermore, several autoencoder-based methods can help to produce good sentence embeddings. We investigate two kinds of autoencoder-based methods Causal Language Modelling (CLM) autoencoder and Masked Language Modelling (MLM) autoencoder in this thesis. Both of them can learn effective sentence embeddings in an unsupervised way. Also, we introduce a multi-task learning approach which combines contrastive learning methods and autoencoder-based methods, and we find this combination can improve the performance of sentence embeddings on retrieval tasks in some cases.

To measure the quality of sentence embeddings, we consider several downstream evaluation tasks which are eProduct challenge, Semantic Textual Similarity (STS) and duplicate questions retrieval. In this thesis, we conduct experiments with different methods, in order to investigate and analyze their respective characteristics.

Contents

Chapter 1

Introduction

Learning effective sentence embeddings is a fundamental task in the Natural Language Processing (NLP) field. Many papers and studies are devoted to finding an efficient way to learn good sentence embeddings [Le & Mikolov 14, Kiros & Zhu$^+$ 15, Hill & Cho$^+$ 16, Conneau & Kiela$^+$ 17, Logeswaran & Lee 18, Cer & Yang$^+$ 18, Reimers & Gurevych 19], since sentence embeddings of high quality benefit a wide range of downstream tasks, like semantic textual similarity comparison [Conneau & Kiela 18] and information retrieval [Reimers & Gurevych 19]. Also, in eBay-specific applications, sentence embeddings play an important part in many scenarios, including retrieving relevant products when a search query is given by customers, similar product recommendations, etc. Therefore, once we have decent sentence embeddings, the user experience would be improved when the customers are shopping on the eBay e-commerce platform.

The sentence embedding techniques aim to represent entire sentences and their semantic information into a fixed-size, dense vector space so that similar semantic sentences are close to each other and dissimilar sentences are far apart. By doing this, the semantic information of sentences would be represented as fix-sized dense vectors, which can be applied to many downstream tasks, like classification tasks and retrieval tasks.

There are many successful approaches to learn sentence embeddings, like InferSent [Conneau & Kiela$^+$ 17], Universial Sentence Encoder [Cer & Yang$^+$ 18], SentenceBERT [Reimers & Gurevych 19]. What these approaches have in common is that all of them need additional supervision to learn sentence embeddings, i.e. they heavily rely on labeled data to train the model. However, labeled data and human annotation are costly and often unavailable in many real-world scenarios. On the other side, there are plenty of unlabelled corpora which can be easily acquired, like Wikipedia corpus, product titles and descriptions from eBay e-commerce platform, blogs from websites, etc. Thus, to overcome the shortage of labelled data and make use of these massive unlabelled data, unsupervised methods have been introduced which embed sentences just using unlabelled data for training, including SkipThought vectors [Kiros & Zhu$^+$ 15], Paragraph Vector [Le & Mikolov 14], contrastive learning methods SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$

21], sequential denoising autoencoders [Vincent & Larochelle$^+$ 10, Hill & Cho$^+$ 16] based method TSDAE [Wang & Reimers$^+$ 21], etc.

Contrastive learning is proven to be an effective method that can boost the quality of sentence embeddings [Gao & Yao$^+$ 21, Yan & Li$^+$ 21, Chuang & Dangovski$^+$ 22] in a supervised or unsupervised way. The goal of contrastive learning is to pull semantic similar sentences that we also call positive sentences together, and push apart semantic dissimilar sentences that we also call negative sentences by using contrastive loss during training [Hadsell & Chopra$^+$ 06]. In this thesis, we focus on two state-of-the-art contrastive learning methods SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$ 21].

Furthermore, autoencoder-based methods like TSDAE [Wang & Reimers$^+$ 21] can also produce decent sentence embeddings. TSDAE uses an encoder-decoder architecture, where the encoder maps sentences into fixed-sized vectors, and the decoder tries to reconstruct the original sentences from this sentence embedding. Inspired by this approach, we consider two kinds of autoencoder-based methods Causal Language Modelling (CLM) autoencoder and Masked Language Modelling (MLM) autoencoder in this thesis.

To measure the quality of sentence embeddings, we consider using three kinds of evaluation tasks: Semantic Textual Similarity (STS) [Agirre & Cer$^+$ 12, Agirre & Cer$^+$ 13, Agirre & Banea$^+$ 14, Agirre & Banea$^+$ 15, Agirre & Banea$^+$ 16, Conneau & Kiela 18], eProduct challenge [Yuan & Chiang$^+$ 21] from eBay and duplicate questions retrieval [Thakur & Reimers$^+$ 21]. We report the evaluation results of these tasks, which are used to measure the performance of sentence embeddings generated by different approaches. Besides, to conduct a fair comparison, we have several baselines: using the Word2vec [Mikolov & Chen$^+$ 13] technique to obtain word embedding of each word in the same sentence and average them to get the sentence embedding, using BM25 [Robertson & Zaragoza 09] to directly calculate sentence similarity scores when doing eProduct challenge and duplicate question retrieval evaluations, and obtaining sentence embeddings derived from the pre-trained BERT [Devlin & Chang$^+$ 19] model.

In this thesis, we will follow research directions corresponding to the following questions:

- **Do contrastive learning methods help to improve the quality of sentence embeddings on downstream tasks?**

  To answer this question, we consider two state-of-the-art contrastive learning methods SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$ 21]. In Section 5.3.1, we conduct experiments to compare these contrastive learning methods with baselines on specific downstream tasks. We observe that contrastive learning methods outperform several baselines, but BM25 is a strong baseline that can not be taken over easily.

- **Is Semantic Textual Similarity (STS) task a good way to measure the quality of sentence embeddings?**

  Since many papers use the STS task to evaluate sentence embeddings [Gao & Yao$^+$ 21, Yan & Li$^+$ 21, Chuang & Dangovski$^+$ 22], we wonder if the STS task is a really appropriate way to measure the quality of sentence embeddings [Wang & Reimers$^+$ 21]. In Section 5.3.1, we compare the STS task evaluation results with results of downstream task eProduct [Yuan & Chiang$^+$ 21]. We observe that the evaluation results of these two tasks are not correlated.

- **How do contrastive learning methods behave differently on downstream tasks compared to BM25?**

  We wonder what is the difference between contrastive learning methods and count-based method BM25 [Robertson & Zaragoza 09] on downstream tasks. In Section 5.3.2, we compare the evaluation results of eProduct for these two methods. We find the two methods are complementary in some cases.

- **Are there any factors that influence the performance of sentence embeddings on downstream tasks when using contrastive learning methods (e.g. loss functions, batch size, size of training dataset, domain of training dataset)?**

  We first conduct an experiment to see how contrastive learning with different loss functions (binary cross-entropy loss and NY-Xent loss [Chen & Kornblith$^+$ 20]) behaves on the downstream task in Section 5.3.3. In Section 5.3.4, we investigate how different batch sizes influence the performance on downstream tasks for contrastive learning by choosing batch size in $\{32, 64, 96, 128, 256, 512\}$. We also explore what is the best size of the training corpus for contrastive learning in Section 5.3.5, where we observe that the size of the training corpus can significantly affect the performance of downstream tasks and a larger size can not give better results. Moreover, in Section 5.3.6, we conduct experiments using different domains of training datasets to discuss how different domains affect contrastive learning.

- **Is there a good way to take advantage of both contrastive learning methods and autoencoder-based methods in order to improve the quality of sentence embeddings?**

  To answer this question, in Section 3.3, we propose a multi-task learning approach that combines contrastive learning methods and autoencoder-based methods. We also conduct experiments to evaluate this multi-task approach on downstream tasks and compare it with "single-task" contrastive learning and autoencoder in Section 5.3.7.

Chapter 2

Background

2.1 Dropout

We first give an introduction of dropout, since it is the key ingredient of state-of-the-art contrastive learning method SimCSE [Gao & Yao$^+$ 21] in Section 3.1.1, and it is widely used in many deep learning models [Vaswani & Shazeer$^+$ 17, Devlin & Chang$^+$ 19, Reimers & Gurevych 19, Liu & Ott$^+$ 19].

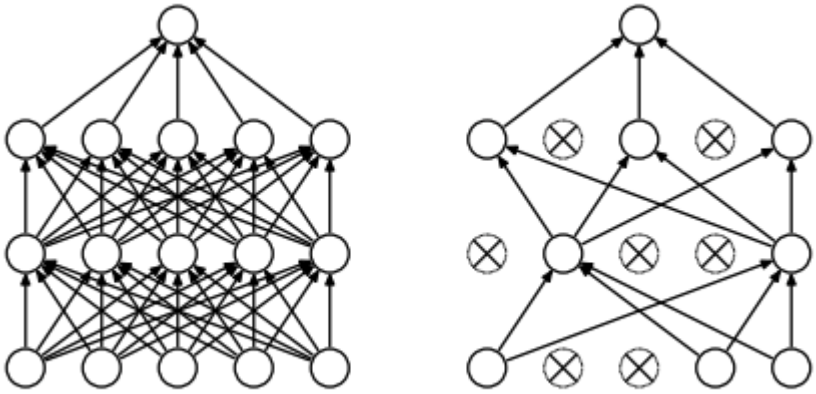
The initial motivation of dropout is to avoid overfitting when training a deep neural network. Overfitting means the trained model can predict the samples of the training dataset accurately but has poor performance on the test or validation dataset.

Dropout is a technique that randomly drops neurons from the neural network during training [Srivastava & Hinton$^+$ 14]. Dropout modifies the way of learning all the parameters in the neural network to learning only a fraction of the parameters in the network. From the Figure 2.1, we can see that during the standard training phase, all neurons are involved, but after applying dropout, only a few selected neurons are involved and the rest of them are removed. Thus, after every iteration during training, different sets of neurons are activated, which could prevent some neurons from dominating the process. Because of this, dropout can help to avoid overfitting, as well as make deeper network architectures generalize well.

2.2 Sentence Representation

2.2.1 Word2vec

The goal of word embedding techniques aims to encode words in a vector space so that the semantic information of each word can be represented by a vector. Word2vec [Mikolov & Chen$^+$ 13] is one of the classical word embedding techniques that uses a neural network model to learn word associations from a large corpus of text. Once trained, the neural network model can map words into word embeddings in a vector space, where similar words are close and dissimilar words are far apart. Thus, these word embeddings can be used to calculate similarity scores (e.g.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.1: Example of how dropout works on neural network [Srivastava & Hinton+ 14]. Left is a standard neural network with 2 hidden layers, right is thinned neural network generated by applying dropout to the network on the left.

the cosine similarity between the vectors), which indicates the level of semantic similarity between the words.

There are two approaches to training the neural network model in Word2vec: either using context words to predict a center word (a method known as continuous bag of words, or CBOW) or using a center word to predict context words, which is called skip-gram. Figure 2.2 illustrate these two approaches for Word2vec.

Given a sentence x_i with a sequence words $(x_i^0, x_i^1, \ldots, x_i^n)$, the Word2vec model maps each word into a vector in order to get contextualized embeddings $(e_i^0, e_i^1, \ldots, e_i^n)$ of all words in a sentence. Then we can take pooling strategies of these contextualized embeddings to obtain sentence representation $f_\theta(x_i)$. Two main pooling strategies would be used in Word2vec: **Average pooling** uses the mean of the contextualized embeddings of all words in a sentence. **Max pooling** uses the max-over-time of the contextualized embeddings of all words in a sentence.

2.2.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is designed to pretrain deep bidirectional representations from unlabeled corpus by jointly conditioning on both left and right context in all layers [Devlin & Chang+ 19]. Currently, it is a state-of-the-art model for a wide range of tasks, like question answering, language inference, and it can also be used to generate sentence representations.

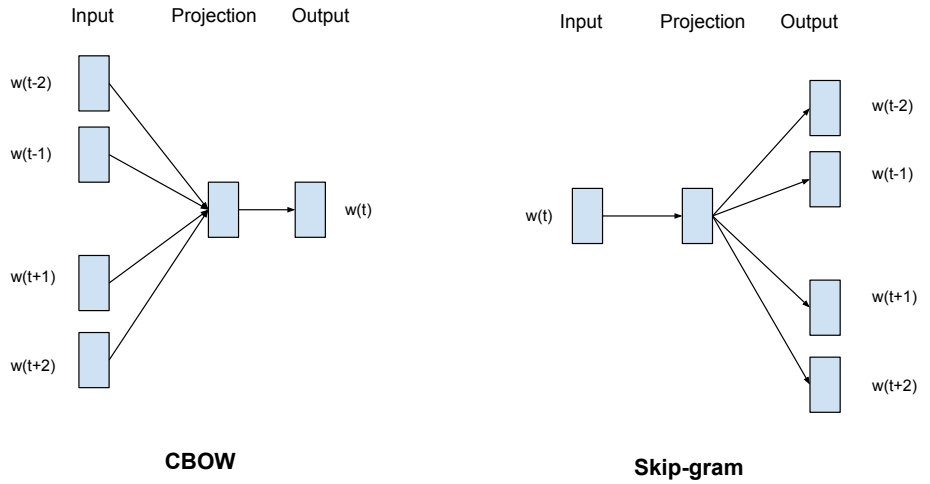


Figure 2.2: The model structure of CBOW and Skip-gram. $w(t)$ is center word and $w(t-2), w(t-1), w(t+1), w(t+2)$ are context words.

BERT is a large model which consists of multiple Transformer [Vaswani & Shazeer$^+$ 17] encoder blocks. The BERT model has been pretrained on a large amount of unlabeled corpus, including BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables, and headers). Then, if apply BERT on different downstream tasks, the pre-trained BERT needs to be fine-tuned with task-specific training datasets, so that the fine-tuned BERT can fit into these specific tasks. During fine-tuning, all parameters of BERT are fine-tuned. This procedure is illustrated in Figure 2.3.

Besides, BERT uses both Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks as the pre-training objectives. The Masked Language Model (MLM) is a fill-in-the-blank task: words are masked from the input and the trans-

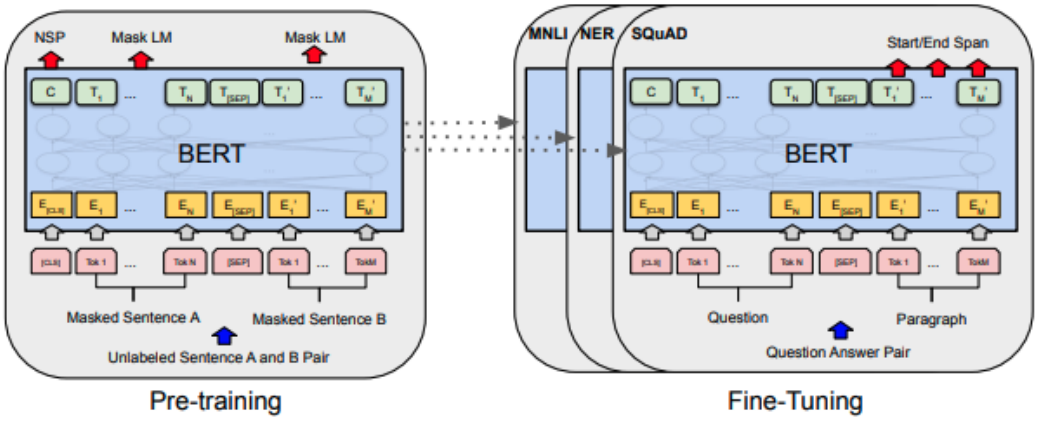


Figure 2.3: Overview of pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. [Devlin & Chang+ 19]

former network tries to predict the missing words. Mask Language Models like BERT predict a masked subset of input tokens based on the remaining context and are effective on downstream tasks due to their bidirectional learning nature. Figure 2.4 gives an example of how MLM works. In the Next Sentence Prediction (NSP), the BERT receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. This task is shown in the left side of Figure 2.3. It is used to learn about the relationships between sentences to indicate whether the previous sentence is followed by the next one as far as the context goes.

When feed a sentence x_i into BERT, x_i will be first tokenized into a sequence of tokens: $([\texttt{CLS}], x_i^0, x_i^1, \ldots, x_i^n, [\texttt{SEP}])$, where [CLS] is a special token added at the beginning of the sentence, [SEP] token is inserted at the end of the sentence. After through the last layer of BERT, contextualized embeddings $(e_{\text{cls}}, e_i^0, e_i^1, \ldots, e_i^n, e_{\text{sep}})$ of all these tokens will be generated. To obtain the sentence representation $f_\theta(x_i)$, either [CLS] token embedding (i.e. $f_\theta(x_i) = e_{\text{cls}}$) or average/max pooling of the contextualized embeddings can be used.

In addition, we have eBert model in eBay which has the identical architecture to the BERT model. The main difference is that eBert model has been pretrained on the BooksCorpus and English Wikipedia (same in Bert-base-uncased), as well as 1B item titles from the eBay e-commerce platform.

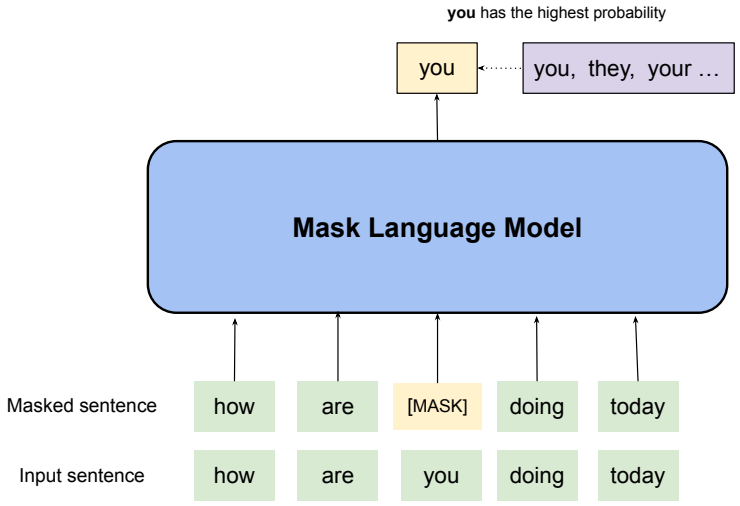


Figure 2.4: Example of how MLM works. In this case, the MLM tries to predict the masked token [you].

2.3 Retrieval Methods

Since we have several information retrieval downstream tasks, including eProduct [Yuan & Chiang+ 21] and duplicate questions retrieval [Thakur & Reimers+ 21], the methods used to do retrieval tasks are necessary. We consider two different kinds of retrieval methods, one is count-based method and the other is embedding-based method. In this thesis, we focus on using one of the count-base methods BM25, and using sentence embeddings derived from Word2Vec [Mikolov & Chen+ 13] model or pre-trained language model BERT [Devlin & Chang+ 19] as embedding-based methods in experiments.

2.3.1 Count-based Retrieval

BM25 [Robertson & Zaragoza 09] is a count-based method, which can be used to calculate the similarity score between two sentences. Given a query q_1^I and a set of N documents D, the BM25 score of the document $[d_n]_1^{J_n}$ is:

$$S(q_1^I, [d_n]_1^{J_n}) = \sum_{i=1}^{I} \mathrm{IDF}(q_i) \cdot \frac{\mathrm{TF}(q_i, [d_n]_1^{J_n}) \cdot (k_1 + 1)}{\mathrm{TF}(q_i, [d_n]_1^{J_n}) + k_1 \cdot (1 - b + b \cdot \frac{J_n \cdot N}{\sum_{n'=1}^{N} J_{n'}})} \tag{2.1}$$

with k_1 and b hyperparameters

$$\mathrm{TF}(q_i, [d_n]_1^{J_n}) = \frac{\sum_{j=1}^{J_n} \delta(q_i, [d_n]_j)}{J_n}$$

and

$$\mathrm{IDF}(q_i) = \log \left(\frac{N}{\sum_{n=1}^{N} \delta(q_i \in d_n)} \right)$$

where $\mathrm{TF}(q_i, [d_n]_1^{J_n})$ is q_i' term frequency in document $[d_n]_1^{J_n}$, $\mathrm{IDF}(q_i)$ is the inverse document frequency weight of query term q_i. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization.

Based on the BM25 scores between query and documents, we could try to retrieve the most relevant documents when given a query.

2.3.2 Embedding-based Retrieval

From the introduction in Section 2.2, we know both Word2vec [Mikolov & Chen$^+$ 13] model and pre-trained language model BERT [Devlin & Chang$^+$ 19] can produce sentence representations. Given a sentence x_i, we can obtain sentence representation $f_\theta(x_i)$ generated by Word2vec or BERT after pooling.

Given a query q and a set of N documents D ($d_i \in D$), we can get their embeddings $f_\theta(q)$ and $f_\theta(d_i)$ through Word2vec or BERT. Cosine similarity could be used to calculate the score between query and document for embedding-based retrieval:

$$S(f_\theta(q), f_\theta(d_i)) = \frac{f_\theta(q) \cdot f_\theta(d_i)}{||f_\theta(q)|| \cdot ||f_\theta(d_i)||} \tag{2.2}$$

Then we can retrieve relevant documents for a specific query based on this score.

2.4 Contrastive Learning

Contrastive learning aims to learn effective representations by pulling semantically close neighbors together and pushing apart non-neighbors [Hadsell & Chopra$^+$ 06].

More specifically, the goal of contrastive representation learning is to learn such an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart [Weng 21]. This method has been widely used in computer vision to learn embeddings for images [Chen & Kornblith$^+$ 20] and in the NLP area to learn sentence embeddings. In this thesis, we focus on using contrastive learning methods to learn sentence embeddings in an unsupervised way. Many recent papers and studies [Gao & Yao$^+$ 21, Yan & Li$^+$ 21, Chuang & Dangovski$^+$ 22] prove that contrastive learning methods can boost the performance of sentence embeddings when training with unlabelled data.

2.4.1 General Framework

It assumes a set of paired examples $D = \{(x_i, x_i^+, x_i^-)\}_{i=1}^M$, for each training sample x_i in our batch of size N we have a positive sample x_i^+ and a negative sample x_i^- . More specifically, positive pairs (x_i, x_i^+) means x_i and x_i^+ are semantically related, whereas x_i^- is not semantically related to x_i. Let $f_\theta(x_i)$, $f_\theta(x_i^+)$ and $f_\theta(x_i^-)$ denote the embedding representation of x_i , x_i^+ and x_i^- after applying encoder model. Therefore, the goal of contrastive learning is to maximize agreement between $f_\theta(x_i)$ and $f_\theta(x_i^+)$ and minimize the agreement between $f_\theta(x_i)$ and $f_\theta(x_i^-)$. This framework is shown in Figure 2.5.

2.4.2 Loss Functions

To achieve the goal of contrastive learning, several different loss functions are used to train the encoder model. In some early versions of loss functions, they only consider one single positive sample x_i^+ and one single positive sample x_i^- for x_i. In this thesis we call these "Pairwise" losses:

Margin Loss [Chopra & Hadsell$^+$ 05]

$$\mathcal{L} = -\sum_{i=1}^{N} max(0, dist(f_\theta(x_i), f_\theta(x_i^+)) - \alpha) + \max(0, \beta - dist(f_\theta(x_i), f_\theta(x_i^-))) \quad (2.3)$$

In the Equation 2.3, N is number of samples, $f_\theta(x_i)$, $f_\theta(x_i^-)$ and $f_\theta(x_i^+)$ are corresponding sentence embeddings to sample x_i, negative sample x_i^- and positive sample x_i^+. $dist(\cdot, \cdot)$ is a distance measure metric between embeddings. Margin parameter α is configured as the higher bound distance between the embeddings of positive sample pairs (x_i, x_i^+), i.e. if the distance between the embeddings of the positive pairs is larger than α, only then there is a penalty in the loss. Also, margin parameter β is configured as the lower bound distance between the embeddings of negative sample pairs (x_i, x_i^-).

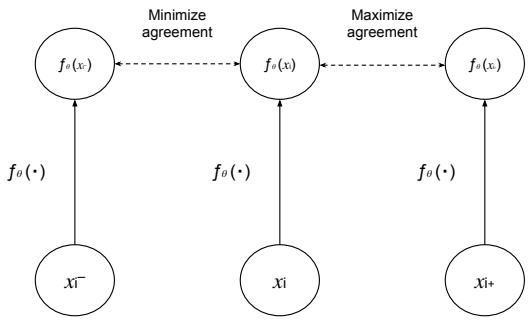


Figure 2.5: A simple framework for contrastive learning for sentence representations. x_i^+ is positive sample sentence to input sentence x_i and x_i^- is negative sample sentence to input sentence x_i. $f_\theta(x_i)$, $f_\theta(x_i^-)$ and $f_\theta(x_i^+)$ are corresponding sentence embeddings. A base encoder network $f_\theta(\cdot)$ is trained to maximize the the agreement between positive pairs and minimize the agreement between negative pairs using a contrastive loss.

The objective of this loss is to learn sentence representations with a smaller distance than the margin value α for positive pairs (x_i, x_i^+), and greater distance than margin value β for negative pairs (x_i, x_i^-), i.e. reduce the distance between positive pairs and increase the distance between negative pairs.

Triplet Loss [Schroff & Kalenichenko$^+$ 15, Li & Ma$^+$ 17]

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \max(0, -sim(f_\theta(x_i), f_\theta(x_i^+)) + sim(f_\theta(x_i), f_\theta(x_i^-)) + \epsilon) \tag{2.4}$$

In the Equation 2.4, N is the number of samples, $sim(\cdot, \cdot)$ is a similarity measure metric between embeddings. The margin parameter ϵ is configured as the minimum offset between distances of similar versus dissimilar pairs.

The objective of this loss is to learn sentence representations that maximize the similarity between the input sample x_i and positive sample x_i^+ and minimize the similarity between the input sample x_i and negative sample x_i^- at the same time.

Compared to margin loss [Chopra & Hadsell$^+$ 05] described before, maximizing the similarity between positive pairs is the same as minimizing the distance between positive pairs in margin loss, and vice versa. More specifically, if we use cosine similarity to measure distance in margin loss and to measure similarity in triplet loss, the common goal of margin loss and triple is to increase cosine similarity score between positive pairs and decrease cosine similarity score between negative pairs.

We also illustrate several loss functions which consider single positive sample x_i^+ and multiple negative samples X_i^- (single negative sample $x_i^- \in X_i^-$). In this thesis, we usually use **in-batch negatives** to construct a set of negative samples for contrastive learning during training, since it allows to efficiently to use more negative samples and reduce computation overhead during training. For a mini-batch of N sentence pairs $\{(x_i, x_i^+)\}_{i=1}^N$, x_i^+ is a positive sample for x_i and in-batch negatives $X_i^- = \{x_j, x_j^+\}_{j\neq i}^N$, i.e. we use all other samples except x_i and x_i^+ from the batch as negatives samples for sample x_i.

NT-Xent loss [Chen & Kornblith$^+$ 20]

Normalized Temperature-scaled Cross-Entropy Loss(NT-Xent loss), which uses categorical cross-entropy loss to identify the positive sample amongst a set of unrelated noise samples. The categorical cross-entropy loss used is as follows:

$$\mathcal{L}_{\text{cross-entropy}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{sim(f_\theta(x_i), f_\theta(x_i^+))}}{\sum_{x_j \in X_i^- \cup \{x_i^+\}} e^{sim(f_\theta(x_i), f_\theta(x_j))}} \tag{2.5}$$

where N is the number of input samples, X_i^- is a set of all negative samples from given input samples. $sim(f_\theta(x_i), f_\theta(x_i^+))$ refers to the cosine similarity in this thesis:

$$sim(f_\theta(x_i), f_\theta(x_i^+)) = \frac{f_\theta(x_i)^T f_\theta(x_i^+)}{||f_\theta(x_i)|| \cdot ||f_\theta(x_i^+)||} \tag{2.6}$$

We first give a set of paired examples $D = \{(x_i, x_i^+)\}_{i=1}^M$ where x_i and x_i^+ are input sentences which are semantically related. Then we take a cross-entropy objective with in-batch negatives X_i^-. Since in-batch negatives X_i^- are negatives samples for both x_i and x_i^+, we always have symmetric loss for x_i and x_i^+ respectively. The NT-Xent loss function for (x_i, x_i^+) in a mini-batch of N pairs is:

$$l_{(i,i^+)} = -\log \frac{e^{sim(f_\theta(x_i), f_\theta(x_i^+))/\tau}}{\sum_{x_j \in X_i^- \cup \{x_i^+\}} e^{sim(f_\theta(x_i), f_\theta(x_j))/\tau}} \tag{2.7}$$

Also the symmetrical loss for (x_i^+, x_i) is:

$$l_{(i^+,i)} = -\log \frac{e^{sim(f_\theta(x_i^+), f_\theta(x_i))/\tau}}{\sum_{x_j \in X_i^- \cup \{x_i\}} e^{sim(f_\theta(x_i^+), f_\theta(x_j))/\tau}} \tag{2.8}$$

where τ is a temperature hyperparameter which is used to tune how concentrated the features are in the representation space. The temperature plays a role in controlling the strength of penalties on the hard negative samples [Wang & Liu 21]. For instance, when temperature τ is low, the loss is dominated by the small distances and widely separated representations cannot contribute much and become irrelevant.

Thus, the training objective of NT-Xent loss for a mini-batch of N pairs is:

$$\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2N} \sum_{i=1}^{N} [l_{(i,i^+)} + l_{(i^+,i)}] \tag{2.9}$$

Binary Cross Entropy (BCE) loss

Given $f_\theta(x_i)$, $f_\theta(x_i^-)$ and $f_\theta(x_i^+)$ are corresponding sentence embeddings of sample x_i, negative sample x_i^- and positive sample x_i^+. X_i^- is a set of in-batch negative samples of x_i. The binary cross entropy loss is as follows:

$$\begin{aligned} \mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \big(&\log g(sim(f_\theta(x_i), f_\theta(x_i^+))) + \\ &\sum_{x_i^- \in X_i^-} \log(1 - g(sim(f_\theta(x_i), f_\theta(x_i^-))))\big) \end{aligned} \tag{2.10}$$

with $g(x) = \sigma(x^{\tau})$, where τ is a temperature hyperparameter which is used to tune how concentrated the features are in the representation space. $sim(f_\theta(x_i), f_\theta(x_i^+))$ is the cosine similarity.

2.5 Autoencoder

Autoencoder is a type of artificial neural network used to learn efficient representations of unlabeled data in an unsupervised manner [Kramer 91]. More specifically, the goal of the autoencoder is to learn a representation for a set of data, typically for dimensionality reduction [Fournier & Aloise 19], by training the neural network to capture the most important parts of the input data. Furthermore, autoencoders are applied to many tasks, like facial recognition [Hinton & Krizhevsky$^+$ 11], feature detection, and anomaly detection to acquire the meaning of words [Liou & Cheng$^+$ 14]. Autoencoders are also generative models, which can randomly generate new data which is similar to the input data.

An autoencoder has two main parts: an encoder that maps the input into compressed representations, and a decoder that maps the compressed representations to the reconstruction of the input. Figure 2.6 illustrates an example of general structure for autoencoder.

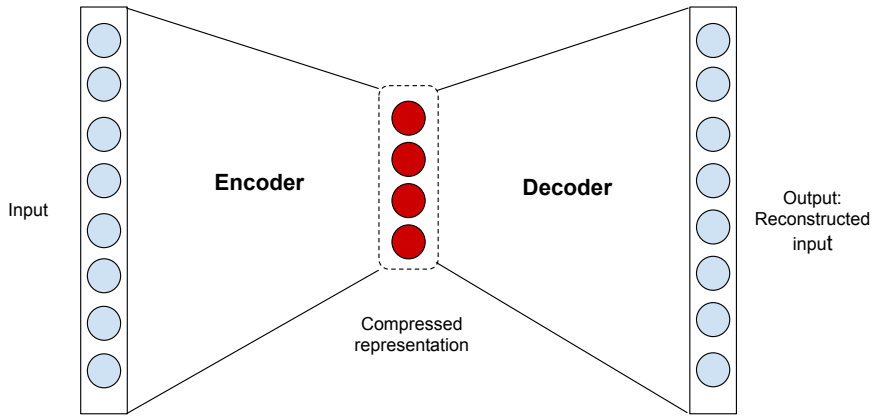


Figure 2.6: Schematic structure of an autoencoder. The encoder map the input into compressed representation, the decoder reconstructs the output based on the compressed representations.

Chapter 3

Methods

There are many novel methods focusing on learning sentence representations in a supervised or unsupervised way [Gao & Yao$^+$ 21, Wang & Reimers$^+$ 21, Chuang & Dangovski$^+$ 22, Yan & Li$^+$ 21]. In this chapter, we mainly focus on learning sentence representation using unsupervised methods. In this chapter, we will focus on two different types of unsupervised methods which are contrastive Learning and autoencoder-based methods.

3.1 Contrastive Learning

From the introduction of contrastive learning in Section 2.4, we find that the positive sample x_i^+ and negative sample x_i^- of the input sample x_i are the key ingredients when we use contrastive learning methods. Thus, the way to construct positive or negative samples plays a significant role in the whole sentence representation learning process. In this thesis, we mainly focus on unsupervised methods to learn sentence embeddings, which means we do not consider any human-annotated data. A simple and effective way is to use data augmentation to construct positive samples and use in-batch negatives at the same time. In this thesis, we consider two state-of-the-art methods (SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$ 21]) which use different data augmentation strategies to create positive samples.

3.1.1 SimCSE

SimCSE [Gao & Yao$^+$ 21] is a state-of-the-art method for contrastive learning to learn sentence embeddings. Pre-trained language models like BERT [Devlin & Chang$^+$ 19] can be used as encoder models to map input sentences into sentence embeddings after pooling. This kind of encoder model has a dropout [Srivastava & Hinton$^+$ 14] mechanism inside during the training. Dropout [Srivastava & Hinton$^+$ 14] is an effective method to avoid overfitting which randomly drops units (along with their connections) from the neural network during training in Section 2.1. So when we pass a sentence twice into the encoder model during training, the gener-

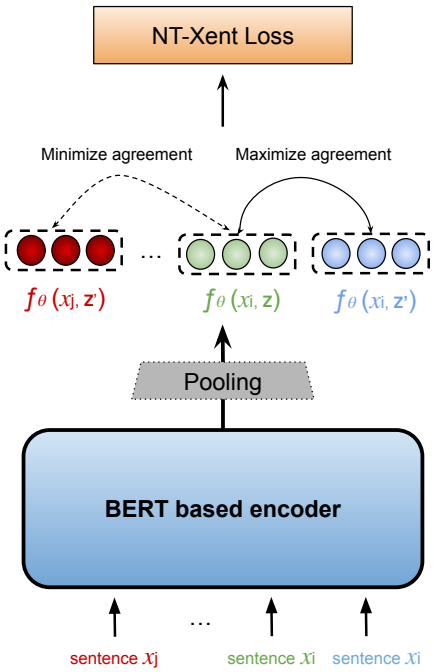


Figure 3.1: The general structure of SimCSE. Input sentences x_i and x_i are identical i.e. (x_i, x_i) is positive pair, (x_i, x_j) is negative pair. z, z' are independent dropout masks of BERT encoder.

ated embeddings for these two same sentences are different because of independent dropout.

The main idea of SimCSE [Gao & Yao+ 21] is to use dropout noise of the encoder model as data augmentation to generate positive pairs and select in-batch negatives. In SimCSE they give sentence pairs $D = \{(x_i, x_i^+)\}_{i=1}^M$ where $x_i^+ = x_i$, then they feed these identical positive pairs into BERT encoder model through the use of independent dropout masks z, z'. $f_\theta(x_i, z)$ represents embedding of x_i generated by the encoder model with dropout mask z. The model structure is shown in Figure 3.1.

For a mini-batch of N sentences they use non-symmetric NT-Xent loss as training objective:

$$\mathcal{L}_{\text{simcse}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{sim(f_\theta(x_i,z),f_\theta(x_i,z'))/\tau}}{\sum_{j=1}^{N}e^{sim(f_\theta(x_i,z),f_\theta(x_j,z'))/\tau}} \tag{3.1}$$

Note that z and z' are just the standard dropout masks in the BERT encoder model. τ is the temperature hyperparameter.

3.1.2 ConSERT

ConSERT [Yan & Li$^+$ 21] use different data augmentation strategies to construct positive pairs compared to SimCSE [Gao & Yao$^+$ 21]. The main idea is to use multiple text-based data augmentation methods to generate positive pairs. Figure 3.2 illustrates the general structure of ConSERT.

Given Sentence pairs $D = \{(x_i, x_i^+)\}_{i=1}^M$ where $x_i^+ = x_i$, the training process is as follows:

1. Use BERT model T as encoder, remove its default dropout
2. Apply token embedding layer of T to sentence pairs $\{(x_i, x_i^+)\}$ and get two same embedding matrices: $\{(e_i, e_i^+)\}$ where $e_i, e_i^+ \in \mathbb{R}^{L\times d}$ and $e_i = e_i^+$, L is the sequence length and d is the hidden dimension, and vector in each row represents embedding of every token
3. Apply different data augmentation strategies to $\{(e_i, e_i^+)\}$ to get new embedding matrices $\{(r_i, r_i^+)\}$
4. Feed $\{(r_i, r_i^+)\}$ to BERT model T to get final embedding $\{(f_\theta(x_i), f_\theta(x_i^+))\}$ through average pooling

In ConSERT [Yan & Li$^+$ 21] they mainly use four different data augmentation strategies in unsupervised learning, which are Token Shuffling, Token Cutoff, Feature Cutoff, and Embedding Dropout.

Token Shuffling aims to randomly shuffle the order of the tokens in the input sentence. Because in the BERT model the position ID is the only factor that contains sequential information. So they implement this strategy by passing the shuffled position ids to the embedding layer of encoder model T while keeping the order of the token ids unchanged.

Token Cutoff is a method which erases some tokens in the $L \times d$ embedding matrix e_i and e_i^+. They implement this method by randomly cut some rows of embedding matrix e_i and e_i^+ across dimension L.

Feature Cutoff is the other direction cutoff which is similar to token cutoff. Given embedding matrices e_i and e_i^+ where $e_i, e_i^+ \in \mathbb{R}^{L\times d}$, this method randomly cuts some columns of embedding matrix e_i and e_i^+ across dimension d.

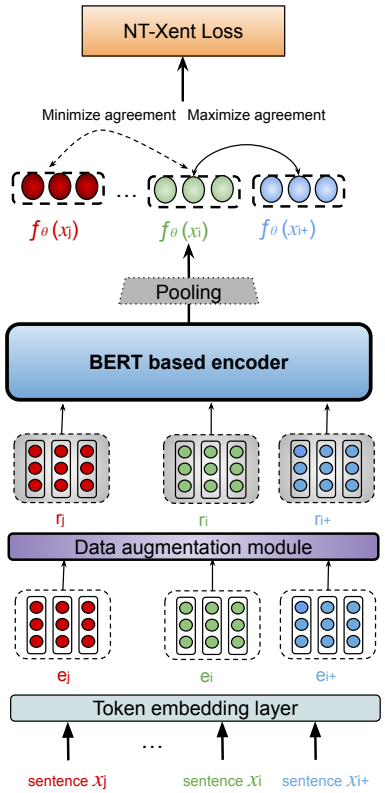


Figure 3.2: The general framework of ConSERT. Input sentences x_i and x_i^+ are identical i.e. (x_i, x_i^+) is positive pair, (x_i, x_j) is negative pair.

Embedding Dropout is a data augmentation method that randomly drops elements in the embedding matrix e_i and e_i^+ by a specific probability and set their values to zero. Note that this strategy is different from Cutoff since each element in the embedding matrix is considered individually. Also, this method is different from the dropout of the encoder model in SimCSE [Gao & Yao$^+$ 21], since embedding dropout is applied to the embedding matrix but dropout from SimCSE is applied to the encoder model.

For a mini-batch of N pairs using NT-Xent loss as training objective:

$$\mathcal{L}_{\text{consert}} = \mathcal{L}_{\text{NT-Xent}} \tag{3.2}$$

3.2 Autoencoder

In this thesis, we focus on using autoencoder-based methods to learn sentence representations. Autoencoder is a promising approach for controllable sentence generation by leveraging their latent sentence representations [Shen & Mueller+ 20], which gives us an idea to learn sentence embedding in an unsupervised way.

Autoencoder is based on Encoder-Decoder architecture, where encoder encodes the input data to a single embedding vector and decoder takes this embedding vector and tries to reconstruct the original input data. More specifically, in our case the encoder maps a sentence x_i to a new embedding $f_\theta(x_i)$ in a latent space (i.e. $f_\theta(x_i) \in R^h$ with h being the hidden dimension), the decoder reconstructs $f_\theta(x_i)$ into $d_\theta(f_\theta(x_i))$ which is in the same token-level of input sentence x_i [Kingma & Welling 14]. The goal of autoencoder is to make $d_\theta(f_\theta(x_i))$ and x_i identical. Thus, the mathematical goal of autoencoder is the maximize the probability of reconstructing input sentence x_i with decoder based on sentence embedding $f_\theta(x_i)$ generated by encoder.

So the training objective for a mini-batch of N sentences in Equation 3.3:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log P_\theta(x_i|f_\theta(x_i)) \tag{3.3}$$

where x_i is input sentence from batch N, the encoder maps x_i into embedding $f_\theta(x_i)$.

In this thesis, we consider two different autoencoders that are used to learn sentence embeddings, which are Causal Language Modeling (CLM) autoencoder and Masked Language Modelling (MLM) autoencoder.

3.2.1 CLM Autoencoder

Causal Language Modeling (CLM) autoencoder, is inspired by the pre-trained Transformers and Sequential Denoising AutoEncoder (TSDAE) approach [Wang & Reimers+ 21]. CLM autoencoder uses pre-trained language model BERT [Devlin & Chang+ 19] as encoder and uses multi-layer Transformer [Vaswani & Shazeer+ 17] (each layer of decoder is Transformer's encoder, always set 2 layers for decoder in this thesis) as a decoder. Thus, the CLM autoencoder combines the autoencoder method with a pre-trained language model for learning sentence embeddings. Furthermore, the decoder of CLM autoencoder is a conditional model which predicts tokens from left to right based on all left tokens.

An input sentence x_i will be fed into BERT encoder and get a new fixed-size sentence embedding $f_\theta(x_i)$ after pooling. Then the Transformer decoder tries to reconstruct input sentence x_i based on the embedding $f_\theta(x_i)$, and the output is

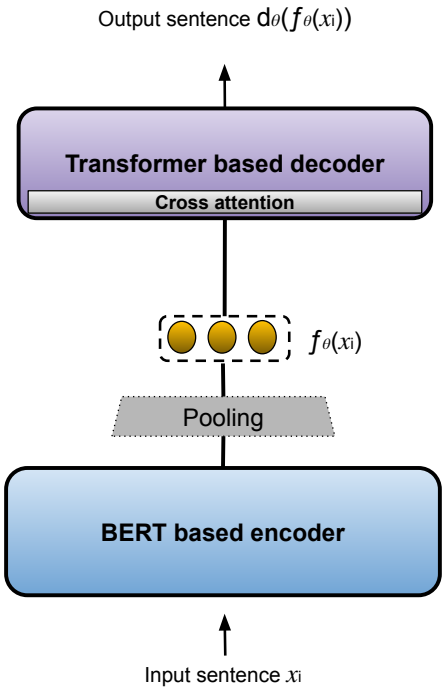


Figure 3.3: A general structure of CLM autoencoder.

$d_\theta(f_\theta(x_i))$. This structure is shown in Figure 3.3. Moreover, the architecture of CLM autoencoder is a modified encoder-decoder Transformer, where the key and value of the cross-attention are only confined to the sentence embedding [Wang & Reimers$^+$ 21]. The formulation of this revised cross-attention is:

$$H_k = \text{Attention}(H_{k-1}, [e^T], [e^T])$$
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

where $H_k \in \mathbb{R}^{t \times d}$ is hidden states of decoder within t decoding steps at the k-th layer, d is the dimension of the sentence embedding, $[e^T] \in \mathbb{R}^{1 \times d}$ is a matrix with

one row including the sentence embedding vector, Q, K and V are the query, key and value respectively.

The training objective of CLM autoencoder for a mini-batch of N sentences is illustrated in Equation 3.4:

$$\begin{aligned}\mathcal{L} &= -\frac{1}{N}\sum_{i=1}^{N}\log P_\theta(x_i|f_\theta(x_i)) \\ &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{t}\log P_\theta\left([x_i]_t \mid [x_i]_0^{t-1}, f_\theta(x_i)\right)\end{aligned} \tag{3.4}$$

where x_i is input sentence from batch N, the encoder maps x_i into a sentence embedding $f_\theta(x_i)$, $[x_i]_t$ is the t-th token of input sentence x_i , $[x_i]_0^{t-1}$ is a sequence of tokens $x_0 x_1 \dots x_{t-1}$ of input sentence x_i.

3.2.2 MLM Autoencoder

Inspired by the Mask Language Model (MLM) of BERT [Devlin & Chang$^+$ 19] in Section 2.2.2, our Masked Language Modelling (MLM) autoencoder model also involves a similar MLM technique to learn sentence embeddings in a specific way.

In the MLM autoencoder, we usually mask some tokens of an input sentence which are fed into the Transformer decoder model, and then the decoder model tries to predict these masked tokens. At first step, an input sentence x_i will be fed into BERT encoder to get a fixed-size sentence embedding $f_\theta(x_i)$ after pooling. Then we copy input sentence x_i and randomly mask some tokens (usually 80% in this thesis) to get a masked sequence $\tilde{x}_i$. We feed this masked sequence $\tilde{x}_i$ into the embedding layer of decoder to get embeddings $e_\theta(\tilde{x}_i)$ of all tokens (including masked and not masked tokens) of $\tilde{x}_i$, then we replace the first token embedding $[e_\theta(\tilde{x}_i)]_0$ (i.e. [CLS] token) with the sentence embedding $f_\theta(x_i)$ generated by BERT encoder to build final embeddings $f_\theta(x_i) + [e_\theta(\tilde{x}_i)]_1^n$, where n is the number of tokens in $\tilde{x}_i$. Finally, we feed the concatenated embeddings $f_\theta(x_i) + [e_\theta(\tilde{x}_i)]_1^n$ into the Transformer decoder (identical as decoder of CLM autoencoder) which is used to predict the masked tokens. Thus, the training objective of MLM autoencoder is different from CLM autoencoder. The structure of the MLM autoencoder is shown in Figure 3.4.

The training objective of MLM autoencoder for a mini-batch of N sentences is in Equation 3.5:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{m\in M_i}\log P_\theta([x_i]_m|f_\theta(x_i), x_i \setminus M_i) \tag{3.5}$$

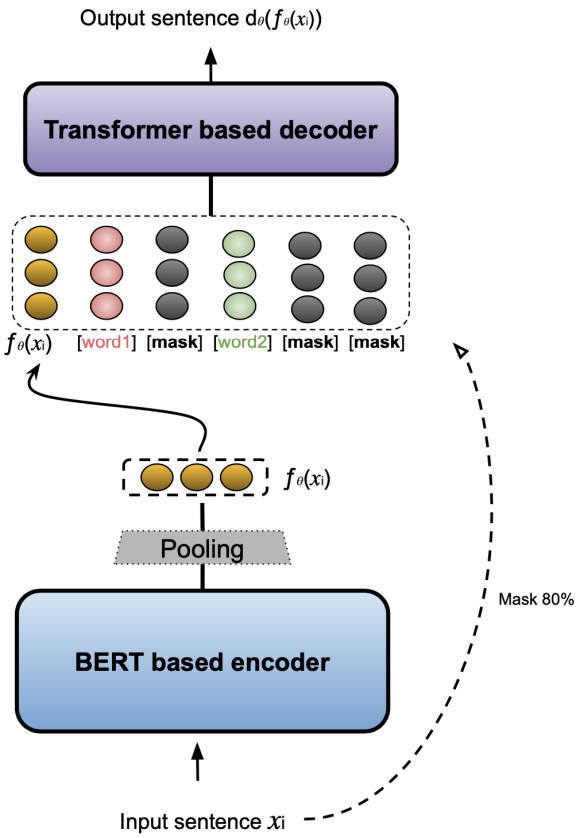


Figure 3.4: The general structure of MLM autoencoder. It masks the tokens of input sentence x_i by replacing the original token with [`mask`] token.

where x_i is input sentence from batch N, M_i $(m \in M_i)$is the set of masked tokens of x_i, $[x_i]_m$ is the original token from input sentence x_i of masked position m.

3.3 Multi-task Learning

Multi-task learning has led to successes in many applications of machine learning, from natural language processing and speech recognition to computer vision and recommendation systems [Ruder 17]. Multi-task learning is a subfield of machine learning where multiple learning tasks are solved at the same time while exploiting commonalities and differences across tasks. Furthermore, the multi-task learning method can learn tasks in parallel while using a shared representation, and what is learned for each task can help other tasks be learned better [Caruana 04]. More

specifically, multi-task learning needs only a single model trained to learn to complete multiple tasks at once, rather than training independent models for each task. In this process, this single model uses a set of training data to learn generalized representations of the training data that are useful in multiple learning tasks.

In this thesis, we aim to learn sentence representations by applying multi-task learning. It shows that contrastive learning methods [Gao & Yao$^+$ 21, Yan & Li$^+$ 21] can let the encoder model learn sentence similarity information, as well as the autoencoder-based methods, which can let the encoder model learn word-level information, since the decoder of CLM autoencoder tries to reconstruct input sentences word by word and MLM autoencoder tries to predict masked words. Therefore, one hypothesis is that the combination between contrastive learning and autoencoder could improve the performance of downstream tasks.

Figure 3.5 illustrates the framework of multi-task learning for the combination of contrastive learning and autoencoder-based methods. Given a batch of N sentences as input, (x_i, x_i^+) are positive pair and these two sentences are identical, (x_i, x_j) are negative pair and x_j is a sample of in-batch negatives of x_i. We feed these input sentences into BERT encoder model to get sentence embeddings $f_\theta(x_i)$, $f_\theta(x_i^+)$ and $f_\theta(x_j)$. Then feed $f_\theta(x_i)$, $f_\theta(x_i^+)$ and $f_\theta(x_j)$ into contrastive loss layer to calculate loss, and feed $f_\theta(x_i)$ into Transformer decoder to get output $d_\theta(f_\theta(x_i))$ then calculate autoencoder loss. We find $f_\theta(x_i)$ is the shared sentence representation for both contrastive learning and autoencoder. The goal of this multi-task learning is to make sentence embedding $f_\theta(x_i)$ learn from both contrastive learning and autoencoder methods, in order to improve the quality of sentence embedding $f_\theta(x_i)$.

In Figure 3.5 we train the encoder model using this multi-task approach so that the encoder can learn the objective of contrastive learning methods and the objective of autoencoder-based methods simultaneously by optimizing the combined loss between contrastive loss and autoencoder loss at once. The combined loss function for multi-task is shown in Equation 3.6.

$$\mathcal{L}_{\text{multi-task}} = \mathcal{L}_{\text{NT-Xent}} + w \times \mathcal{L}_{\text{autoencoder}} \tag{3.6}$$

where $\mathcal{L}_{\text{NT-Xent}}$ is the contrastive learning loss, $\mathcal{L}_{\text{autoencoder}}$ indicates the loss of autoencoder-based methods and w is the weighted coefficient for autoencoder loss.

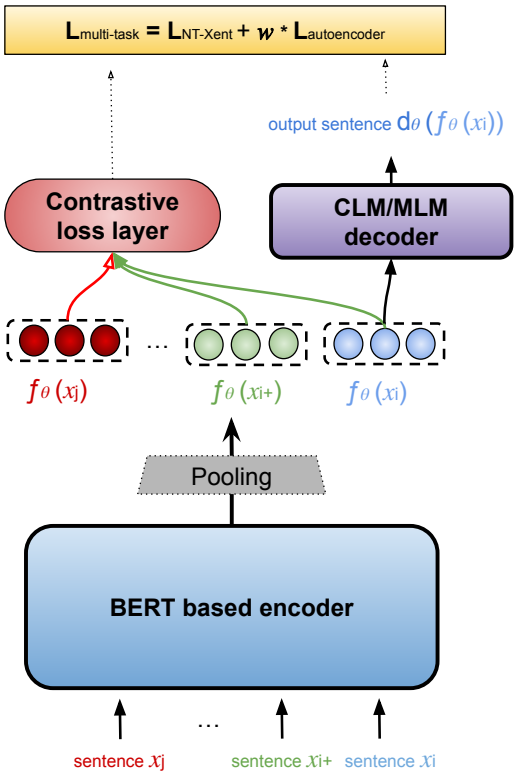


Figure 3.5: Multi-task learning approach for the combination between contrastive learning and autoencoder. The red x_j represent negative samples of x_i, the green x_i^+ represents positive samples of x_i. $f_\theta(x_i)$ is the shared sentence representation for both contrastive learning and autoencoder. The loss of multi-task learning approach is a combination of contrastive learning loss and autoencoder loss.

Chapter 4

Evaluation

The crucial shortcoming of previous studies [Gao & Yao+ 21, Yan & Li+ 21, Chuang & Dangovski+ 22] is the narrow evaluation, which indicates most methods mainly evaluate sentence representations on a single task Semantic Textual Similarity (STS) from SemEval [Conneau & Kiela 18]. However, the performance of STS task can not prove these sentence representations also have similar performance on other downstream tasks, like eProduct [Yuan & Chiang+ 21] and duplicate questions retrieval [Thakur & Reimers+ 21] in this thesis. To clarify whether the sentence representation generalizes well on relevant downstream tasks, we involve two different domains specific tasks, which are eProduct [Yuan & Chiang+ 21] challenge and duplicate question retrieval [Thakur & Reimers+ 21] (Quora and CQADupStack).

4.1 Evaluation Tasks

4.1.1 eProduct

eProduct challenge [Yuan & Chiang+ 21] is a retrieval task, which is eBay's internal large-scale products similarity search benchmark. The datasets of eProduct have been collected from eBay e-commerce platform which contains a query set and an Index set, and each item in the index set has a human-annotated binary label "match" or "not match" when given a query from the query set. Besides, each item from query and index set is provided with an image, a title, and/or attributes, and categorical IDs. We only use item titles to do text-based retrieval tasks in this thesis.

More specifically, we give query set Q and index set D ($d_i \in D$), both are eBay item titles, i.e. a sequence of words. For each query $q_i \in Q$, the task of eProduct is to retrieve top 10 relevant documents d_i from D. Table 4.1 gives an example of query and its similar documents match.

Figure 4.1 gives an evaluation pipeline on eProduct task. We try to find the top 10 similar documents for each query q_i based on embedding-based retrieval. After that, we use evaluation metrics on these retrieved results to get evaluation feedback to see if the trained encoder model performs well or not.

Query	Cisco 5500 Series Wireless Controller, model AIR-CT5508-K9, active licenses
Relevant documents	Cisco 5500 Series Wireless Controller AIR-CT5508-K9 25 AP License Cisco 5500 Series Wireless Controller AIR-CT5508-K9 50 AP License Cisco 5500 Series AIR-CT5508-K9 5508 Wireless LAN Controller 25 AP License Cisco 5500 Series AIR-CT5508-K9 Wireless Controller Model 5508 *Tested*

Table 4.1: Example of eProduct challenge. Given a query q_i from query set, this table shows the top4 relevant documents from Index set D.

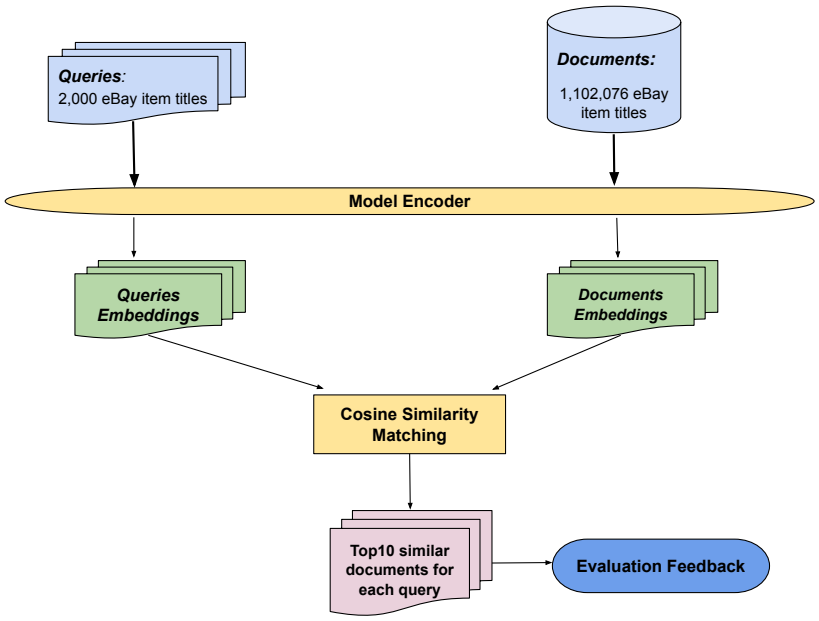


Figure 4.1: eProduct challenage evaluation diagram. Quries are from query set and documents are from index set.

Quora	
Query	How long does it take to methamphetamine out of your blood?
Relevant Document	How long does it take the body to get rid of methamphetamine?
CQADupStack	
Query	Command to display first few and last few lines of a file <*Title*>Combing head and tail in a single call via pipe
Relevant Document	<*Body*>On a regular basis, I am piping the output of some program to either 'head' or 'tail'. Now, suppose that I want to see the first AND last 10 lines of piped output, such that I could do something like ./lotsofoutput — headtail...

Table 4.2: Example of Duplicate Question Retrieval with one query and one relevant document. (<*Title*>) and (<*Body*>) are used to distinguish the title separately from the paragraph within a document in the table above.

4.1.2 Duplicate Question Retrieval

To evaluate the quality of sentence embedding generated from the model, duplicate question retrieval [Thakur & Reimers$^+$ 21] is also a valuable evaluation task. Duplicate question retrieval is the task of identifying duplicate or similar questions. In this thesis, we utilize two duplicate question retrieval tasks Quora [Thakur & Reimers$^+$ 21]and CQADupStack [Hoogeveen & Verspoor$^+$ 15] from different domains. The task of duplicate question retrieval is: given a question q as an input query, retrieve top k similar questions from questions candidate set as output, which is similar to eProduct challenge [Yuan & Chiang$^+$ 21]. Examples of Quora and CQADupStack retrieval are shown in Table 4.2. An evaluation pipeline is shown in Figure 4.2.

Quora

The Quora datasets are from the question-answers platform Quora [1] which identify whether two questions are duplicates. It has a query set and a candidate set, both consist of questions from Quora. When given a question from the query set as an input query, each question in the candidate set has a human-annotated binary label to indicate if this question is duplicated or not for the given query.

[1] Quora `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

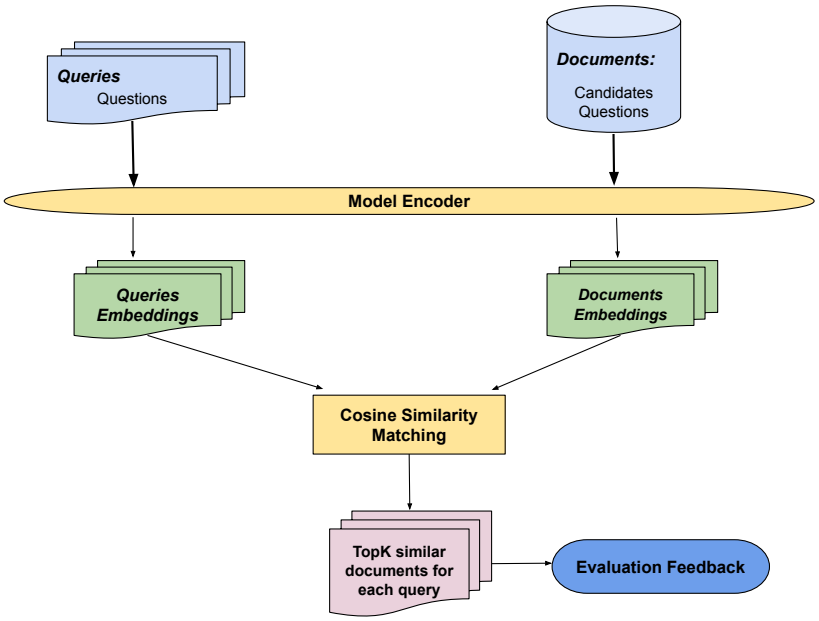


Figure 4.2: Duplicate Question Retrieval evaluation diagram. Both queries and documents are questions.

CQADupStack

CQADupStack [Hoogeveen & Verspoor+ 15] is a benchmark dataset for community question-answering (cQA) research[2]. It contains threads from twelve StackExchange subforums, annotated with duplicate question information. Stackexchange is a collection of 149 question-answering communities (subforums) on a wide range of topics. The CQADupStack datasets also have a query set and a candidate set, each item from the query set is a question title, and each item from the candidate set is a full question title + body (detailed description). Furthermore, each item in the candidate set has a human-annotated binary label to indicate whether this candidate is duplicated or not for the given query.

[2]CQADupStack `http://nlp.cis.unimelb.edu.au/resources/cqadupstack/`

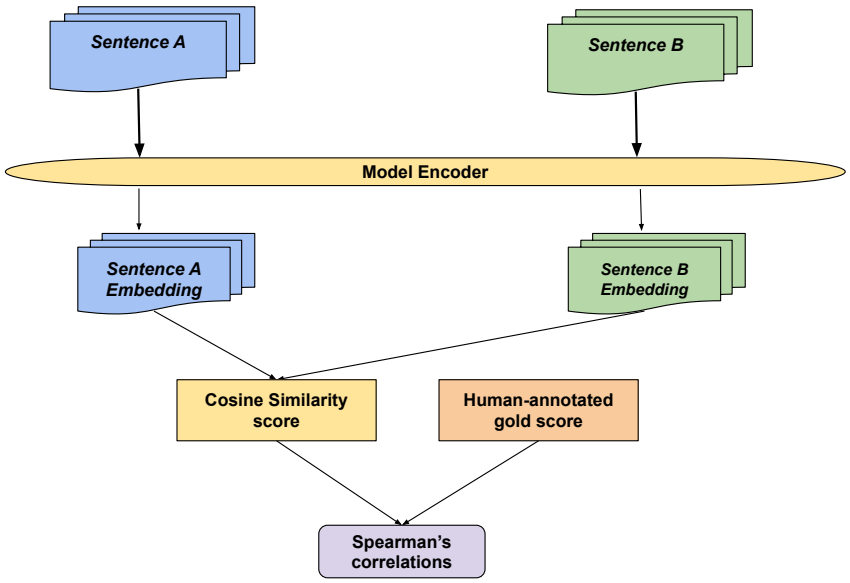


Figure 4.3: STS evaluation diagram. Sentence A and Sentence B are one pair from STS task dataset. We report average Spearman's correlation on all sentence pairs from STS task dataset.

4.1.3 Semantic Textual Similarity

Semantic Textual Similarity (STS) task can be used to measure the semantic similarity of sentences. The STS datasets include pairs of sentences taken from news articles, forum discussions, news conversations, headlines, images, and video descriptions labeled with a similarity score between 0 and 5. Examples of sentences from STS datasets are shown in Table 4.3. The goal of STS is to evaluate how a semantic similarity score between two sentences calculated by a trained model correlates with a human-labeled similarity score.

In this thesis, the evaluation dataset includes STS tasks from 2012 [Agirre & Cer$^+$ 12], 2013 [Agirre & Cer$^+$ 13], 2014 [Agirre & Banea$^+$ 14], 2015 [Agirre & Banea$^+$ 15], 2016 [Agirre & Banea$^+$ 16] and STS-Benchmark [Cer & Diab$^+$ 17].

Sentence pairs	Similarity score
The black dog is running through the snow. A race car driver is driving his car through the mud.	0
The woman is playing the violin. The young lady enjoys listening to the guitar.	1
They flew out of the nest in groups. They flew into the nest together.	2
John said he is considered a witness but not a suspect. “He is not a suspect anymore.” John said.	3
Two boys on a couch are playing video games. Two boys are playing a video game.	4
The bird is bathing in the sink. Birdie is washing itself in the water basin.	5

Table 4.3: Example of Sentence pairs with Similarity scores from STS 13 [Agirre & Cer$^+$ 13].

The STS Benchmark is a careful selection from the corpus of English STS shared task data (2012-2017) [Cer & Diab$^+$ 17].

4.2 Evaluation Metrics

4.2.1 eProduct

eProduct challenge simulates a real-world information retrieval task in eBay. Given a query title q_i, the goal is to retrieve top k relevant titles from the million-scale Index set. The major criterion is Capped Recall@k (R_{cap}@k) and Precision@k (P@k), see Equation 4.1. We set $k = 10$ for eProduct challenge same as the original paper [Yuan & Chiang$^+$ 21].

$$\begin{aligned} R_{cap}@k &= \frac{1}{N}\sum_{i=1}^{N}\frac{r_{i@k}}{g_{i@k}} \\ P@k &= \frac{1}{N}\sum_{i=1}^{N}\frac{r_{i@k}}{k} \end{aligned} \tag{4.1}$$

where N is the total number of queries, $r_{i@k}$ is the number of titles retrieved from k are true match for query title i, $g_{i@k}$ is the capped number of groundtruth matches for title i. Note that the number of groundtruth matches for a given query is at least 1 and could be greater than k, thus $g_{i@k} \in [1, k]$, i.e. $g_{i@k} = min(k, \#\text{of groundtruth matches for query } i)$.

Given a query i, Capped Recall@k (R_{cap}@k) is the proportion of true match items found in the top-k recommendations, Precision@k (P@k) is the rate of correct match items found in the retrieved top-k recommendations. This evaluation setting is similar to a real production scenario where a product might have many more matches but we are usually focused on the recalls at the top-k's.

4.2.2 Duplicate Question Retrieval

Given a question q from the query set, the goal is to retrieve the top k duplicate questions from a candidate set. The major criterion is Recall@k (R@k) and Precision@k (P@k), see Equation 4.2. We evaluate duplicate question retrieval with different k, i.e. $k \in \{1, 3, 5, 10, 100, 1000\}$.

$$\begin{aligned} R@k &= \frac{1}{N}\sum_{i=1}^{N}\frac{r_{i@k}}{g_i} \\ P@k &= \frac{1}{N}\sum_{i=1}^{N}\frac{r_{i@k}}{k} \end{aligned} \tag{4.2}$$

where N is the total number of queries, $r_{i@k}$ is the number of questions retrieved from k are true matches for query question i, g_i is the number of groundtruth matches for query i.

4.2.3 Semantic Textual Similarity

In this thesis, we report Spearman's correlation between the cosine similarity of the sentence representations and the human-annotated gold scores for STS-Benchmark (STS-B) and STS-Avg. (weighted average score for STS12, STS13, STS14, STS15, STS15, STS16).

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.3}$$

where ρ is the Spearman's correlation coefficient, n is the number of samples, d_i is the pairwise distances between cosine similarity score based on sentence embeddings and the human-annotated gold score of i-th sentence pair, i.e. $d_i = |\text{Score}_i^{\text{cosine}} - \text{Score}_i^{\text{human}}|$.

eProduct	*Number of Titles*	*Words per title*
Query@dev	2,000	10.97
Index	1,102,076	10.73

Table 4.4: eProduct evaluation dataset statistics, only consider item titles.

4.3 Evaluation Datasets Statistics

Section 4.1 illustrates detailed descriptions of all three evaluation tasks, which are eProduct, Semantic Textual Similarity (STS), and duplicate question retrieval. This section will focus on the datasets statistics of these evaluation datasets.

4.3.1 eProduct

eProduct challenge [Yuan & Chiang+ 21] is eBay internal retrieval task based on product titles from the eBay e-commerce platform. The dataset consists of about 1.2 million selected listings from the data on the public eBay site. All data have been split into two subsets: Query set and Index set. A query is a given item associated with an image, a title and/or attributes. The query set consists of 5,000 items, which have been split into devset 2,000 of them and testset 3.000 of them. We only consider using devset 2,000 queries in the following experiments. And all the items from the Index set will be used as the target retrieval data. Each item is provided with an image, a title, and/or attributes, and categorical IDs. In this thesis, we only use item titles from the Query set and Index set to do text-based retrieval tasks.

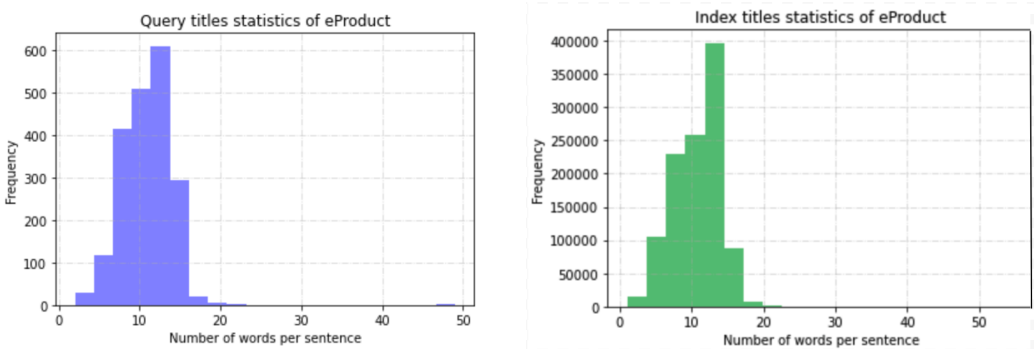


Figure 4.4: The frequency of sentence length of evaluation datasets eProduct.

	Quora	CQADupStack
#Queries	10,000	13,145
Avg. Query Lengths	9.53	8.59
#Documents	522,931	457,199
Avg. Document Lengths	11.44	129.09
Avg. D / Q	1.6	1.4

Table 4.5: Quora and CQADupStack statistics. Avg. Query/Document length indicates the average number of words per query/document. Avg. D/Q indicates the average relevant documents per query.

4.3.2 Duplicate Question Retrieval

In this thesis, we use Quora and CQADupStack as two different domains to duplicate question retrieval evaluation tasks [Thakur & Reimers$^+$ 21]. The datasets statistics are shown in Table 4.5. Quora duplicate questions dataset identifies whether two questions are duplicates, which has been split into the train, dev, and test sets with a ratio of about 85%, 5%, and 10% of the question pairs. All overlaps between the splits have been removed to ensure that a question in one split of the dataset does not appear in any other split. In this thesis, we report the results of the evaluation on the Quora test set. The corpus of CQADupStack [Hoogeveen & Verspoor$^+$ 15] contains queries from 12 different StackExchange subforums: Android, English, Gaming, Gis, Mathematica, Physics, Programmers, Stats, Tex, Unix, Webmasters, and Wordpress. This evaluation dataset utilizes the original test split for queries, and the task involves retrieving a duplicate query (title + body) given an input query title. In this thesis, we report the average of the results for these 12 subforums.

4.3.3 Semantic Textual Similarity

From the previous chapters, we know that Semantic Textual Similarity (STS) [Conneau & Kiela 18] is also a widely used public task to evaluate sentence representation [Yan & Li$^+$ 21, Gao & Yao$^+$ 21, Chuang & Dangovski$^+$ 22, Wang & Reimers$^+$ 21]. The statistics of this dataset are shown in Table 4.6. Each sample in these datasets contains a pair of sentences as well as a gold score between 0 and 5 to indicate their semantic similarity. In this thesis, we use STS-Avg to indicate the collection of datasets STS12, STS13, STS14, STS15, and STS16. Also, we report Spearman's correlation score of STS-Avg and STS-B.

	STS12	STS13	STS14	STS15	STS16	STS-B
Number of train samples	0	0	0	0	0	5749
Number of valid samples	0	0	0	0	0	1500
Number of test samples	3108	1500	3750	3000	1186	1,379

Table 4.6: Datasets statistics of Semantic Textual Similarity.

Chapter 5

Experiments and Analysis

In this chapter, we focus on the experiments and analysis. First, we give the overview statistics of training/validation datasets and evaluation datasets. Then we give more details about baseline methods and basic experimental setup for the following experiments, including different methods used, hyperparameters setting, training dataset selection, and base models. Finally, we introduce different experiments, which include the goal, motivation, setting, and results of the experiment, and do an analysis of the experiments' results.

5.1 Training and Validation Datasets Statistics

In this thesis, we use different domains of the corpus as training dataset and validation datasets which only contains unlabelled sentences for unsupervised learning. We sample OpenWebText 1M/100k/10k from OpenWebTextCorpus [Gokaslan & Cohen] which is web text content extracted from URLs. eBay Titles 1M/100k/10k have been sampled from eBay's internal database, which is eBay product item titles, i.e each title is a list of words. eProduct titles 1M/100k have been sampled from the eProduct task's query and Index dataset [Yuan & Chiang$^+$ 21], which are similar to eBay titles. Quora titles 100k/10k have been sampled from query and documents of Quora evaluation dataset [Thakur & Reimers$^+$ 21]. Generally, the datasets with 10k size are used as validation datasets during training, and the other sizes of datasets are used as training datasets. More details of datasets statistics are shown in Table 5.1 and Table 5.2.

To explore more details about these different domain training datasets, we analyze the frequency of sentence length of each dataset in Figure 5.1. The figure only depicts the datasets with 100,000 sentences, the other sizes of datasets in each domain also have similar statistics. We observe most sentences from OpenWebText 100k contain about 20 (+5 or -5) words, and most sentences in Quora titles 100k contain about 10 words. Besides, we find eBay titles 100k and eProduct titles 100k have similar statistics distribution in the figures, since they have been sampled from the same domain.

	Sentences	*Words per sentence*	*Running words*	*Vocabularies*
OpenWebText 1M	1,000,000	19.85	19,857,849	866,206
OpenWebText 100k	100,000	19.85	1,985,570	190,191
eBay Titles 1M	1,000,000	10.90	10,908,538	774,092
eBay Titles 100k	100,000	10.90	1,090,510	158,338
eProduct titles 1M	1,000,000	10.62	10,625,764	592,624
eProduct titles 100k	100,000	10.64	1,064,034	128,080
Quora titles 100k	100,000	11.41	1,141,440	85,014

Table 5.1: Training datasets statistics

	Sentences	*Words per sentence*	*Running words*	*Vocabularies*
OpenWebText 10k	10,000	19.82	198,286	40,507
eBay Titles 10k	10,000	10.97	109,793	32,727
eProduct titles 10k	10,000	10.83	108,300	27,893
Quora titles 10k	10,000	11.52	115,206	20,038

Table 5.2: Validation datasets statistics

5.2 Experimental Setup

5.2.1 Baselines

We use several methods which can produce sentence embeddings as the baselines to see how much improvement can be achieved by using other proposed methods.

Pre-trained Language Models

Bert-base-uncased [Devlin & Chang+ 19] is is a Transformers model pretrained on a large corpus of English data in a self-supervised fashion[1]. eBert is an eBay internal model which has the identical model architecture to Bert-base-uncased but with different training corpus.

We use these two pre-trained language model checkpoints shared by the authors, and feed the sentences into these models using average pooling in order to obtain sentence embeddings. Then we calculate the similarity scores between queries and documents based on these generated sentence embeddings when doing evaluation tasks eProduct [Yuan & Chiang+ 21] and duplicate questions retrieval [Thakur & Reimers+ 21].

[1] `https://huggingface.co/bert-base-uncased`

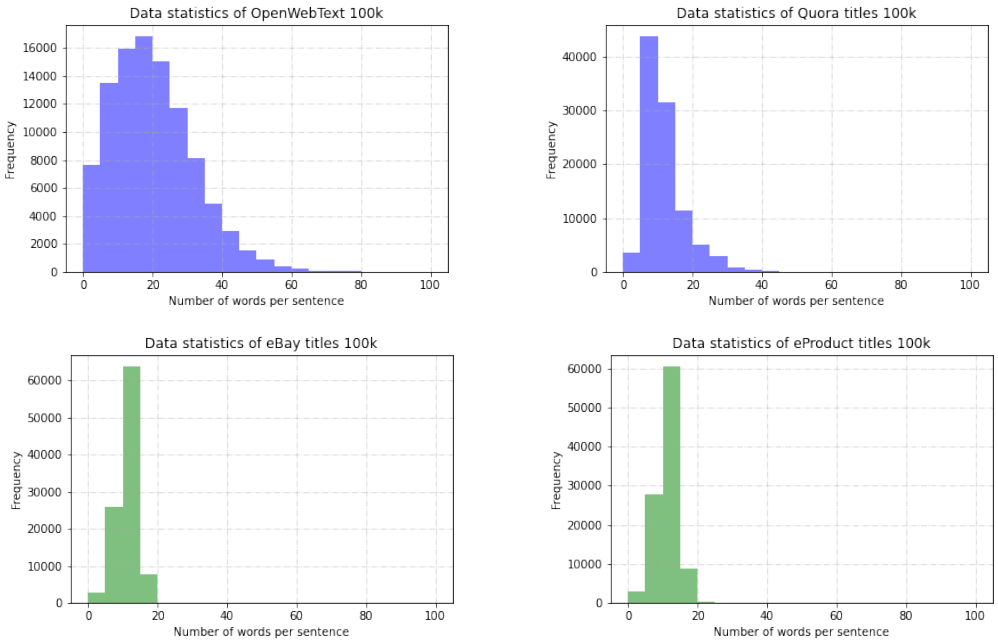


Figure 5.1: The frequency of sentence length of different domains training datasets. To make a fair observation, we plot datasets with 100,000 sentences.

Word2vec

In this thesis, we train a Word2vec model on the downstream tasks datasets, i.e. train Word2vec model with eProduct [Yuan & Chiang+ 21] evaluation dataset if doing an evaluation on eProduct task. We use a toolkit `gensim` [Řehůřek & Sojka 10] to train this model with skip-gram approach[2]. More specifically, we train Word2vec for eProduct task using all titles from query set and index set (over 1,100,000 titles) from eProduct evaluation dataset as training corpus. We embed each word of eProduct titles into a 100-dimensional embedding. We set window (contains center word and context words) size as 5 and train the model with 5 epochs.

To obtain sentence embeddings, we use average pooling for all word embeddings in a sentence generated by the trained Word2vec model. Then we use this average-word sentence embedding to do downstream tasks.

[2] `https://radimrehurek.com/gensim/models/word2vec.html`

BM25

We use `Pyserini` [Lin & Ma+ 21] implemented BM25 as baseline[3], and using default parameters $k = 0.9, b = 0.4$ according to Equation 2.1 to calculate BM25 scores between query and documents. On the downstream evaluation retrieval tasks, we use BM25 scores to measure similarity between queries and documents.

5.2.2 Basic Experimental Setup

In this section, we give the basic experimental setup for the contrastive learning methods and autoencoder-based methods, instead of baseline methods. In many experiments some parameters could be different from the basic experimental setup, we will point it out at the beginning of each experiment section.

- The training and validation datasets are from OpenWebText, eBay titles, or Quora titles with different sizes.
- Besides the validation datasets we have two main validation tasks during training: STS-Benchmark(devset) evaluation with Spearman's correlation, Quora (devset) evaluation with recall@10 & precision@10.
- The pre-trained base model used is Bert-based-uncased [Devlin & Chang+ 19] or eBert model. We usually use the eBert model as a base model when doing an evaluation of eBay internal task eProduct [Yuan & Chiang+ 21].
- For the best model checkpoint selection we use validation loss i.e. loss on validation dataset as default. The domain of the validation dataset is consistent with the training dataset. (e.g. if use eBay titles 100k/1M as the training dataset, the validation dataset should be eBay titles 10k.) We evaluate the model checkpoint every 250 steps during training.
- If train the model with multiple epochs, we shuffle the whole training dataset before each epoch.
- We run every experiment three times with three different random seeds $(1, 7, 42)$, and report the average results on these 3 different random seeds as final results.
- Some other important hyperparameters settings are shown in Table 5.3.

[3] `https://github.com/castorini/pyserini`

Hyperparameters	Value
epochs	1
batch size	64
warmup steps	910
temperature	0.05
weight decay	0.0
max seq. length	32
learning rate	3e-5
pooling	avg.
decoder layers	2

Table 5.3: Basic Experiments Hyperparameters Setup. Temperature plays a part if and only if use contrastive learning methods, and decoder layers works when use autoencoder based methods. Besides, we only consider [CLS] token or average as the pooling strategy in following experiments.

5.2.3 Training Details

We implement all methods with `transformers` package [Wolf & Debut$^+$ 20], and build the implementation on top of existing SimCSE source code[4]. We conduct every model training with a single TESLA V100 GPU. Additionally, since we don't have access to the ground truth of eProduct evaluation dataset, we can only evaluate it via an eBay internal online evaluation interface to obtain results. For duplicate question retrieval task Quora and CQADupstack, we use `beir` package[5] to do evaluations. Also using `SenEval`[6] to do evaluation on Semantic Textual Similarity (STS) tasks.

5.3 Results and Analysis

In this section, we focus on different experimental results and analyze these results. Each subsection aims to investigate the research questions proposed in Introduction chapter.

5.3.1 Comparison of SOTA Methods for Contrastive Learning

This experiment aims to compare the current state of the art methods SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$ 21] on our eBay internal task eProduct [Yuan

[4] `https://github.com/princeton-nlp/SimCSE`
[5] `https://github.com/beir-cellar/beir`
[6] `https://github.com/facebookresearch/SentEval`

Model	eProduct[%]		STS-B[%]	STS-Avg[%]
	R_{cap}@10	P@10		
ConSERT$_1$	68.7	61.6	62.1	58.4
ConSERT$_2$	69.1	61.9	60.6	59.4
SimCSE	69.1	61.9	61.9	56.7
eBert	50.9	45.9	61.2	57.9
BM25	73.1	65.8	-	-
Word2vec	48.4	43.7	-	-

Table 5.4: Comparison results between ConSERT and SimCSE. ConSERT$_1$ uses *Token Shuffling + Feature Cutoff* combination and ConSERT$_2$ uses *Token Shuffling + Embedding Dropout* combination. The score of STS is Spearman's correlation. We report the scores averaged over 3 random seeds.

& Chiang$^+$ 21] as well as Semantic Textual Similarity (STS) [Conneau & Kiela 18]. We use the pre-trained eBert model as the base model and fine-tune this model by using eBay titles 100k as the training dataset. Also, we use validation loss as the best model checkpoint selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3.

We follow the approaches from the original papers. SimCSE [Gao & Yao$^+$ 21] uses the default dropout of the model with probability 0.1 as a simple data augmentation to create positive samples. ConSERT [Yan & Li$^+$ 21] uses two different data augmentation combinations *Token Shuffling + Feature Cutoff* and *Token Shuffling + Embedding Dropout* to create positive samples, which are proved to have the best performance compared to other data augmentation strategies according to the original paper. We denote these two combinations as ConSERT$_1$ and ConSERT$_2$ respectively. In ConSERT we use *Feature Cutoff* with probability 0.2 and *Embedding Dropout* with probability 0.2. We also use a pre-trained model eBert with average pooling as our baseline to do the evaluation. The comparison results are shown in Table 5.4.

We observe that both ConSERT and SimCSE can achieve a huge improvement in performance on the eProduct task with almost 20% in Capped Recall@10 when compared to base model eBert in Table 5.4. Besides, all these three different data augmentation strategies have close performance on eProduct and STS tasks. We can conclude that fine-tuning the base model with an in-domain training dataset (eBay titles) using contrastive learning methods as well as appropriate data augmentation strategies can boost the quality of sentence embedding on the eBay internal evaluation information retrieval task.

Model	Random seeds	eProduct[%]		STS-B[%]	STS-Avg[%]
		R_{cap}@10	P@10		
ConSERT_2	seed 1	69.0	61.7	55.4	55.1
	seed 7	68.8	61.7	63.3	60.2
	seed 42	69.3	62.2	63.1	63.0
SimCSE	seed 1	69.4	62.1	66.6	60.7
	seed 7	68.2	61.1	55.8	50.9
	seed 42	69.6	62.5	63.2	58.4

Table 5.5: Results of ConSERT and SimCSE with different random seeds. ConSERT_2 uses *Token Shuffling + Embedding Dropout* combination. The score of STS is Spearman's correlation.

Furthermore, it shows that the performance on the STS tasks does not correlate with our downstream task performance in Table 5.4, which means an approach working well on the STS tasks did not be a good choice for the eProduct task, and vice versa. Also, we can see that even though there is a huge improvement in eProduct using contrastive learning methods, the scores of STS tasks are almost unchanged when compared to our baseline model eBert. There are multiple possible reasons for this difference. The first reason is that there is a mismatch between domain of STS datasets and domain of eBay internal datasets. Thus, STS task does not benefit from the additional training on eBay training datasets. The second reason is that the STS task consists of sentence pairs with human-annotated gold scores between 0 and 5. So the model requires to give each sentence pair a score in order to rank dissimilar pairs and similar pairs equally well. However, most real-world tasks, like eProduct task, just require identifying similar pairs out of a large-scale candidates pool.

In Table 5.5 we observe quite different behavior when evaluating STS tasks compared to evaluating our domain-specific tasks eProduct with different random seeds. Both ConSERT_2 with random seed 1 and SimCSE with random seed 7 have relatively lower results on the STS task when compared to other random seeds. Whereas both methods have much more stable results on eProduct with these three random seeds. It is clear that different random seeds can produce different results since the dropout of SimCSE and shuffle & cutoff of ConSERT depend on them, which are vital for creating positive samples. Moreover, STS tasks are more sensitive to random seeds than eProduct, which can support our previous conclusion that performance on STS tasks is not correlated to domain-specific task eProduct. Thus, the STS task is not a good way to evaluate the quality of sentence embedding when we do domain-specific tasks based on sentence representations.

Query	Vintage Fisher-Price Elephant Rattle Baby Toy Take Along Crib Stroller #619
BM25	Vintage Fisher-Price Elephant Rattle Baby Toy Take Along Crib Stroller #619 Blue Cute Baby Crib Stroller Rattles Seat Take Along Travel Arch Development Toys Fisher-Price Ocean Wonders Take-Along Projector Soother Baby - Kids Toy New
SimCSE	Vintage Fisher-Price Elephant Rattle Baby Toy Take Along Crib Stroller #619 Blue VINTAGE FISHER PRICE LOOK AT ME ELEPHANT RATTLE #429-1977 Vintage Baby Rattle Toy 1988 Discovery Toys elephant learning toy infant

Table 5.6: Compare BM25 with SimCSE (random seed 42 with training dataset eBay titles 100k) on eProduct retrieval results. Given a query, retrieve top 3 relevant documents using BM25 and SimCSE methods. In this case, SimCSE outperforms BM25.

Query	ASICS GEL-Sonoma 3 Trail Running Shoes - Navy - Womens
BM25	Asics Gel Sonoma 3 Women's T774N 9667 Shoes Trail Running Grey Aqua Asics Women's GEL - Sonoma Trail Running Shoe - Assorted Sizes & Colors Asics Gel Sonoma 2 Running Sneakers Womens Trail Shoes Flat Heel
SimCSE	Asics Women's GEL - Sonoma Trail Running Shoe - Assorted Sizes & Color Asics Gel Sonoma 2 Running Sneakers Womens Trail Shoes Flat Heel B-604 Asics Women's GEL-Sonoma 2 Trail Running Shoes 9

Table 5.7: Compare BM25 with SimCSE (random seed 42 with training dataset eBay titles 100k) on eProduct retrieval results. Given a query, retrieve top 3 relevant documents using BM25 and SimCSE methods. In this case, BM25 outperforms SimCSE.

5.3.2 Comparison between Contrastive Learning Methods and BM25

From Table 5.4 we find BM25 [Yang & Fang+ 17] method outperform all contrastive learning based methods on eProduct [Yuan & Chiang+ 21] evaluation. Also, many previous papers and studies prove that BM25 is a very strong baseline in information retrieval tasks [Rosa & Rodrigues+ 21]. This experiment aims to do a comparison between BM25 and the contrastive learning methods on the retrieval results of eProduct and investigate their differences.

We choose the eProduct retrieval results of SimCSE with seed 42 from Table 5.5 to do the comparison with BM25, since SimCSE with random seed 42 has the best performance on eProduct task. Table 5.6 gives an example of eProduct retrieval results by using BM25 and SimCSE methods. In this case, we find SimCSE outperforms BM25, because SimCSE focuses on "Elephant Rattle" which is the most important term of the query, while BM25 focuses on other less relevant keywords. Thus, SimCSE retrieves more relevant documents than BM25 when given an identical query in this case. It is typically because the dense retrieval model can get semantic

Model	eProduct[%]	
	R_{cap}@10	P@10
SimCSE	69.6	62.5
BM25	73.1	65.8
Rescoring $\alpha = 20$	74.5	67.0

Table 5.8: eProduct evaluation results for BM25, SimCSE and their combination with $\alpha = 20$.

information and modeling of term importance. However, in another Table 5.7 we find BM25 outperforms SimCSE, since BM25 can retrieve a relevant document with "Gel Sonoma 3" which is exactly same part of query, whereas SimCSE can not retrieve relevant documents with product number "3". It is typical because the dense retrieval model confuses product numbers or product specifications. Therefore, we can make a hypothesis that BM25 and SimCSE could be complementary on the eProduct retrieval task in some cases.

To test this hypothesis, we do a combination with SimCSE(select random seed 42 with training dataset eBay titles 100k from Table 5.5) and BM25. We rescore BM25 top1000 retrieval results with SimCSE results according to:

$$\text{Score}(q, d) = \text{Score}_{bm25}(q, d) + \alpha \cdot \text{Score}_{simcse}(q, d) \tag{5.1}$$

where q is query and d is document. $\alpha \in \{0, 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120\}$, $\alpha = 0$ means BM25 only.

We evaluate the combination between BM25 and SimCSE on eProduct task using different α. Figure 5.2 shows the the evaluation results Capped Recall@10 and Precision@10 when we do combination between BM25 and SimCSE by rescoring the retrieval results with different $\alpha \in \{0, 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120\}$, where we observe $\alpha = 20$ has the best performance on eProduct task. It is shown in Table 5.8.

Thus, we can conclude although BM25 has better performance on the eProduct task than SimCSE, BM25 and SimCSE methods are complementary in some cases. One explanation for this is that the BM25 method focuses on word-level overlap and frequency which helps to identify product numbers or product specifications efficiently. Whereas the SimCSE method can identify semantic information like synonyms or titles has a similar meaning, which helps to retrieve more semantic relevant documents when given a query. So their combination can learn not only words level information (like frequency, overlap) but also semantic-level information (like semantic similarity), which can achieve better performance on eProduct task than either BM25 or SimCSE.

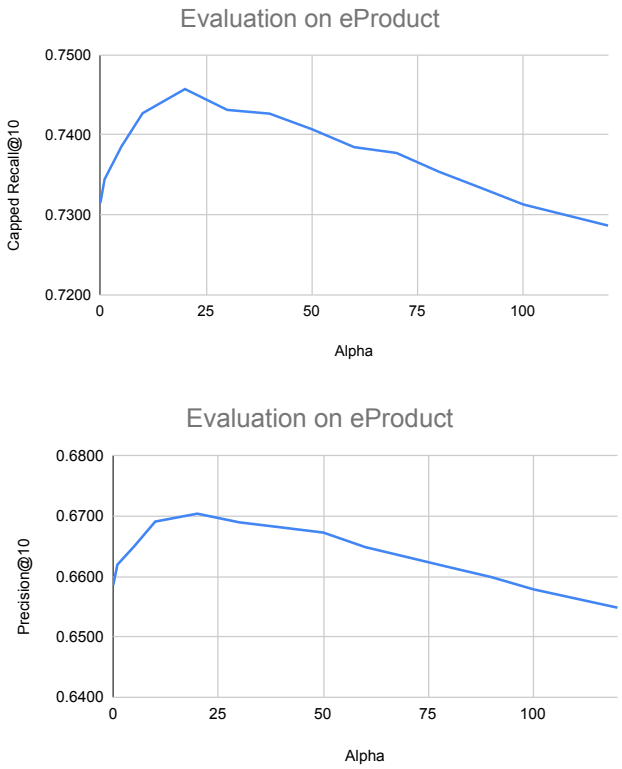


Figure 5.2: The combination of BM25 and SimCSE(random seed 42 with training dataset eBay titles 100k) evaluate on eProduct using different α.

5.3.3 Comparison between BCE and NT-Xent Loss for Contrastive Learning

From Section 2.4 we know that both Binary Cross-Entropy (BCE) loss and NT-Xent loss [Chen & Kornblith$^+$ 20] can be used as contrastive learning loss functions. This experiment aims to investigate how these two different loss behaves on the downstream tasks.

To do a fair comparison, we use contrastive learning method SimCSE [Gao & Yao$^+$ 21] as the training framework and replace its loss function by BCE and NT-Xent loss respectively. We use Bert-base-uncased [Devlin & Chang$^+$ 19] as base model and open domain dataset OpenWebtext 1M as training dataset. Also, we use validation loss as best model checkpoint selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3. We evaluate the trained model on STS tasks [Conneau & Kiela 18] and Duplicate Question Retrieval (Quora

Model	STS-B[%]	STS-Avg[%]
SimCSE$_{NT-Xent}$	71.4	71.9
SimCSE$_{BCE}$	43.3	44.2
Bert-base-uncased	47.3	51.6

Table 5.9: Comparison between BCE (Binary Cross Entropy) loss and NT-Xent loss in STS tasks evaluation. SimCSE$_{NT-Xent}$ indicates using NT-Xent loss in SimCSE framework and SimCSE$_{BCE}$ indicates using BCE loss in SimCSE framework. We report the scores averaged over 3 random seeds.

Model	R@100[%]	
	Quora	CQADupStack
SimCSE$_{NT-Xent}$	96.2	38.2
SimCSE$_{BCE}$	59.8	8.7
Bert-base-uncased	86.1	19.0
BM25	97.3	60.6

Table 5.10: Comparison between BCE and NT-Xent loss in duplicate question retrieval evaluation. We report Recall@100 in Quora as well as CQADupStack averaged over 3 random seeds.

and CQADupStack) [Thakur & Reimers$^+$ 21]. Besides, we also include Bert-base-uncased [Devlin & Chang$^+$ 19] and BM25 [Yang & Fang$^+$ 17] as baselines. The results are shown in Table 5.9 and Table 5.10.

We observe that the evaluation results of NT-Xent loss always outperform the evaluation results of BCE loss in Table 5.9 and Table 5.10. SimCSE with NT-Xent loss can achieve a huge improvement in both STS tasks and duplicate question retrieval tasks compared to baseline model Bert-base-uncased. However, the result of SimCSE with BCE loss is even worse than the baseline model Bert-base-uncased in these two evaluation tasks. We make a tentative conclusion that the proposed Binary Cross Entropy may not be directly applied to contrastive learning. How to use BCE loss in contrastive learning requires further exploration.

Furthermore, although we use the open domain dataset OpenWebtext 1M as a training dataset, which is not related to our downstream tasks, the trained model SimCSE with NT-Xent loss can achieve decent results in our downstream evaluation tasks. The investigation of the dataset domain for contrastive learning will be in Section 5.3.6.

We evaluate duplicate question retrieval with Recall@k and Precision@k ($k \in \{1, 3, 5, 10, 100, 1000\}$), and we wonder what the difference is when using different k.

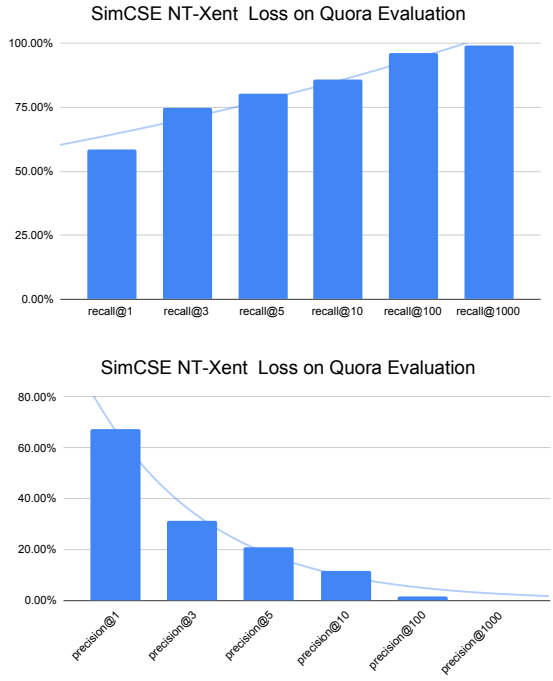


Figure 5.3: The evaluation results on Quora using SimCSE$_{NT-Xent}$ with different Recall@k and Precision@k ($k \in \{1, 3, 5, 10, 100, 1000\}$). Results are averaged over 3 random seeds.

Figure 5.3 and Figure 5.4 illustrate how results behave on Quora and CQADupStack for different k. We observe when k increases, Recall@k goes up but Precision@k goes down. This is because larger k can provide more retrieved documents as well as true matches when given a query. However, since k is larger than before but the number of ground truth matches for a query in the whole candidate documents is limited, according to Equation 4.2 the Precision@k will decrease.

5.3.4 Influence of Batch Size for Contrastive Learning

Some previous papers on contrastive learning reported that a large batch size benefits the final performance on downstream tasks and accelerates the convergence of the model since it provides more in-batch negatives for contrastive learning [Chen & Kornblith$^+$ 20]. More specifically, When we increase the batch size the number of negatives for each sample will increase during training, and vice versa. In this experiment, we also investigate the influence of the batch size for contrastive learning on our downstream tasks.

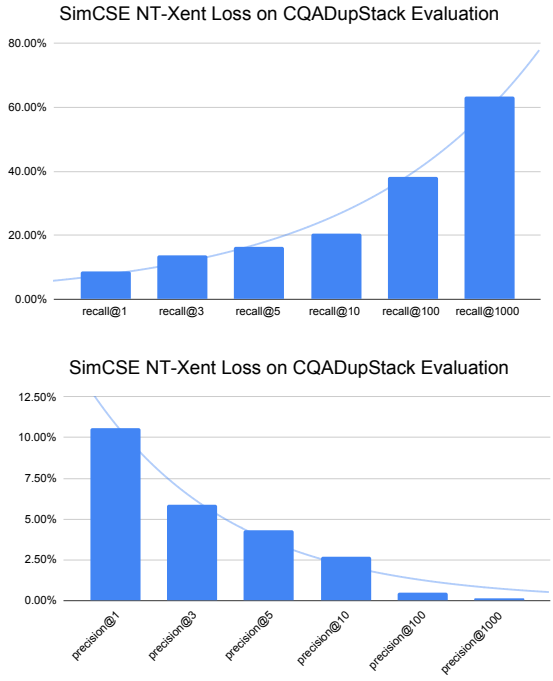


Figure 5.4: The evaluation results on CQADupStack using SimCSE$_{NT-Xent}$ with different Recall@k and Precision@k ($k \in \{1, 3, 5, 10, 100, 1000\}$). Results are averaged over 3 random seeds.

We use SimCSE [Gao & Yao$^+$ 21] as the training framework, the pre-trained eBert model as the base model, and eBay titles 100k as the training dataset. Also, we use validation loss as the best model checkpoint selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3. Besides, we try different batch size $\in \{32, 64, 96, 128, 256, 512\}$ during training. Then we evaluate the trained model on eProduct [Yuan & Chiang$^+$ 21] task. The results are shown in Table 5.5.

From Figure 5.5 We observe when increasing the batch size, the performance of the eProduct task is also improved. Larger batch size does achieve better performance on the downstream task, though the improvement from batch size 64 to 512 is not significant. Meanwhile, a larger batch size does speed up the training process because of reducing the training steps. Table 5.11 shows the exact results of batch size and training steps. In conclusion, it's worth using a larger batch size for contrastive learning during training in this case.

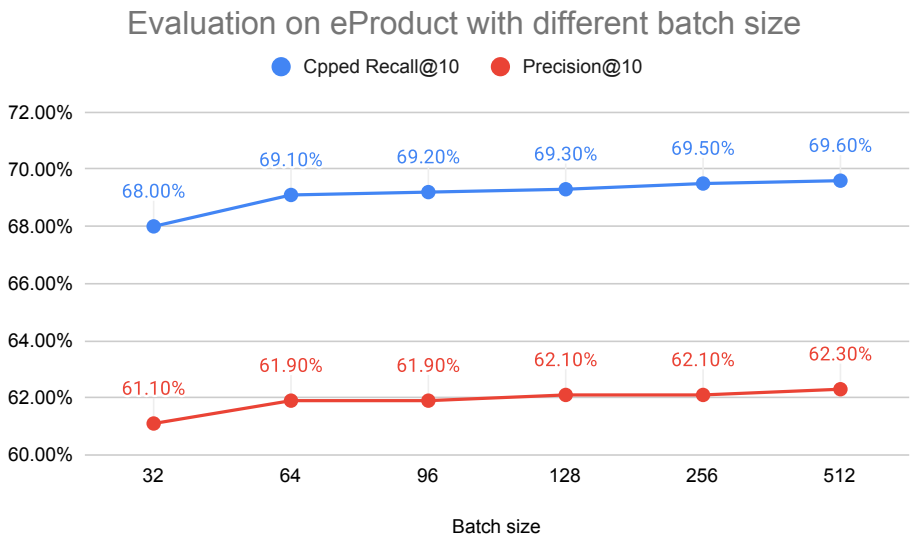


Figure 5.5: Evaluate SimCSE with different batch sizes on eProduct task. Results are averaged over 3 random seeds.

Batch Size	32	64	96	128	256	512
Number of Steps	3125	1562	1041	781	390	195
Capped Recall@10 [%]	68.0	69.1	69.2	69.3	69.5	69.6
Precision@10 [%]	61.1	61.9	61.9	62.1	62.1	62.3

Table 5.11: SimCSE uses eBay titles 100k as training dataset and train one epoch with different batch sizes. Then evaluate the trained model on eProduct task to get Capped Recall@10 and Precision@10. Results are averaged over 3 random seeds.

5.3.5 Influence of Training Corpus Size for Contrastive Learning

In many real-world scenarios, getting a sufficiently high number of unlabeled sentences can be challenging. Hence, deriving good sentence representations even with very small unlabeled training data is significant. To validate the reliability and the robustness of the contrastive learning method under the data scarcity scenarios, we conduct experiments with different training corpus sizes.

We use SimCSE [Gao & Yao+ 21] as the training framework and the pre-trained eBert model as the base model. We use different sizes of eBay titles 1M/100k/50k/20k/10k as training datasets. Also, we use validation loss as the best model checkpoint se-

Training Dataset (eBay titles)	1M	100k	50k	20k	10k
Training Epoch	1	1	1	1	1
Number of Steps	15625	1562	781	312	156
Capped Recall@10 [%]	65.3	69.1	68.8	68.2	66.4
Precision@10 [%]	58.6	61.9	61.7	60.9	59.4

Table 5.12: SimCSE uses eBay titles 1M/100k/50k/20k/10k as training dataset and train one epoch. Then evaluate the trained model on eProduct task to get Capped Recall@10 and Precision@10. Results are averaged over 3 random seeds.

lection. Other hyperparameters setting is same as basic experimental setup from Table 5.3. Then we evaluate the trained model on the eProduct [Yuan & Chiang+ 21] task.

Table 5.12 shows the eProduct evaluation results with different size of corpus. We observe that training with eBay titles 100k has the best performance on the eProduct task. Besides, when increasing the training corpus size from 10k to 100k, the performance increases as well. However, training with eBay titles 1M even harms the performance. To investigate why the largest training dataset eBay titles 1M is the worst, we record the train/validation loss as well as eProduct evaluation results during training in Figure 5.6. We find that the train and validation loss always decreases during training, whereas the Capped Recall@10 and Precision@10 of eProduct evaluation go up rapidly at first, after about 1000 training steps they slightly go down and fluctuate. The reason would be increasing training steps may lead to overfitting of the trained model, since the train and validation loss values remain low, but the performance of eProduct task goes down. Furthermore, smaller datasets like eBay titles 50k/20k can also achieve decent results on downstream tasks on eProduct in Table 5.12, which indicates contrastive learning methods can be applied for few-shot learning. We can make a hypothesis that the contrastive learning method SimCSE can achieve very good performance on downstream tasks after a certain number of training steps and then the performance will go down and fluctuate because of overfitting.

To explore more about the relationship between training steps and performance of the downstream tasks, we train the SimCSE model with 3 epochs with training dataset eBay titles 1M/100k/50k/20k/10k. The results are shown in Table 5.13. We observe train 3 epochs with eBay titles 20k has the best performance on the eProduct task. This supports our hypothesis that after a certain number of training steps SimCSE can achieve very good performance on downstream tasks and then the performance will go down and fluctuate. In this case, the certain number of

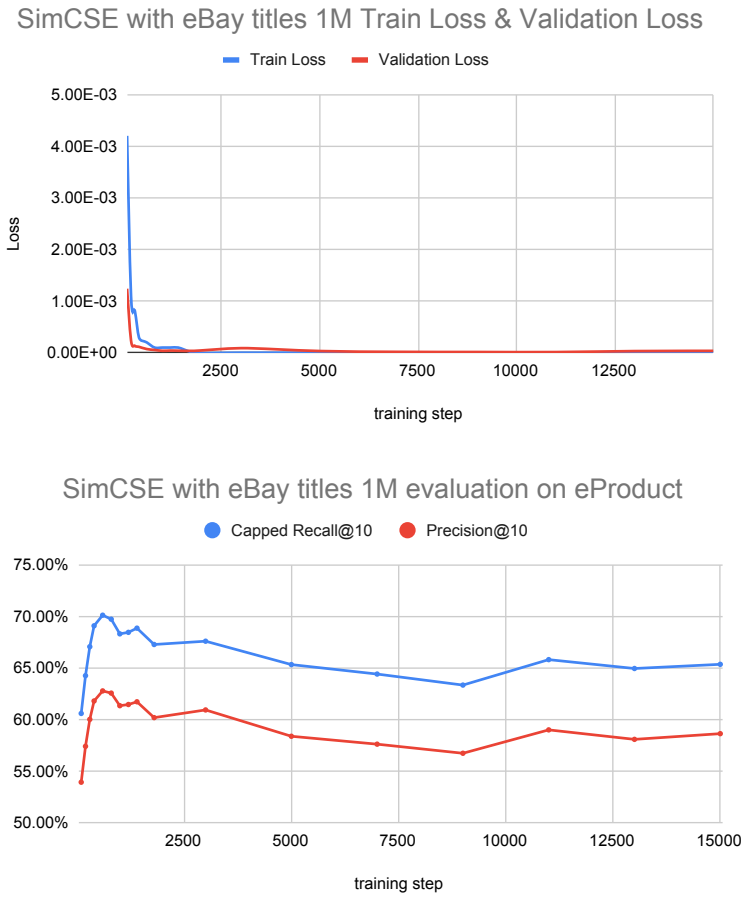


Figure 5.6: SimCSE uses eBay titles 1M as training dataset and train one epoch. Record the train/validation loss and evaluation results of eProduct during training.

training steps is about ∼1000, which is close to the warmup steps 910 in Table 5.3. Therefore, we can conclude when feeding about ∼64,000 unlabeled sentences (training step 1000 × batch size 64), the contrastive learning method SimCSE can produce very good sentence embeddings which can significantly improve the performance on downstream tasks.

Training Dataset (eBay titles)	1M	100k	50k	20k	10k
Training Epoch	3	3	3	3	3
Number of Steps	46875	4686	2343	936	468
Capped Recall@10 [%]	61.6	67.9	68.5	69.4	67.9
Precision@10 [%]	55.3	60.9	61.4	62.1	60.5

Table 5.13: SimCSE uses eBay titles 1M/100k/50k/20k/10k as training dataset and train 3 epochs. Then evaluate the trained model on eProduct task to get Capped Recall@10 and Precision@10. Results are averaged over 3 random seeds.

5.3.6 Different Domain Effects for Contrastive Learning and Autoencoder

In previous experiments, we always use the pre-trained model Bert-based-uncased [Devlin & Chang$^+$ 19] or eBert as the base model then fine-tune the model with domain-specific datasets (e.g. if do eProduct [Yuan & Chiang$^+$ 21] task, we fine-tune eBert model with in-domain datasets eBay titles or eProduct titles) using unsupervised methods. In this experiment, we aim to investigate the domain effects of training datasets for contrastive learning and autoencoder. More specifically, we fine-tune the pre-trained model using different domain training datasets and evaluate the trained model on the downstream tasks.

We use SimCSE [Gao & Yao$^+$ 21] as a training framework for contrastive learning and CLM Autoencoder as well as MLM Autoencoder to do a comparison. We set the pre-trained eBert model as the base model, and use different domain datasets eBay titles 100k, OpenWebText 100k and Quora titles 100k as training datasets to fine-tune the base model. We train the model with 3 epochs in order to let the model learn enough information from the training dataset. Also, we use validation loss as the best model checkpoint selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3. Then we evaluate the trained model on the eProduct [Yuan & Chiang$^+$ 21] task.

Table 5.14 shows eProduct evaluation results with different domains of training corpus for contrastive learning and autoencoder-based methods. We observe that both contrastive learning method SimCSE and autoencoder-based methods can improve the performance on eProduct evaluation compared to the baseline pre-trained model eBert. Besides, autoencoder-based methods have better performance on in-domain dataset eBay titles 100k which is intuitive that autoencoder-based methods can learn much word information from in-domain datasets [Wang & Reimers$^+$ 21] since its objective is to reconstruct sentence word by word, whereas

Model	Training Dataset	eProduct[%]	
		R_{cap}@10	P@10
SimCSE	eBay titles 100k	67.9	60.9
	OpenWebText 100k	71.2	63.7
	Quora titles 100k	70.5	63.0
CLM autoencoder	eBay titles 100k	68.3	61.4
	OpenWebText 100k	66.5	59.9
	Quora titles 100k	66.1	59.7
MLM autoencoder	eBay titles 100k	68.9	61.9
	OpenWebText 100k	67.0	60.3
	Quora titles 100k	66.4	59.8
eBert	-	50.9	45.9
BM25	-	73.1	65.8
Word2vec	-	48.4	43.7

Table 5.14: SimCSE and autoencoder-based methods use different domain training datasets and evaluate trained model on eProduct task. The base model used is eBert which is trained with 3 epochs. Results are averaged over 3 random seeds.

contrastive learning method has better performance on out-domain datasets OpenWebText [Gokaslan & Cohen] and Quora titles [Thakur & Reimers$^+$ 21]. We can make a hypothesis that contrastive learning methods are not sensitive to the domain of training datasets, since it focuses on the sentence similarity information which is used to tune the parameters of the pre-trained model to make the model learn similarity information of sentences [Yan & Li$^+$ 21] despite the domains of training corpus during training. Another reason could be the base model eBert has already been pre-trained on eBay titles 1B, so fine-tuning eBert with eBay titles 100k will lead to overfitting on the training corpus. However, fine-tuning eBert with out-domain datasets OpenWebText 100k and Quora titles 100k can add some noise to the eBert model to make the trained model more robust.

To investigate if the pre-trained model eBert has a large influence on the performance of downstream tasks when fine-tuning with different domain datasets, we also use Bert-base-uncased [Devlin & Chang$^+$ 19] as the base model to train with different domain datasets. The results are shown in Table 5.15.

From Table 5.15 we see SimCSE method with Bert-base-uncased [Devlin & Chang$^+$ 19] has very close results when training with different domain datasets, which can support our hypothesis that contrastive learning methods are not sensitive to the domain of training datasets. However, autoencoder-based methods still have better performance when training with in-domain dataset eBay titles 100k.

Model	Training Dataset	eProduct[%]	
		R_{cap}@10	P@10
SimCSE	eBay titles 100k	65.7	59.2
	OpenWebText 100k	64.0	57.5
	Quora titles 100k	65.3	48.7
CLM autoencoder	eBay titles 100k	66.9	60.0
	OpenWebText 100k	62.4	56.2
	Quora titles 100k	63.2	56.9
MLM autoencoder	eBay titles 100k	67.5	60.5
	OpenWebText 100k	63.7	57.1
	Quora titles 100k	63.3	56.8
Bert-base-uncased	-	55.1	49.7
BM25	-	73.1	65.8
Word2vec	-	48.4	43.7

Table 5.15: SimCSE and autoencoder-based methods use different domain training datasets and evaluate trained model on eProduct task. The base model used is Bert-base-uncased which is trained with 3 epochs. Results are averaged over 3 random seeds.

Furthermore, the pre-trained model we choose is a key factor for downstream tasks, since fine-tuning the eBert model outperforms fine-tuning the Bert-base-uncased model for both contrastive learning methods and autoencoder-based methods.

5.3.7 Multi-task Learning for Contrastive Learning and Autoencoder

Section 3.3 proposed a multi-task learning approach to combine contrastive learning-based methods and autoencoder-based methods. This section aims to investigate whether this multi-task learning approach can help to improve the quality of sentence embeddings in order to achieve better performance on downstream tasks.

We use SimCSE [Gao & Yao$^+$ 21] as training framework for contrastive learning. Then we use CLM autoencoder and MLM autoencoder to combine with contrastive learning according to the Equation 3.6 with different weights $w \in \{0.1, 0.01, 0.001, 0.0001\}$ respectively, which is the multi-task learning approach in this thesis.

First, we do an evaluation on eBay internal task eProduct [Yuan & Chiang$^+$ 21]. We use a pre-trained eBert model as the base model and an in-domain dataset eBay titles 100k as the training dataset. We train the model with 3 epochs using a multi-task learning approach in order to make the trained model learn more from the training dataset. Also, we use validation loss as the best model checkpoint

Loss	eProduct[%]	
	R_{cap}@10	P@10
$\mathcal{L}^1_{simcse}$	69.1	61.9
$\mathcal{L}_{simcse}$	67.9	60.9
$\mathcal{L}_{CLM\_autoencoder}$	68.3	61.4
$\mathcal{L}_{MLM\_autoencoder}$	68.9	61.9
$\mathcal{L}_{simcse} + 0.1 * \mathcal{L}_{CLM\_autoencoder}$	68.8	61.7
$\mathcal{L}_{simcse} + 0.01 * \mathcal{L}_{CLM\_autoencoder}$	68.9	61.8
$\mathcal{L}_{simcse} + 0.001 * \mathcal{L}_{CLM\_autoencoder}$	69.7	62.6
$\mathcal{L}_{simcse} + 0.0001 * \mathcal{L}_{CLM\_autoencoder}$	68.6	61.6
$\mathcal{L}_{simcse} + 0.1 * \mathcal{L}_{MLM\_autoencoder}$	67.9	60.9
$\mathcal{L}_{simcse} + 0.01 * \mathcal{L}_{MLM\_autoencoder}$	68.2	61.3
$\mathcal{L}_{simcse} + 0.001 * \mathcal{L}_{MLM\_autoencoder}$	68.6	61.6
$\mathcal{L}_{simcse} + 0.0001 * \mathcal{L}_{MLM\_autoencoder}$	67.8	60.8
eBert	50.9	45.9
BM25	73.1	65.8

Table 5.16: The evaluation results on eProduct. $\mathcal{L}^1_{simcse}$ indicates using SimCSE and train with 1 epoch, which is used as a comparison here, since SimCSE training with 1 epoch can achieve better results than 3 epochs from Section 5.3.5. We used multi-task loss with different weight $w \in \{0.1, 0.01, 0.001, 0.0001\}$. Results are averaged over 3 random seeds.

selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3. Then we evaluate the trained model on the eProduct task, and evaluation results, as well as comparison results, are shown in Table 5.16.

From Table 5.16 we observe that the multi-task combination of SimCSE and CLM autoencoder can achieve better results than both SimCSE and CLM autoencoder on the eProduct task with 3 training epochs, especially when the weight $w = 0.001$. Besides, the multi-task combination of SimCSE and MLM autoencoder performs not well as the combination of SimCSE and CLM autoencoder, but the performance is still better than SimCSE when $w = 0.01$ or 0.001.

We also do an evaluation on duplicate retrieval task Quora [Thakur & Reimers$^+$ 21]. In this experiment, we use the pre-trained model Bert-base-uncased [Devlin & Chang$^+$ 19] as the base model and in-domain dataset Quora titles 100k as the training dataset. Besides, We train the base model with 3 epochs using multi-task loss in order to make the trained model learn more from the training dataset. We use Recall@10 of Quora devset (validation task) as the best model checkpoint selection. Other hyperparameters setting is same as basic experimental setup from Table 5.3.

Loss	R@100[%]	R@10[%]
$\mathcal{L}^1_{simcse}$	96.5	86.8
$\mathcal{L}_{simcse}$	97.2	88.3
$\mathcal{L}_{CLM\_autoencoder}$	95.9	84.9
$\mathcal{L}_{MLM\_autoencoder}$	91.2	77.5
$\mathcal{L}_{simcse} + 0.1 * \mathcal{L}_{CLM\_autoencoder}$	97.1	87.3
$\mathcal{L}_{simcse} + 0.01 * \mathcal{L}_{CLM\_autoencoder}$	97.4	88.4
$\mathcal{L}_{simcse} + 0.001 * \mathcal{L}_{CLM\_autoencoder}$	97.4	88.8
$\mathcal{L}_{simcse} + 0.0001 * \mathcal{L}_{CLM\_autoencoder}$	97.3	88.6
$\mathcal{L}_{simcse} + 0.1 * \mathcal{L}_{MLM\_autoencoder}$	93.5	81.4
$\mathcal{L}_{simcse} + 0.01 * \mathcal{L}_{MLM\_autoencoder}$	95.7	85.0
$\mathcal{L}_{simcse} + 0.001 * \mathcal{L}_{MLM\_autoencoder}$	97.0	87.7
$\mathcal{L}_{simcse} + 0.0001 * \mathcal{L}_{MLM\_autoencoder}$	97.3	88.3
Bert-base-uncased	86.1	71.7
BM25	97.3	88.9

Table 5.17: The evaluation results on Quora task. We only report Recall@100 and Recall@10 in this Table. $\mathcal{L}^1_{simcse}$ indicates using SimCSE and train with 1 epoch, which is used as a comparison here. We used multi-task loss with different weight $w \in \{0.1, 0.01, 0.001, 0.0001\}$. Results are averaged over 3 random seeds.

Then we evaluate the trained model on the Quora task, and the evaluation results, as well as comparison results, are shown in Table 5.17.

From Table 5.17 we find that the multi-task combination for SimCSE and CLM autoencoder can achieve comparable results when $w = 0.01, 0.001, 0.0001$. It even outperform the strong baseline BM25 on Recall@100 when $w = 0.01$ or 0.001. Moreover, the combination of SimCSE and MLM autoencoder can also achieve better results Recall@100 on Quora task than both SimCSE and MLM autoencoder when $w = 0.0001$ with 3 training epochs.

Altogether, we can conclude that the multi-task learning approach of SimCSE and CLM autoencoder does help to improve the performance on downstream tasks using appropriate weight w. Furthermore, the multi-task learning approach of SimCSE and MLM autoencoder also helps with appropriate weight w in some cases. Generally, the multi-task learning approach in Section 3.3 is worth a try when considering improving the quality of sentence embeddings in an unsupervised way.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, we focus on improving the quality of sentence embeddings using unsupervised contrastive learning methods. In order to evaluate the quality of sentence embeddings, we utilize several downstream tasks which are retrieval tasks (eProduct [Yuan & Chiang$^+$ 21] challenge in eBay, duplicate question retrieval tasks (Quora and CQADupStack) [Thakur & Reimers$^+$ 21]) and semantic textual similarity (STS) tasks [Conneau & Kiela 18]. For each retrieval task, we have queries and documents, aiming to retrieve the top k relevant documents based on sentence embeddings when given a query. For the semantic textual similarity tasks, we aim to calculate Spearman's correlation between similarity score based on sentence embedding and the human-annotated score of each sentence pair.

We consider contrastive learning methods, autoencoder-based methods, and their combination as a multi-task learning method, which can be used to improve sentence embedding in an unsupervised way. The goal of contrastive learning is to make similar sentences close and dissimilar sentences far apart in a vector space. We use ConSERT [Yan & Li$^+$ 21] and SimCSE [Gao & Yao$^+$ 21] as two main state-of-the-art contrastive learning methods in this thesis. The goal of the autoencoder is to reconstruct the input sentence based on the input sentence embedding. In this thesis, we use MLM autoencoder and CLM autoencoder as two main autoencoder-based methods. We proposed a multi-task learning approach that combines contrastive learning and autoencoder-based methods since we want to investigate if we can take advantage of both methods.

Moreover, we have several baseline methods: the pre-trained model (Bert-base-uncased [Devlin & Chang$^+$ 19], eBert) maps sentence to embedding directly, word2vec [Mikolov & Chen$^+$ 13] averaging all words embedding from the same sentence to obtain sentence embedding, BM25 (a count-based method) [Robertson & Zaragoza 09] calculates query-document similarity score in retrieval tasks directly. These baselines methods are the comparison for contrastive learning and autoencoder-based methods when doing an evaluation.

To investigate how contrastive learning works on downstream tasks, we evaluate the state-of-the-art contrastive learning methods SimCSE [Gao & Yao$^+$ 21] and ConSERT [Yan & Li$^+$ 21] on the eProduct task and find that both methods can achieve better results than our baseline pre-trained model, which indicates contrastive learning methods can boost the quality of sentence embeddings. Also, we report the evaluation results of STS tasks, from which we observe the results of STS tasks are not correlated to eProduct task results. It shows that STS tasks are insufficient evaluation tasks if we want to utilize sentence embedding on downstream tasks.

We also find that baseline method BM25 outperforms both ConSERT and SimCSE results on the eProduct task. We analyze the retrieval results of eProduct from the SimCSE method and BM25 respectively. In some cases, we observe that BM25 outperforms SimCSE. And in some other cases, SimCSE outperforms BM25. Thus, we make a hypothesis that BM25 can catch product numbers or product specifications efficiently and the SimCSE method can get semantic information and modeling of term importance. Furthermore, we combine BM25 and SimCSE by rescoring BM25 top1000 retrieval results with SimCSE results on the eProduct task, from which we observe this combination achieves better performance than both BM25 and SimCSE. Therefore, we also conclude that in some cases BM25 and SimCSE are complementary.

Moreover, we investigate if binary cross-entropy loss can also work in contrastive learning. We use the SimCSE method and replace its NT-Xent loss with binary cross-entropy loss. On the evaluation of duplicate question retrieval, we observe that SimCSE with binary cross-entropy loss has worse performance than SimCSE with NT-Xent loss as well as the baseline pre-trained eBert model. We can conclude that the proposed binary cross-entropy loss may not be directly applied to contrastive learning.

To explore which factors can influence the performance of contrastive learning methods, we conduct several experiments. First, to observe how batch size affects the performance of contrastive learning, we use the SimCSE method with different batch sizes and report the results of the eProduct evaluation task. We find that when increasing the batch size the performance also goes up. However, when the batch size is large enough, it will meet a bottleneck of performance. Also, we find a large batch size can speed up the training process. Then we wonder if contrastive learning can also fit into a smaller or larger size of training datasets, we use SimCSE methods with different sizes of training datasets. We report the evaluation results on the eProduct task, where we find that our contrastive learning methods can perform well on a small size training corpus, but perform badly on a large size training corpus. Besides, after a certain number of training steps, we observe contrastive learning can achieve a decent result on the eProduct task. More specifically, we conclude that contrastive learning requires about 64,000 unlabeled sentences to achieve good performance on the eProduct task. Additionally, we investigate the

domain effect of the training corpus for contrastive learning and autoencoder-based methods. We report the eProduct evaluation results of SimCSE, CLM autoencoder, and MLM autoencoder using three different domains of the training corpus. We observe that even the out-domain training datasets have better performance than in-domain training datasets on the eProduct task using the SimCSE method.

Finally, to see how the multi-task learning approach (combination of contrastive learning methods and autoencoder-based methods) works on the downstream tasks. We compare "single-task" (i.e. only contrastive learning or only autoencoder) with multi-task learning. We report the evaluation results of both the eProduct task and duplicate question retrieval Quora task, from which we observe multi-task learning outperforms "single-task" in some cases. Thus, we conclude that multi-task learning can help to improve the quality of sentence embeddings in some cases.

6.2 Future Work

In this thesis, we investigate how unsupervised contrastive learning methods and autoencoder-based methods improve the quality of sentence embeddings by evaluating the sentence embeddings on downstream tasks. Also, we analyze several factors of contrastive learning which could affect the quality of sentence embeddings. However, there are still some experimental results required to do detailed analysis and some methods can be improved accordingly.

In Section 5.3.6, we observe that even training with out-domain datasets can achieve better performance on downstream task eProduct [Yuan & Chiang$^+$ 21] than in-domain datasets when using the contrastive learning method SimCSE [Gao & Yao$^+$ 21]. The reason for this phenomenon is still unclear. It requires more analysis and further exploration.

Furthermore, we always use 2 layers of Transformer decoder for autoencoder-based methods in this work. Thus, the exploration for more configurations of the autoencoder-based methods is necessary, which may help to improve the performance on downstream tasks. For MLM autoencoder, we can try to include a cross-attention mechanism into the decoder like what CLM autoencoder does, which may lead to further improvement for the sentence embeddings.

Additionally, for multi-task learning which combines the contrastive learning methods and autoencoder-based methods, one possible attempt is to involve the mask language modeling task in this multi-task. We believe this combination would give us some interesting results.

Another extension related to autoencoder-based method is to use it in the pre-training procedure, which could make the pre-trained language model focus on learning semantically relevant sentence embeddings. We expect that the sentence

embedding extracted from such a pre-trained language model should be more beneficial for the performance on the downstream tasks.

Altogether, if we want to take the quality of sentence embedding a step further, much more research and exploration into this field are badly needed.

List of Figures

List of Tables

Bibliography

[Agirre & Banea$^+$ 14] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, J. Wiebe. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 81–91, Dublin, Ireland, Aug. 2014. Association for Computational Linguistics.

[Agirre & Banea$^+$ 15] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, J. Wiebe. 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)*, pp. 252–263, Denver, Colorado, June 2015. Association for Computational Linguistics.

[Agirre & Banea$^+$ 16] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, J. Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 497–511, San Diego, California, June 2016. Association for Computational Linguistics.

[Agirre & Cer$^+$ 12] E. Agirre, D. Cer, M. Diab, A. 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)*, pp. 385–393, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics.

[Agirre & Cer$^+$ 13] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, W. 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*, pp. 32–43, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[Caruana 04] R. Caruana. Multitask learning. *Machine Learning*, Vol. 28, pp. 41–75, 2004.

[Cer & Diab$^+$ 17] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, L. Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused

evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017.

[Cer & Yang+ 18] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, R. Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 169–174, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[Chen & Kornblith+ 20] T. Chen, S. Kornblith, M. Norouzi, G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[Chopra & Hadsell+ 05] S. Chopra, R. Hadsell, Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, pp. 539–546 vol. 1, 2005.

[Chuang & Dangovski+ 22] Y.-S. Chuang, R. Dangovski, H. Luo, Y. Zhang, S. Chang, M. Soljačić, S.-W. Li, W.-t. Yih, Y. Kim, J. Glass. Diffcse: Difference-based contrastive learning for sentence embeddings, 2022.

[Conneau & Kiela+ 17] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Conneau & Kiela 18] A. Conneau, D. Kiela. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[Devlin & Chang+ 19] J. Devlin, M. Chang, K. Lee, K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

[Fournier & Aloise 19] Q. Fournier, D. Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE*

Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). IEEE, jun 2019.

[Gao & Yao+ 21] T. Gao, X. Yao, D. Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[Gokaslan & Cohen] A. Gokaslan, V. Cohen. Openwebtext corpus.

[Hadsell & Chopra+ 06] R. Hadsell, S. Chopra, Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2, pp. 1735–1742, 2006.

[Hill & Cho+ 16] F. Hill, K. Cho, A. Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics.

[Hinton & Krizhevsky+ 11] G. E. Hinton, A. Krizhevsky, S. D. Wang. Transforming auto-encoders. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, 44–51, Berlin, Heidelberg, 2011. Springer-Verlag.

[Hoogeveen & Verspoor+ 15] D. Hoogeveen, K. M. Verspoor, T. Baldwin. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium (ADCS)*, ADCS '15, pp. 3:1–3:8, New York, NY, USA, 2015. ACM.

[Kingma & Welling 14] D. P. Kingma, M. Welling. Auto-encoding variational bayes. *CoRR*, Vol. abs/1312.6114, 2014.

[Kiros & Zhu+ 15] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, S. Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 3294–3302, Cambridge, MA, USA, 2015. MIT Press.

[Kramer 91] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal*, Vol. 37, pp. 233–243, 1991.

[Le & Mikolov 14] Q. Le, T. Mikolov. Distributed representations of sentences and documents. In E. P. Xing, T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32 of *Proceedings of Machine Learning Research*, pp. 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR.

[Li & Ma+ 17] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, Z. Zhu. Deep speaker: an end-to-end neural speaker embedding system. *ArXiv*, Vol. abs/1705.02304, 2017.

[Lin & Ma+ 21] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, R. Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pp. 2356–2362, 2021.

[Liou & Cheng+ 14] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou. Autoencoder for words. *Neurocomput.*, Vol. 139, pp. 84–96, sep 2014.

[Liu & Ott+ 19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, Vol. abs/1907.11692, 2019.

[Logeswaran & Lee 18] L. Logeswaran, H. Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018.

[Mikolov & Chen+ 13] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio, Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[Řehůřek & Sojka 10] R. Řehůřek, P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[Reimers & Gurevych 19] N. Reimers, I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[Robertson & Zaragoza 09] S. Robertson, H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, Vol. 3, No. 4, pp. 333–389, apr 2009.

[Rosa & Rodrigues+ 21] G. M. Rosa, R. C. Rodrigues, R. de Alencar Lotufo, R. Nogueira. Yes, BM25 is a strong baseline for legal case retrieval. *CoRR*, Vol. abs/2105.05686, 2021.

[Ruder 17] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, Vol. abs/1706.05098, 2017.

[Schroff & Kalenichenko$^+$ 15] F. Schroff, D. Kalenichenko, J. Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[Shen & Mueller$^+$ 20] T. Shen, J. Mueller, R. Barzilay, T. Jaakkola. Educating text autoencoders: Latent representation guidance via denoising. In *International Conference on Machine Learning*, pp. 8719–8729. PMLR, 2020.

[Srivastava & Hinton$^+$ 14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 56, pp. 1929–1958, 2014.

[Thakur & Reimers$^+$ 21] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

[Vaswani & Shazeer$^+$ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[Vincent & Larochelle$^+$ 10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, Vol. 11, pp. 3371–3408, 2010.

[Wang & Liu 21] F. Wang, H. Liu. Understanding the behaviour of contrastive loss. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2495–2504, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[Wang & Reimers$^+$ 21] K. Wang, N. Reimers, I. Gurevych. TSDAE: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. In *EMNLP*, 2021.

[Weng 21] L. Weng. Contrastive representation learning. *lilianweng.github.io*, 2021.

[Wolf & Debut$^+$ 20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush. Transformers: State-of-the-art natural language processing.

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, Oct. 2020. Association for Computational Linguistics.

[Yan & Li$^+$ 21] Y. Yan, R. Li, S. Wang, F. Zhang, W. Wu, W. Xu. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5065–5075, Online, Aug. 2021. Association for Computational Linguistics.

[Yang & Fang$^+$ 17] P. Yang, H. Fang, J. Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, 1253–1256, New York, NY, USA, 2017. Association for Computing Machinery.

[Yuan & Chiang$^+$ 21] J. Yuan, A.-T. Chiang, W. Tang, A. Haro. eProduct: A million-scale visual search benchmark to address product recognition challenges. *arXiv preprint arXiv:2107.05856*, 2021.