Transformer-Based Multi-lingual sentence embeddings

Pei WANG

Supervisor
Prof. Martin JAGGI

Contributors
Prakhar GUPTA
Matteo PAGLIARDINI

A thesis submitted in fulfillment of the requirements for the degree of Master in Electrical Engineering with contributions from Machine Learning and Optimization Laboratory School of Computer and Communication Sciences at École Polytechnique Fédérale de Lausanne

École Polytechnique Fédérale de Lausanne, 2020
Department of Electrical Engineering

Abstract

In this thesis, We present a transformers-based multi-lingual embedding model to represent sentences in different languages in a common space. To do so, our system uses the structure of a simplified transformer with a shared byte-pair encoding vocabulary for two languages (English and French) and trained on publicly available parallel corpora. Also, new objective losses have experimented including a cross-lingual loss and a sentence alignment loss for presenting better representation quality. We evaluate our generated sentence representations on the sentence retrieval task from MUSE, multi-lingual zero-shot document classification and natural language inference task from MLDoc and XNLI respectively compared with competitors like Bi-Bert2Vec (Sabet et al., 2020, LASER(Artetxe and Schwenk, 2019 and Multi-lingual BERT(mBERT proposed by Devlin et al., 2018).

Our proposed model obtains state-of-art results on the cross-lingual sentence retrieval task and it outperforms other competitors like Bi-Bert2Vec and LASER on the MLDoc task(Schwenk and Li, 2018) as well. We also experiment with model architectures, objectives and the tensors used to represent sentences and then proposed a new sentence alignment loss which has a positive impact on the quality of sentence representation.

Acknowledgment

I am very grateful for MLO to provide me diverse topics at first, a pleasant working environment, sufficient computing resources and hardware to complete this project.

Especially, I would like to offer my gratitude to the following people for their support during this master thesis:

Prof. Martin Jaggi for supervising this thesis, providing invaluable advice and feedback throughout this project. He consistently allowed this project to be my own work but steered me in the right direction whenever he thought I needed it.

Prakhar Gupta and **Matteo Pagliardini** for being always available to answer my questions, discuss ideas, implementations and offer advice. Without their passionate participation, guidance, and suggestions, this master thesis could not have been successfully conducted.

I take this opportunity to express gratitude to my family, who supports me not only in finance. My parents always encourage me and care about me for my daily life and study. This thesis stands as a testament to your unconditional love and encouragement.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Language models and transfer learning have been paid much attention in recent years. Phenomenal results were achieved by first learning general language representations, and then being integrating into task-specific downstream systems such as sentiment analysis, question answering and others. This approach was first popularized by word embeddings (Mikolov et al., 2013b, Pennington et al., 2014), but has recently been superseded by sentence-level representations (Peters et al., 2018, Devlin et al., 2018).

However, these monolingual models are unable to leverage information across different languages since they were built for a single language or several languages separately and the representation qualities and downstream tasks performances of these models depend on their available training resources. Thus, an increasingly popular direction that enables us to compare the meanings across languages and also enables model transfer between languages is to develop multilingual embeddings, that is, between resource-rich and low-resource languages, to provide a common representation space. This is essential for downstream tasks such as bilingual lexicon induction, machine translation, or cross-lingual information retrieval, or zero-shot learning from resource-rich to resource-poor languages.

Most recently, Artetxe and Schwenk, 2019 used the Seq2Seq architecture composed by BiLSTM encoder to get the embeddings of s sentence and the BiLSTM was trained more than 90 languages at the same time. Devlin et al., 2018, Lample and Conneau, 2019 introduced mBERT and XLM - masked language models pre-trained on multiple languages by large Transformer models (Vaswani et al., 2017) without any cross-lingual supervision. Lample and Conneau, 2019 proposed translation language modeling (TLM) as a way to leverage parallel data and obtain a new state of the are on the cross-lingual natural language inference (XNLI) benchmark (Conneau et al., 2018b).

Later, Conneau et al., 2019 proposed XLM-R which extended XLM by incorporating more training data and languages. In consideration of such a scale of training data, a similar model proposed by Arivazhagan et al., 2019 was trained in 103 languages on over 25 billion parallel sentences. But, massively multilingual machine translation models from large parallel corpora are expensive on collecting data, computation resources and training time.

Contributions With the goal of presenting a light-version cross-lingual embedding model who get general sentence representations, we propose the **Bi-Bert2Sent** algorithm, which extends the Sentence-BERT algorithm (Reimers and Gurevych, 2019) to the cross-lingual setting but uses a fewer layers Transformers (Vaswani et al., 2017). Inspired by Sabet et al., 2020, we employ monolingual and cross-lingual objectives and experiment with a new sentence alignment loss to optimize.

In this work, we are concerned in a universal language-agnostic sentence embeddings, that is, vector representations of sentences that are general concerning two dimensions: the input language and the NLP task. Unlike the large and complex models requiring massive parallel data mentioned above, our supervised model pre-trained on 32 million sentence pairs in 2 languages has a 2-layer, 8-head Transformer as the encoder and leverages parallel data with a new cross-lingual language model objective. Precisely, we make the following contributions:

- We use the siamese Transformers-like network structure as an encoder with a shared vocabulary to encode parallel sentences separately, which enables us to get contextual and aligned sentence embeddings.
- We experiment with several supervised learning objectives and their combinations which benefit cross-lingual pre-training when parallel data is available.
- We experiment with different approaches to representing sentences that improve the performances for downstream tasks.

We evaluate our model **Bi-Bert2Sent** on tasks including sentence translation retrieval(Conneau et al., 2017) with Europarl[1], cross-lingual document classification on (Schwenk and Li, 2018), and natural language inference on XNLI (Conneau et al., 2018b).

1.1 Related Work

A series of follow-up studies indicate that cross-lingual representations can be used to improve the quality of monolingual representations (Faruqui and Dyer, 2014), thus multilingual representations have recently attracted large attention. This variation is mainly caused by the used data, the monolingual and cross-lingual objectives, and how these are optimized.

The initial research direction was multilingual representations of words (Ruder et al., 2017). Mapping-based approaches were the mainstream in that time, which trains monolingual word representations independently on large monolingual corpora and then seek to learn a transformation matrix that maps representations of one language to the representations of the other language (Mikolov et al., 2013a, Zhang et al., 2017, Joulin et al., 2018). Another approach learns word representations jointly by minimizing monolingual losses with the cross-lingual regularization terms like TRANSGRAM(Coulmance et al., 2016) and Cr5(Josifoski et al., 2019). The cross-lingual items encourage words that are often aligned with each other in a parallel corpus to be similar(Klementiev et al., 2012). Both of them require parallel data to construct their objectives so that source and target languages can be aligned in a common space.

Making the most of cross-lingual word-level representations, compositional sentence models use word representations to construct sentence representations of aligned sentences, e.g. the sum of word embeddings within one sentence(Hermann and Blunsom, 2013). Such an idea was later extended by applying the composition and objective function recursively to compose sentences into documents (Hermann and Blunsom, 2014). It was also improved by adding a monolingual objective that operates on the phrase level (Soyer et al., 2015). Although the advantage of this method is that it requires weak or no cross-lingual signals, it has been proven that in actual cross-language conversion settings, the final sentence embedding effect is poor (Conneau et al., 2018a).

[1]Source: https://www.statmt.org/europarl

Levy et al., 2016 used sentence IDs of the aligned sentence pairs that are already a powerful bilingual signal in a parallel corpus. Rajendran et al., 2015 proposed a method that exploits the idea of using pivot languages, which can learn a shared embedding space for two languages without any direct alignment signals because the alignment is implicitly learned via their alignment with the pivot language by optimizing a correlation term. These approaches perform better by adding strong sentence-level supervision, which supports that sentence-level supervision is beneficial.

In order to better express the sentences, more attention was paid on sequence models who wisely combine or encode word-level information to higher levels and the corresponding cross-lingual objectives were designed to leverage more parallel data for adding strong sentence-level supervision. Recent approaches (Artetxe and Schwenk, 2019, Lample and Conneau, 2019) had started to incorporate parallel resources as the additional supervision and context provided by sentence-level alignment may prove to be a valuable resource and complementary to word-level alignment. Early results in this direction indicate that joint training on parallel corpora yields embeddings that are more isomorphic and less sensitive to hubness than mapping-based approaches (Ormazabal et al., 2019) because mapping-based approaches have been shown to rely on the assumption that the embedding spaces in two languages are an approximately isomorphic struggle when mapping between a high resource and a more distant low-resource language (Søgaard et al., 2018).

Johnson et al., 2017 proposed one of the most successful recent approaches of cross-lingual encoders for multilingual machine translation, which outperformed the state of the art on low-resource language pairs and enabled zero-shot translation by using a single shared LSTM encoder and decoder. Artetxe and Schwenk, 2019 showed that the resulting encoder can be used to produce cross-lingual sentence embeddings by training a BiLSTM encoder with a shared vocabulary on parallel data of many languages.

Fueled by work on pre-trained language models (Devlin et al., 2018, Vaswani et al., 2017), mBERT, XLM and XLM-R are introduced. Lample and Conneau, 2019 proposed translation language modeling (TLM) in XLM as a way to leverage parallel data and obtain a new state of the are on the cross-lingual natural language inference (XNLI) benchmark (Conneau et al., 2018b). These models are masked language models pre-trained on multiple languages by large Transformer models (Vaswani et al., 2017) without any cross-lingual supervision.

1.2 Outline

We collect some background information on the Transformers or typical technologies applied in multi-lingual embedding models in chapter 2. In chapter 3, we dive into the detailed explanation of our proposed model including the architecture, the training data and the new objective we add. Later, several experiments about the model structures as well as evaluation tasks are presented in chapter 4. chapter 5 gives some thoughts and ideas on this work.

Chapter 2

Background

In this part, I will briefly present some ideas I will use in my project. It would be helpful to read my proposed model in Chapter 3 if you have basic knowledge of these following concepts.

2.1 Tokenizer

For NLP tasks, raw text data is non-numeric and cannot be processed in models. Usually, the first step is to tokenize text into ids and then map ids to the corresponding vectors. This step converts non-numeric text into numeric data and each token can be represented as vectors in space. There are many tokenizers with pros and cons.

Pre-processing raw text data is a basic and essential step for NLP tasks. Classic word representation cannot handle unseen words or rare words well. Character embedding is one of the solutions to conquer the problem of out-of-vocabulary. Nonetheless, it may too fine-grained to miss some important information. The subword is in the middle of the word and character. Thus, it is appropriate-grained while able to handle unseen words and rare words.

2.1.1 Difficulties

Usually, the tokenization method is simply to split sentences or sequences from the corpus by white space at first. It is unlikely to put all tokens into our vocabulary since the amount of tokens is extensive, which leads to the correspondingly high demand for the memory and the time complexity when training models. The common solution to this problem is to set a threshold. When the appearance of a certain token is higher than this threshold, this token should be added. The rest low-frequency tokens will be encoded as UNKNOWN.

If the training and test set are not in a similar distribution, that is there are considerable low-frequency tokens in the test set, the performance of the model trained by the training set will be influenced. So, the tokenization method by simply splitting text by white space and adding them into vocabulary is not that pleasant.

Some people suggested that the tokenization method is not supposed to be simple and rough. There should be a better way to deal with those low-frequency words. In the field of machine translation, Yang and Kirchhoff, 2006 proposed a backoff model for phrase-based machine translation that translates unseen word forms in the foreign-language text

by hierarchical morphological abstractions at the word and the phrase level. Also, Char-level tokenization tokenizes words into characters, which solve the unseen word problem to some extent as all words consist of characters.

Both of them have pros and cons. The first one could improve the representation of low-frequency words by the backoff table but it all depends on the quality of the backoff table. While although it solves unseen words by splitting them into characters in the second method, the granularity of this approach is too fine.

2.1.2 BPE

NMT models normally take a fixed vocabulary as input, but translation is an open vocabulary problem. Previous work solved the OOV in translation by the back-off dictionary.

In the paper by Sennrich et al., 2015, they introduce a simpler and more effective approach shown in Figure 2.1, realizing the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of sub-word units. It was thought that various word classes can be translated by using smaller units than words. Sennrich et al., 2015 discussed the applicability of various word segmentation techniques, including simple character n-gram models and word segmentation based on byte pair encoding compression algorithms.

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Figure 2.1: The BPE Algorithm[1]

.

This vocabulary may have some combinations that are not words, but this is a

[1]This figure is from the paper by Sennrich et al., 2015

meaningful form, speeding up the learning of NLP and improving the semantic distinction between different words. Because a model might know the relationship between "smart", "smarter", and "smartest" when the model saw "old", "older", and "oldest" in the training process. This does not benefit from the word-level tokenization method.

Meanwhile, OOV cannot be seen if you use such segmentation methods because any word which does not appear in the vocabulary will be broken down into sub-word units.

2.2 Transformers

Moreover, a well-known architecture name **Transformers** proposed by Vaswani et al., 2017 will be used in our proposed model **Bi-Bert2Sent**. Since we aim to get general sentence representations, we use the encoders from Transformers only and define the number of encoders as **the number of layers**.

Neural networks for machine translation typically contain an encoder reading the input sentence and generating a representation of it. A decoder then generates the output sentence word by word while consulting the representation generated by the encoder. The Transformer starts by generating initial representations, or embeddings, for each word. These are represented by unfilled circles. Then, using self-attention, it aggregates information from all of the other words, generating a new representation per word informed by the entire context. This step is then repeated multiple times in parallel for all words, successively generating new representations.

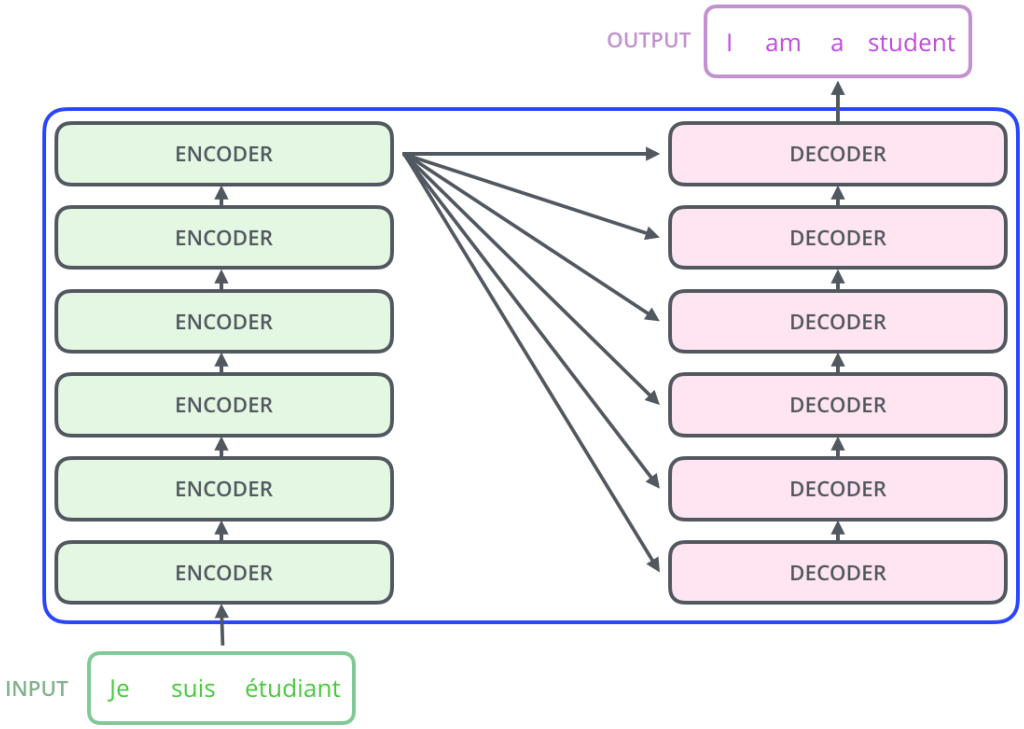

Figure 2.2: Transformers: A High-level Look[2]

.

[2]Alammar, Jay (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer

Shown in Figure 2.2, the Transformer consists of an encoding component, a decoding component, and connections between them. The encoding component is a stack of encoders (6 stacked encoders in Vaswani et al., 2017). The decoding component is a stack of decoders of the same number.

The attention mechanism is a ubiquitous method in modern deep learning models and is applied in Transformers to boost the speed with which these models can be trained. One of the biggest benefits, however, comes from how The Transformer lends itself to parallelization, while one of the drawbacks of RNN architecture (Artetxe and Schwenk, 2019) is serial. Also, it is easy with Attention to detect correlations in one step between items among one sequence no matter how distant they are but the RNN learn correlations through encoding the sequence one by one and updating its state during the whole process.

The encoders are all identical in architecture (yet they do not share weights). Each one can be broken down into two sub-layers: **Self Attention** and **Feed Forward Neural Network**. The encoder's inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word. A feed-forward layer is then following the self-attention layer.

In general NLP applications, the word is normally turned into a trainable vector using an embedding table. The embedding only happens in the bottom-most encoder. The concept which is common to all the encoders is they receive a list of vectors each of the same size - the embedding size. In the first encoder that would be the embeddings of words, while in other encoders, it would be the output of the encoder that's directly below. The size of this list is a hyper-parameter we can set. It would be the length of the longest sentence in the training dataset generally, thus 200 in our situation.

One key property of the Transformer is that the word in each position flows through its path in the encoder. Dependencies between these paths exist in the self-attention layer. However, the feed-forward layer does not have those dependencies, however, and thus the various paths can be executed in parallel while flowing through the feed-forward layer.

As we have mentioned already, an encoder receives a list of vectors as input. It processes this list by proceeding these vectors into a 'self-attention' layer, then into a feed-forward neural network, then sends out the output upwards to the following encoder.

2.2.1 Self-attention mechanism

As the model processes each word (each position within the input sequence), self-attention allows it to seem at other positions within the input sequence for clues which will help lead to a better encoding for this word. Self-attention is that the method the Transformer uses to bake the "understanding" of other relevant words into the one we're currently processing. The self-attention mechanism used is that the scaled dot-product attention according to:

$$Attention(K, Q, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

where d_k is the dimension of the input queries Q, keys K and values V. By using self-attention, transformers can account for the entire sequence in its entirety and bi-directionally. We use dot-product attention because it is much faster and more space-efficient in practice.

The self-attention layer is further refined by adding a mechanism called "multi-headed" attention. This improves the performance of the eye layer in two ways:

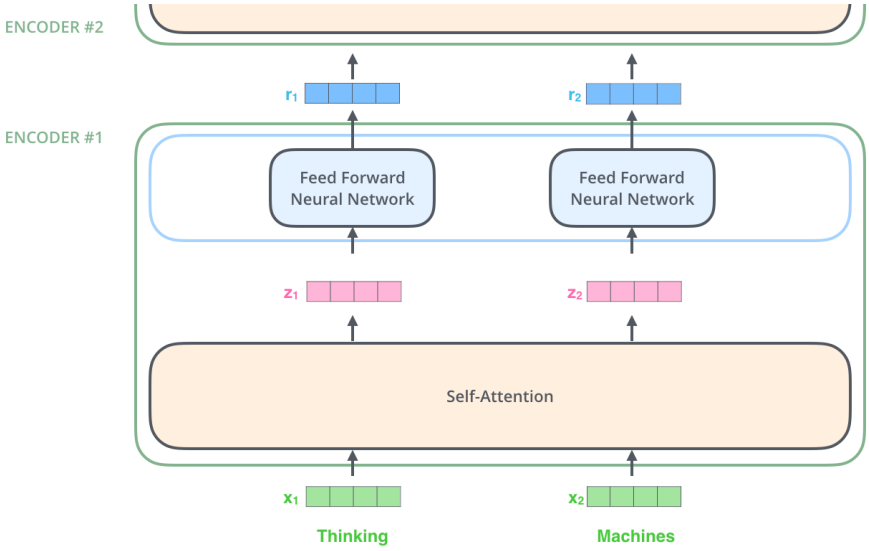

Figure 2.3: Transformers Encoder[4]: The word at each position passes through a self-attention process. Then, they each pass through a feed-forward neural network – the exact same network with each vector flowing through it separately.

- It expands the model's ability to focus on different positions.
- It gives the attention layer multiple "representation subspaces". Since multiple heads have multiple sets of parameters. Each of these sets is randomly initialized. Then, after training, each set can be used to project the input into a special representation subspace.

For multi-head attention with h heads that jointly attend to different representation subspaces at different positions given a sequence of length m and the matrix $H \in R^{m \times d}$, with a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.2}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

where the projections are learned parameter matrices $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ and $W^O \in R^{hd_v \times d_{model}}$

2.2.2 The Residuals and Layer Normalization

One thing needed to mention in the encoder structure is each sub-layer in each encoder has a residual connection around it and then is applied a layer-normalization.

The fully connected feed-forward network is applied to each position separately and identically. This consists of two linear transformations with a ReLu activation ion between.

$$FFN(x) = max(0, xW_1 + b_2)W_2 + b_2 \tag{2.4}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. A residual structure is used wrapped around each of the

[4]Alammar, Jay (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer

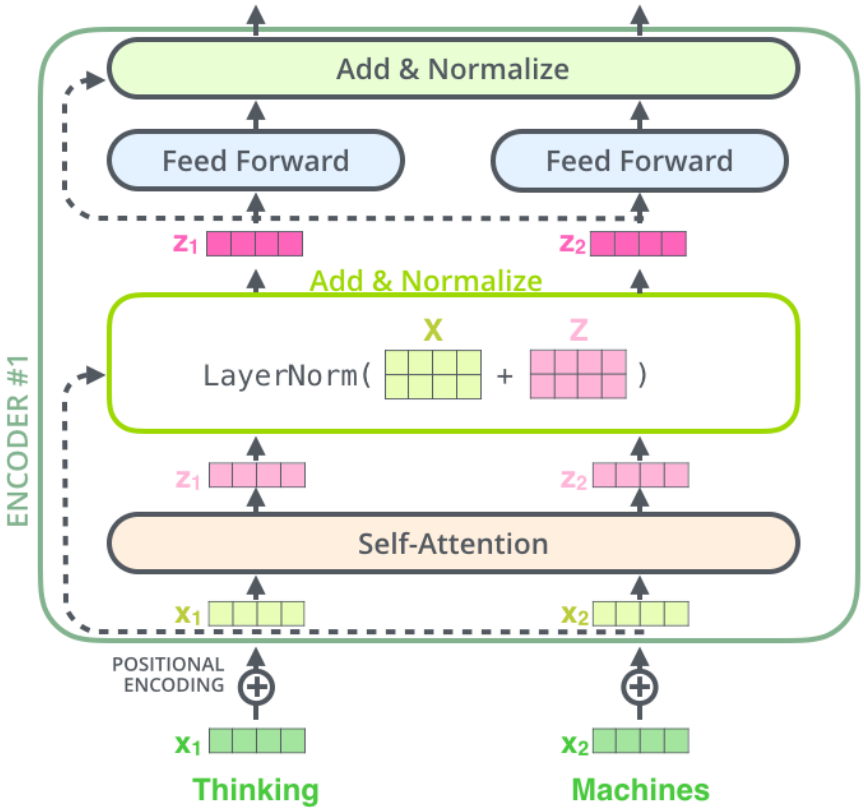

Figure 2.4: Details in Encoder[5]: The Residual Architecture.

two sub-layers in both the Encoder and Decoder. Layer normalization is followed. Skip connections or residual connections are used to allow gradients to flow through a network directly, without passing through non-linear activation functions. Skip connections form conceptually a 'bus' which flows right the way through the network, and in reverse, the gradients can flow backward along with it too.

Normalization helps with the problem called the internal Covariate Shift. Internal Covariate Shift refers to a co-variate shift occurring within a neural network. The reason why this happens is that as the network learns and the weights are updated, the output distribution of specific layers in the network will also change. This forces higher layers to adapt to the drift, which slows down learning. After normalizing the input in the neural network, we don't have to worry about the scale of input features being terribly dissimilar.

Unlike Batch Normalization normalizes the input features across the batch dimension, the key feature of layer normalization is that it normalizes the inputs across the features and is independent of other examples. Also in batch normalization, the statistics are computed across the batch and are the same for each example in the batch. Whereas, the statistics are computed across each feature and are independent of other examples in

[5]Alammar, Jay (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer

Layer Normalization.

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij} \tag{2.5}$$

$$\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{ij} - \mu_i)^2 \tag{2.6}$$

$$\hat{x_{ij}} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma^2 + \epsilon}} \tag{2.7}$$

where x_{ij} is the $i, j-$th element of the input, the first dimension represents the batch and the second represents the feature.

2.3 CSLS

Most existing methods retrieve translations as the nearest neighbors of the source word/sentence in the cross-lingual embedding space based on cosine similarity or other distances measuring similarity. Hubness(Dinu et al., 2014, Shigeto et al., 2015), a phenomenon observed in high-dimensional spaces where some points (known as hubs) are the nearest neighbors of many other points, has been reported to affect cross-lingual models. Conneau et al., 2017 proposes an alternative similarity measure called cross-domain similarity local scaling (CSLS). We denote $N_T(Wx^s)$ as the neighborhood associated with a mapped source word embedding Wx_s and all K elements of $N_T(Wx^s)$ are words from the target language. Similarly $N_s(x^t)$ is used to denote the K source neighborhood of a word x^t. Consider the mean similarity of a source embedding x^s to its target neighborhood as:

$$r_t(Wx^s) = \frac{1}{K}\sum_{x^t \in N_T(Wx^s)} cos(Wx^s, x^t) \tag{2.8}$$

where $cos(..)$ is the cosine similarity. It is apparently that $r_t(Wx^s)$ is large when K neighborhood are all close to source word projection. Likewise, the mean similarity of a target word x^t to its neighborhood is denoted as $r_s(x^t)$. Then the similarity measure CSLS is defined as:

$$CSLS(Wx^s, x^t) = 2cos(Wx^s, x^t) - r_t(Wx^s) - r_s(x^t) \tag{2.9}$$

This process increases the similarity associated with isolated word vectors but decreases the similarity of vectors lying in dense areas, in other words, the hubs. CSLS has been shown to significantly increase the accuracy of bilingual lexicon induction and is nowadays mostly used in lieu of cosine similarity for nearest neighbor retrieval.

Chapter 3

Model

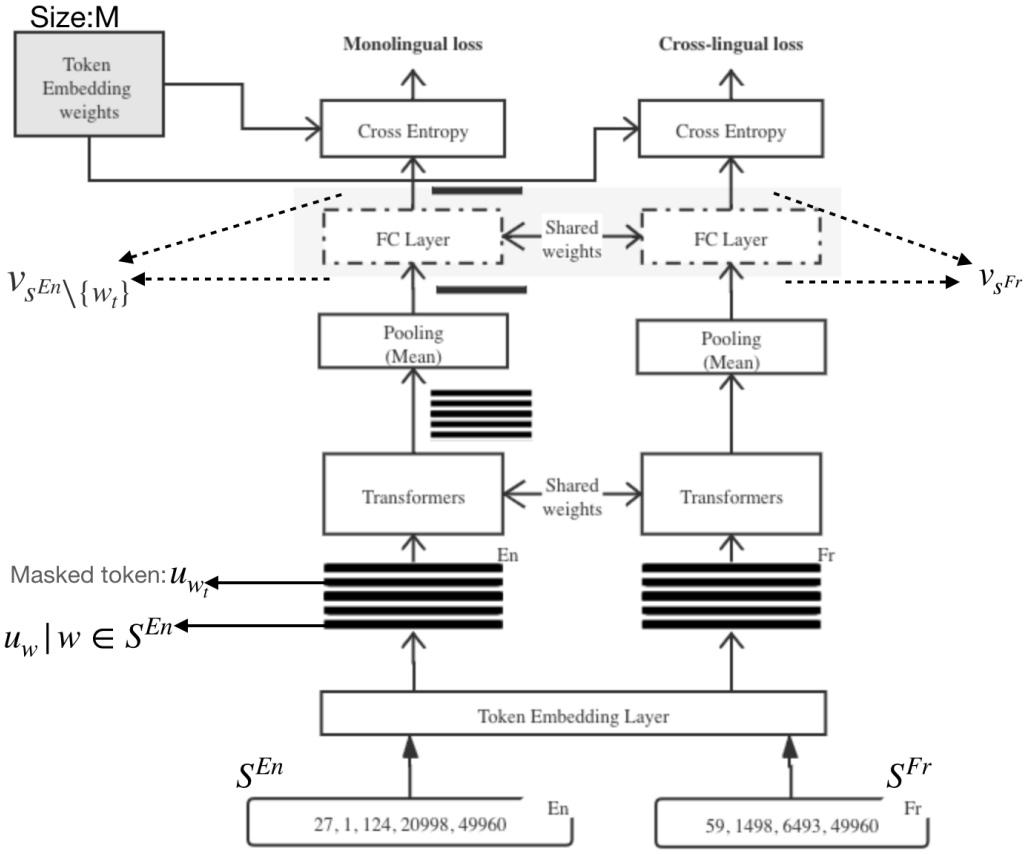

Figure 3.1: Model Architecture. Here SentencePiece[2]tokenizer would tokenize raw text (English and French in our case) and then map tokens into ids. One token would be masked randomly according to rules. u_w and v_s are vectors to represent token w and sentence s respectively. w is the token from S^{En} from which you could get full English sentence. FC_layer can be added or not added to the final layer before the criteria, which contributes to 2 different models and 2 different sentence representations when FC_layer was added.

Our Bi-Bert2Sent model shown in Figure 3.1 is a cross-lingual extension of Bert2sent proposed by Reimers and Gurevych, 2019 based on siamese Transformers (Vaswani et al.,

[2]Source: https://github.com/google/sentencepiece

2017). Our objectives are learned from Sabet et al., 2020 but we add one new objective focused on sentence alignment.

Shown in Table 3.1, our Bi-Bert2Sent uses a small size of training data and contains only 2 layers Transformers with 256 dimensions of embedding. 5 stacked-BiLSTM (Artetxe and Schwenk, 2019) represents sentence by 1024 dimension vector but runs in serial. XLM uses 12-layer Transformers and 1024 hidden units while mBERT uses 12 layers Transformers and 768 hidden units. Also, compared with XLM and mBERT who also use the Transformers to encode sentence pairs, Bi-Bert2Sent encodes each sentence among one sentence pair independently and parallel.

Model name	Base structure	embedding size	depth
Bi-Bert2Sent	Transformers	256	2
LASER	Bi-directional LSTM	1024	5
mBERT	Transformers	768	12
XLM	Transformers	1024	12

Table 3.1: Models Information. Here the depth means the number of the basic layer used in the model.

Bi-Bert2Sent model takes a parallel dataset (English and French sentence pairs) with one random-masked token as input and then encodes the sentence pair into vectors separately. A Transformer encodes a sequence and outputs one vector for each token hence we applied a pooling layer to represent a sentence by averaging length-variable vectors into a fixed-length vector. Formally, our model encodes an input English sentence S^{En} as an embedding $v_{s^{En}}$ by Transformers-like encoder and a pooling layer in the following manner:

$$v_w = Transformers(u_w)$$
$$v_{s^{En}} = Pool(\{v_{w_t} | w_t \in S^{En}\})$$

where w is the token in shared vocabulary, u_{w_t} is the t^{th} in English token embedding vector and S^{En} is the set of tokens of English sentence. Furthermore, $Transformers$ includes self-attention part which uses all input sequences S^{En} rather than just u_w to output v_w. The sentence representation could also be $v_{s^{En}} = FC\_layer(Pool(\{v_{w_c} | w_c \in S^{En}\}))$ which will be delved into in chapter 4.

3.1 Pre-processing

Like current neural machine translation models and pre-trained language models like BERT and GPT-2, the tokenization is done by the SentencePiece sub-word tokenization. Subword tokenization strikes a balance between the two approaches by using a mixture of character, sub-word and word tokens, depending on how common they are. In this way, a shorter (on average) representations of sentences can be used, yet are still able to encode rare words.

This is a data-driven tokenization method that aims to achieve a balance between vocabulary size and out-of-vocabulary words. SentencePiece sub-word tokenization has two desirable properties for multilingual language modeling:

- Sub-words represent inflections more naturally, including common prefixes and suffixes and, are thus well-suited for morphologically rich languages.

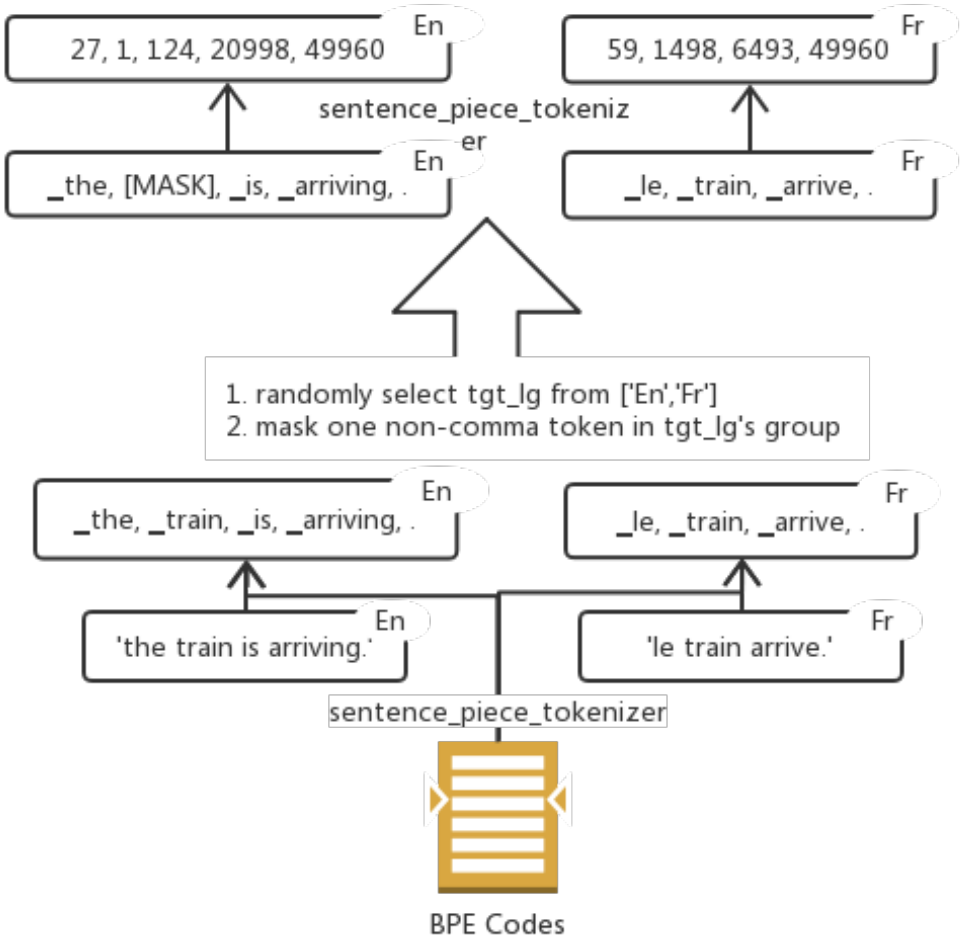

Figure 3.2: Preprocessing process from Raw text to token ids

- Sub-word tokenization is a good fit for open-vocabulary problems and eliminates out-of-vocabulary tokens, as the coverage is close to 100% tokens.

We use a unigram language model established on ParaCrawl (Esplà-Gomis et al., 2019)[3] that learns a shared vocabulary of tokens together with their probability of occurrence. It suspects that tokens occur independently (hence the unigram in the name). During tokenization, this method finds the most probable segmentation into tokens from the vocabulary and stored the codes for further usage.

In our case, we first train a tokenizer using the ParaCrawl dataset in English and French and get a shared BPE code. The codes are used to tokenize raw text into tokens and each token can be converted into one id. Like the process shown in Figure 3.2, two parallel sentences will be tokenized into pieces along with one random-masked token converted to [MASKED] for each sentence pair. Moreover, the masked token would be different to enhance our dataset for each epoch.

[3]Source: http://opus.nlpl.eu/ParaCrawl-v5.php

3.2 Embeddings Layer

Like other NLP tasks, Bi-Bert2sent passes each input token through a Token Embedding layer so that each token is mapped into a vector representation. Like BERT, Bi-Bert2sent has additional embedding layers in the form of Positional Embeddings.

The input text is first tokenized before it passes to the Token Embeddings Layer, which converts the tokenized ids into vectors of embedding size. We experiment models with embedding size 256 and 128, which is detailed described in chapter 4 .

Since tokens are equally encoded in the proposed architecture, it is necessary to tell the model the order of each token. In the proposed architecture, all tokens of the input sequence are fed to the network with no special order or position unlike common RNN or ConvNet architectures, thus, the model has no idea how the tokens are ordered. Consequently, a position-dependent signal is added to each token-embedding to help the model incorporate the order of words. For Positional Embeddings, we allow our model to learn during training rather than using fixed cosine or sine signals added into token embeddings, which is used in BERT(Devlin et al., 2018).

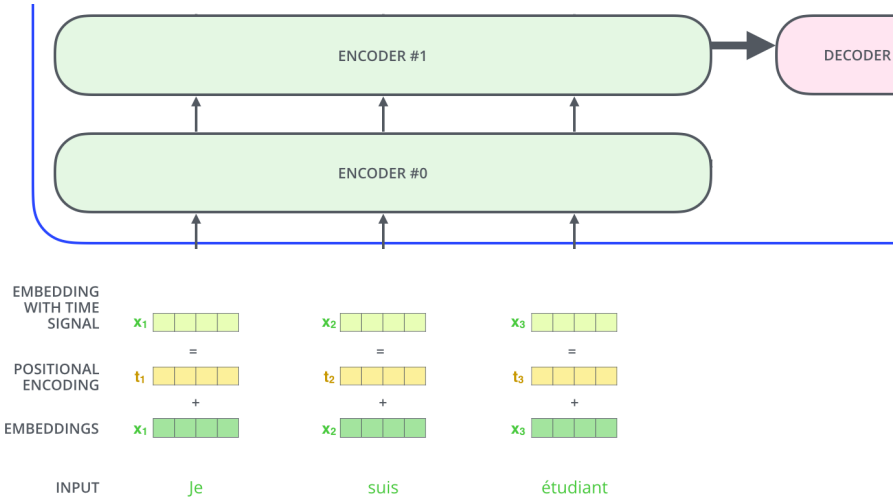

Figure 3.3: Embedding Layer[4]: Token Embedding Layer and Positional Embedding

3.3 Encoder

In our **Bi-Bert2Sent** model, we used a 2-layer, 256-hidden, 8-heads Transformers' encoder rather than 12-layer, 768-hidden, 12-heads used in BERT-Base(Multilingual Cased) [5]. The allowed maximum length of sequence is set to 200 according to our dataset distribution, shown in Figure 3.4.

3.4 Losses

We experiment 2 level losses on token and sentence where sentence alignment loss is optional. However, since the models using the second one Equation 3.5 perform much

[4]Alammar, Jay (2018). The Illustrated Transformer [Blog post].
[5]The data is from: https://github.com/google-research/bert/blob/master/multilingual.md

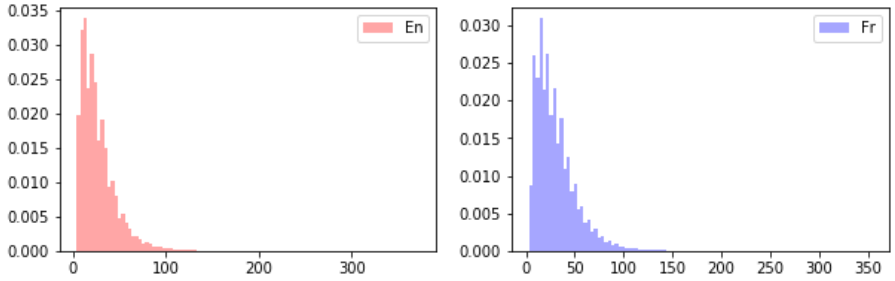

Figure 3.4: Sentence length distribution. It covers 99.9% samples when max_length is 200.

better than those using Equation 3.4, our evaluation tasks are conducted on models using Equation 3.5.

3.4.1 Monolingual and cross-lingual losses

The Bi-Bert2Sent training objective aims to predict a masked word token w_t^{En} or w_t^{Fr} where the target language En or Fr is firstly randomly selected with equal likelihoods and then we will mask a token in the selected language.

To formulate the training objective, we use cross-entropy loss $l : -\sum_{c=1}^{M} y_c log(p_c)$. More precisely, suppose we masked a token from English side [6], for a raw English sentence s^{En} with one masked token $w_t \in S^{En}$, the training monolingual objective for Bi-Bert2Sent is given by:

$$min\left(-log^{\frac{product_t}{\sum_{c=1}^{M} product_c}}\right) \tag{3.1}$$

where $product_c$ is an inner product (scores) between masked token w_t from S^{En} and encoded output of the rest tokens from English sentence except w_t.

$$product_c^{mono} = \langle u_{w_c}, v_{s^{En}\setminus\{w_t\}} \rangle \tag{3.2}$$

where, M is the size of shared vocabulary and $v_{s^{En}\setminus\{w_t\}}$ is the output just before the final Cross-entropy layer.

We adapt the Bi-Bert2sent model to bilingual corpora by introducing a cross-lingual loss in addition to the monolingual loss since it would be possible to predict the masked token using the information in French sentence when the parallel corpus is available. The cross-lingual objective differs only in $product_c$:

$$product_c^{cross} = \langle u_{w_c}, v_{s^{Fr}} \rangle \tag{3.3}$$

Note that the tensors used for the token losses are the final output just after the previous structures.

XLM mentioned in Conneau et al., 2019 uses a similar train objective called Translation Language Model to predict masked tokens in parallel sentences. Since they inherit Masked Language Model from BERT, they concatenate multiple sentences together, which disables their model to represent one sentence individually.

[6] We will suppose that the masked token is from an English sentence token set all the time in the report.

3.4.2 Sentence alignment loss

Besides the losses above, we also experiment with sentence alignment loss which aims at adding more strong supervision information and tells the model the true sentence pairs. For this loss, we designed and experimented 2 different objectives:

- Binary classification: score every English sentence with French sentences and predict if each sentence pair is similar or not.

  we use logistic loss $l : log(1 + e^{-x})$ in conjunction with negative sampling within one batch. In a batch, the corresponding sentence alignment loss for one sample at k is given by:

$$min\left(l(\langle v_{s_k^{En}\setminus\{w_t\}}, v_{s_k^{Fr}}\rangle) + \frac{1}{\#neg\_samples}\sum_{s'\in N_{s_k^{Fr}}} l(-\langle v_{s_k^{En}\setminus\{w_t\}}, v_{s'^{Fr}}\rangle)\right) \tag{3.4}$$

  where $v_{s_k^{En}}$ and $v_{s_k^{Fr}}$ are sentence pair for English and French respectively. Here $N_{s_k^{Fr}}$ is uniformly sampled from French samples within one batch except the $v_{s_k^{Fr}}$. Note that for each sample, its masked token comes from different language sentence and here we use English as an example.

- Batch-size classification based on cross-entropy loss.

  We used the inner products between sentence vectors as the scores and apply the cross-entropy loss by row or by column. To be specific, the first English sentence within one batch can be calculated similarities with all French sentences within one batch and the correct label for these batch size similarity scores is 1 since only the first similarity score comes from the correct sentence pair (first English sentence with its corresponding French sentence). More precisely, the objective in one batch should be:

$$\begin{aligned} min\Big\{ &-(\sum_{k=I_1}^{I_B}\sum_{i=I_1}^{I_B} label_i \log\frac{\langle v_{s_k^{En}\setminus\{w_t\}}, v_{s_i^{Fr}}\rangle}{\sum_{j=I_1}^{I_B}\langle v_{s_k^{En}\setminus\{w_t\}}, v_{s_j^{Fr}}\rangle}) \\ &-(\sum_{k=I_1}^{I_B}\sum_{i=I_1}^{I_B} label_i \log\frac{\langle v_{s_k^{Fr}}, v_{s_i^{En}\setminus\{w_t\}}\rangle}{\sum_{j=I_1}^{I_B}\langle v_{s_k^{Fr}}, v_{s_j^{En}\setminus\{w_t\}}\rangle})\Big\} \end{aligned} \tag{3.5}$$

  where $\{I_1, I_2, \ldots, I_B\}$ is a subset of indices for the sentence sets s^{En} and s^{Fr} ($s_{I_i}^{En}$ and $s_{I_i}^{Fr}$ consist of a sentence pair), B is the batch size, *label* is one-hot vector with size B and $label_{i\neq k} = 0, label_{i=k} = 1$.

3.5 Implementation Details

We build our model under the Pytorch framework with the help of Transformers[7] implemented by Hugging Face. The model parameters are updated by Adam.

Our model is trained on the ParaCrawl (Esplà-Gomis et al., 2019) v4.0 datasets for the English-French language pair, of which there are about 32 million sentence pairs and 665 million English tokens. All the sentences were tokenized using SentencePiece Tokenizer[8] by Google with a shared vocabulary and the size of this vocabulary is 50'000.

[7] https://github.com/huggingface/transformers
[8] Source: https://github.com/google/sentencepiece

Since there are two types of sentence alignment loss and a fully-connected layer can be added or not into the model simply after the Transformers' encoder and before the final Softmax layer, we trained several models separately and additionally experimented two approaches to representing sentences.

Chapter 4

Evaluation

As our model encodes the whole sentence at once in parallel and the model can understand a word in its context, each word representation coming out of our encoder now includes contextual information about the surrounding phrases of the word. There is no appropriate approach to evaluate the quality of the word. Thus, our evaluation task is to assess the quality of the sentence embeddings obtained as well as the quality of alignment between the two-sentence embedding sets, we compare our results using the following benchmarks:

- Cross-lingual sentence retrieval
- Multi-lingual document classification
- Cross-lingual natural language inference

Besides, we also analyze the effect of representation vectors, objectives and model structures in the view of using models' performance on the cross-lingual sentence retrieval task.

Our evaluation tasks are based on sentence embeddings obtained from the outputs of the pooling layer or the fully-connected layer, the v_s shown in Figure 3.1. We use the code available in the MUSE library[1] (Conneau et al., 2017) for cross-lingual sentence retrieval task. The MLDoc dataset[2](Schwenk and Li, 2018) is used for multilingual document classification while XNIL[3] is used for cross-lingual natural language inference task.

4.1 Cross-lingual sentence retrieval

We test sentence embeddings obtained from the outputs of the Transformers or the fully-connected layer for sentence retrieval on the Europarl corpus[4]. In particular, we use $v_{s^{En} \setminus w_t}$ shown in Figure 3.1 as the sentence embeddings. We consider 2000 sentences in the source language dataset and retrieve their translation among 200K sentences in the target language dataset. Results for P@1 of benchmarks vs our models are included in Table 4.1.

For sentence translation retrieval task, the 'fr-en' p@1 evaluated on our best model *Bi-Bert2sent - CSLS* is significantly larger than that of *MUSE* and it also outperforms

[1]Source: https://github.com/facebookresearch/MUSE
[2]Source: https://github.com/facebookresearch/MLDoc
[3]Source: https://github.com/facebookresearch/XNLI
[4]Source: https://www.statmt.org/europarl

Method	en-fr	fr-en
MUSE	69.2	68.8
TRANSGRAM	80.4	81.6
BI-SENT2VEC uni. + bi. NN	86.1	83.9
BI-SENT2VEC uni. + bi. CSLS	**87.8**	87.4
Bi-Bert2sent - NN	74.2	85.6
Bi-Bert2sent - CSLS	84.2	**87.6**

Table 4.1: Cross-lingual Sentence Retrieval Results. We report P@1 scores for 2'000 source queries searching over 200'000 target sentences under two retrieval criterion: NN for Nearest Neighbor and CSLS for Cross-Domain Similarity Local Scaling(Conneau et al., 2017). 'en-fr' denotes retrieving correct French sentence given English sentence, vise versa. The results of MUSE and TRANSGRAM are from Sabet et al., 2020

BI-SENT2VEC uni. + bi. CSLS by 0.2%. Moreover, when using NN as retrieval criteria, our model *Bi-Bert2sent - NN* goes beyond the corresponding model *BI-SENT2VEC uni. + bi. NN* as well. Although both two models of us underperform those of *Bi-Bert2sent* when retrieving correct French sentences, there is a great deal of difference in p@1 between *Bi-Bert2sent - CSLS* and *MUSE* or *TRANSGRAM*.

4.2 Multilingual document classification

The Multilingual Document Classification task better known as MLDoc (Schwenk and Li, 2018) aims at classifying news documents. There are 1,000 training and development documents and 4,000 test documents for each language, divided into 4 different genres. To measure the robustness of zero-shot document classifier transfer, we train the classifier on embeddings in the source language and report the accuracy of the same classifier applied to the target languages. The classifier is a feed-forward neural network with two hidden layers of size 64 and 16 respectively with dropout rate 0.2 and is optimized with the Adam optimizer.

We compare the performance of Bi-Bert2sent with the BiLSTM who is used as LASER sentence embeddings(Artetxe and Schwenk, 2019). LASER sentence embedding model is a multi-lingual sentence embedding model which is composed of a BiLSTM encoder and an LSTM decoder. It uses a shared byte pair encoding based vocabulary of 50k operations. The model LASER was trained over 93 languages compared to our model which train on 32M sentences for only English and French languages.

Method	en-fr	fr-en
LASER	80.1	78.0
Bi-Sent2Vec	82.2	81.6
Bi-Bert2sent	**83.3**	**81.6**

Table 4.2: MLDoc Benchmark results (Schwenk and Li, 2018). A document classifier was trained on one language and tested on another without additional training/fine-tuning. Here 'en-fr' means training on English corpus and testing on French text.

4.3 Cross-lingual natural language inference

XNLI(Conneau et al., 2018b) is an evaluation corpus for language transfer and cross-lingual sentence classification in 15 languages, which is to evaluate cross-lingual sentence understanding methods. The Cross-lingual Natural Language Inference (XNLI) corpus is a crowd-sourced collection of 5,000 test and 2,500 dev pairs for the MultiNLI corpus. The pairs are annotated with textual entailment and translated into 14 languages including French.

In Table 4.3, we evaluate our model only on zero-shot cross-lingual classification. We report the **XNLI baselines** of Conneau et al., 2018b, the **multilingual BERT** approach of Devlin et al., 2018, **LASER** from Artetxe and Schwenk, 2019 and the recent work **XLM** of Lample and Conneau, 2019. All hyper-parameters were optimized on the English XNLI development corpus only, and then, the same classifier was applied to French of the XNLI test set.

Method	en	fr	Fine-tune strategy
mBERT	81.4	-	add a dense layer
XLM(MLM)	83.2	76.5	add a dense layer
XLM(MLM+TLM)	85.0	78.7	add a dense layer
XNLI(baseline)	73.7	67.7	use a new MLP model
LASER	73.9	71.9	use a new MLP model
Bi-Bert2sent	63.4	59.8	use a new MLP model

Table 4.3: Test accuracies on the XNLI cross-lingual natural language inference dataset. All other results are from Lample and Conneau, 2019. The last column shows the fine-tune strategy of 6 models for the zero-shot classification task. The first three models add a dense layer on the top of their models as a classifier and fine-tune them together and the last three fine-tune a new multi-layer perceptron.

However, these 6 results are not fully comparable because of different fine-tune strategies. Shown in Table 4.3, in the first three models, one dense layer is added to the end of the original one, which is the classifier for zero-shot classification and then the classifier is fine-tuned on the NLI dataset. In that way, the pre-trained architecture of the first three models contributes to the final classification decision as well.

While the rest three models including ours use the NLI training dataset to train a new multi-layer perceptron classifier whose inputs are the usual combination of the two-sentence embeddings: $(p,\ h,\ p \cdot h,\ |p - h|)$, where p and h are the premise and hypothesis. Note, moreover, that the multilingual sentence embeddings p and h generated from the pre-trained models are fixed and not fine-tuned on the task or the language to do the zero-shot classification task. Due to the time limitation, the experiment on mBERT using the second fine-tune strategy has no chance to finish.

num_layer	2	3	4
XNLI	56.8	57.2	57.9

Table 4.4: Impact of the depth. We report the accuracy on French test dataset. These results are all from the models trained just 3 epochs.

It must be mentioned that our model performs the worst among all mainly because our pre-trained model uses 2 layers Transformers encoder with embedding size of 256 compared with 5-stacked BiLSTM of 1024 dimensions used by LASER. To support that,

we experimented with our models with different settings on the number of layers. The results are shown in Table 4.4. You can see an increase in accuracy occurred when the number of Transformers layer increments. It is easier for models with deeper Transformers structures to learn about syntax and high-level linguistic features.

To sum up, there are different approaches to training the classifier. The first three methods benefit from their powerful models to better understand the deeper meanings inside sentence pairs. That might be the reason why they outperform other models dramatically. However, for the rest models (XNLI baseline, LASER, Bi-Bert2Sent), the accuracy of classification seems to be limited by the 2 hidden-layer feed-forward network because of the model's capacity. It is also mentioned by Artetxe and Schwenk, 2019 that the XNLI task does benefit from deeper models. Sufficient depth is required to efficiently learn high-level linguistic features which is exactly what the XNLI task relies on.

4.4 Experiments

Due to the time limitation, we didn't do a normal grid-search on hyper-parameters of our model since it takes 18 hours to train 1 epoch on one single Titan X 12GB GPU. We trained several models from one starting point and we have shown the results obtained from the best model. Training the model with more choice of num_layer, emb_size, and num_head can be one future work.

In order to analyze the impacts of our designed objectives, structures, and some hyper-parameters, our experiments are as follows as shown in Table 4.5.

Has FC_layer whether there is a fully-connected layer after a transformer

Has Sentence Loss whether our designed sentence alignment loss Equation 3.4 or Equation 3.5 is optimized with token losses Equation 3.1 or not.

Use pooled_out whether using pooled out as sentence representations. Only when FC_layer exists, the vector used to represent sentences can be different. If FC_layer exists, both *pooled_out* and *FC_out* can be used as sentence representations.

emb_size the embedding size of the sentences, that is, the dimension of the sentence representation. The default value is 256.

num_layer as a transformer consists of the quantity-definable encoders and decoders, we use num_layer as the encoder's quantity we used in our model. num_layer is 2 in default.

We use the sentence retrieval task to explore deeper on our model including the structures and objectives under settings shown in Table 4.5. Note that the graphs below are p@1 behavior curves as training time increases. Each graph shown below can be regarded as one control experiment.

In the following figures, the 'nn' denotes 'nearest neighbors' and 'CSLS' is Cross-Domain Similarity Local Scaling. 'en-fr' means English sentences are the queries and French sentences are the keys, that is to say, we are supposed to find the corresponding French sentence given the English sentence, vice versa.

Condition	Has FC_layer	Has Sentence Loss	Use pooled_out	emb_size	num_layer
model01	False	False	—		
model02	False	True	—		
model03	True	False	True		
model04	True	True	True		
model05	True	False	False		
model06	True	True	False		
model07	True	False	True	128	
model08	True	True	True		3

Table 4.5: Models setting. The first row shows the settings for models. The sentence alignment loss used here is Equation 3.4. Note that the blank indicates the default value.

4.4.1 Fully-Connected layer and Proper Position of Sentence Embeddings

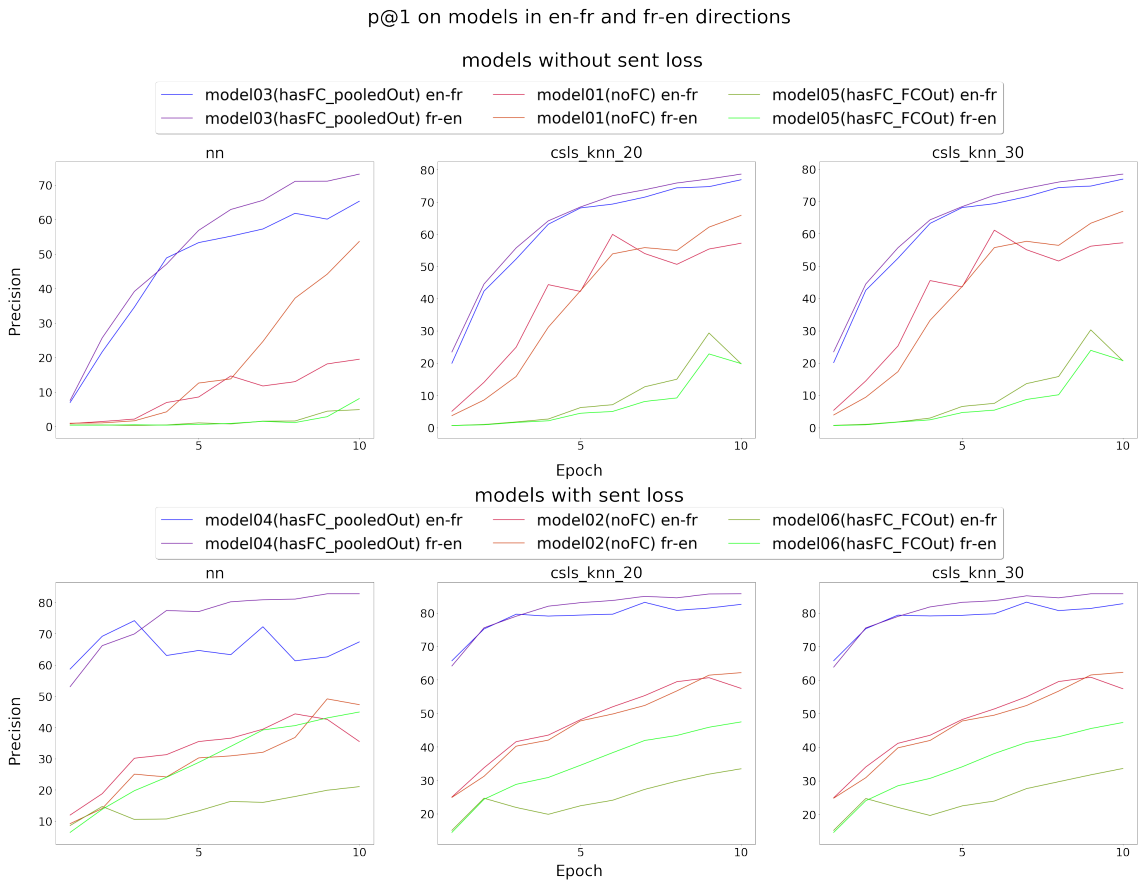

Figure 4.1: The Impact of the FC layer. The figure above shows the results of models without sentence alignment loss while the figure below shows the results of models with it. Here 'hasFC'('noFC') means this model has(hasn't) a fully-connect layer before the Sotmax layer; 'FCOut' denotes using the FC_out as sentence embeddings while 'pooledOut' uses $pooled\_out$ as sentence embeddings.

In common use of BERT[5], there is always a fully-connected layer after the pooling layer. We experiment with several models with or without the fully-connected layer just as shown in Figure 3.1 see the effect of the fully-connected layer.

[5]Implementation provided by Google: https://github.com/google-research/bert

In Figure 4.1, the three models of each experiment optimize the same objective function, however, since they use the fully-connected layer differently, they represent sentences differently. It is obvious that model03(model04) outperforms the rest two models and the results of model05(ex06) are the worst. This phenomenon can be explained by the presence of a fully-connected layer as well as the vector we used to represent the sentences.

When we add the fully-connected layer before the Softmax layer, the output of the fully-connected layer is more likely to focus on tokens and loses some sentence information because this output is closer to the final layer who predicts the masked token. In this case, the vector before the fully-connected layer is likely to contain more information about the whole sentence to some extent. Thus, the models used pooled_out (model03 and model04) beat these (model05 and model06) who use the output of the fully-connected layer as sentence embeddings.

But if we don't put a fully-connected layer after the pooling layer(model01 & model02), the pooled_out vector is fed into the final layer directly and predicts the masked token. Thus the pooled_out must contain more token information to fulfill the mission of prediction rather than sentence information. Of course, the FC layer brings model complexity and it might help the performance as well.

Overall, benefiting from the fully-connected layer, model03 (model04) can represent the entire sentence as much as possible because they have the chance to stay far away from the final prediction layer whose mission is about the token.

4.4.2 Model Complexity

Dimensionality selection for embedding is a well-known open problem. The impact of dimensionality on the embedding has not yet been fully understood. As a critical hyper-parameter, the choice of dimensionality for sentence vectors has a great influence on the performance of a sentence embedding. First, it directly impacts the quality of sentence vectors - a sentence embedding with a small dimensionality is typically not expressive enough to capture all possible semantic information, whereas one with a very large dimensionality suffers from over-fitting. Second, the number of parameters for a sentence embedding or a model that builds on sentence embeddings (e.g.recurrent neural networks) is usually a linear or quadratic function of dimensionality, which directly affects training time and computational costs. Therefore, large dimensionalities tend to increase model complexity, slow down training speed, and add inferential latency, all of which are constraints that can potentially limit model applicability and deployment (Wu et al., 2016).

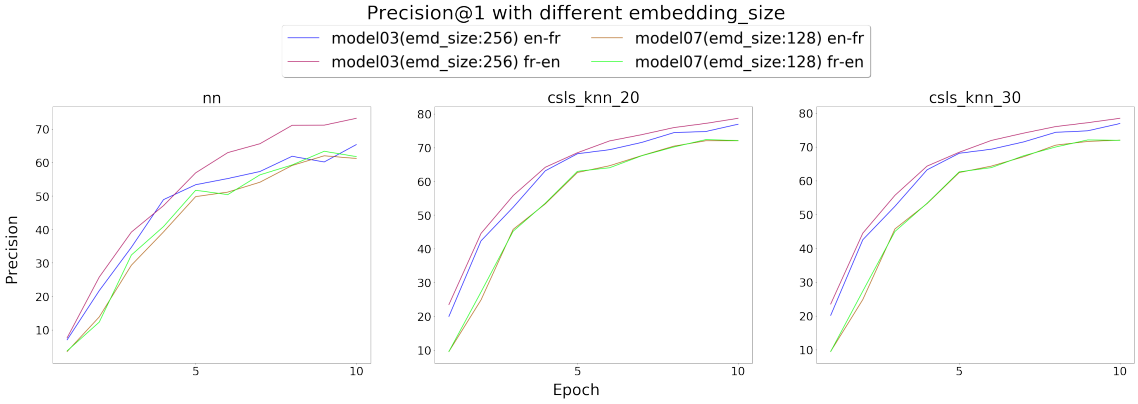

Figure 4.2: The Impact of the embedding size. Here the embedding size of model03 is 256 while it is 128 for model07. The other settings are the same for both two.

It is expected to see the results in Figure 4.2 since model07 with 128 embedding size is supposed to underperform model03 with 256 embedding size before overfitting. However, a great difference between the p@1 of 'en-fr' and 'fr-en' appears in model03 (at epoch 5) while such a phenomenon hasn't shown in model07 yet.

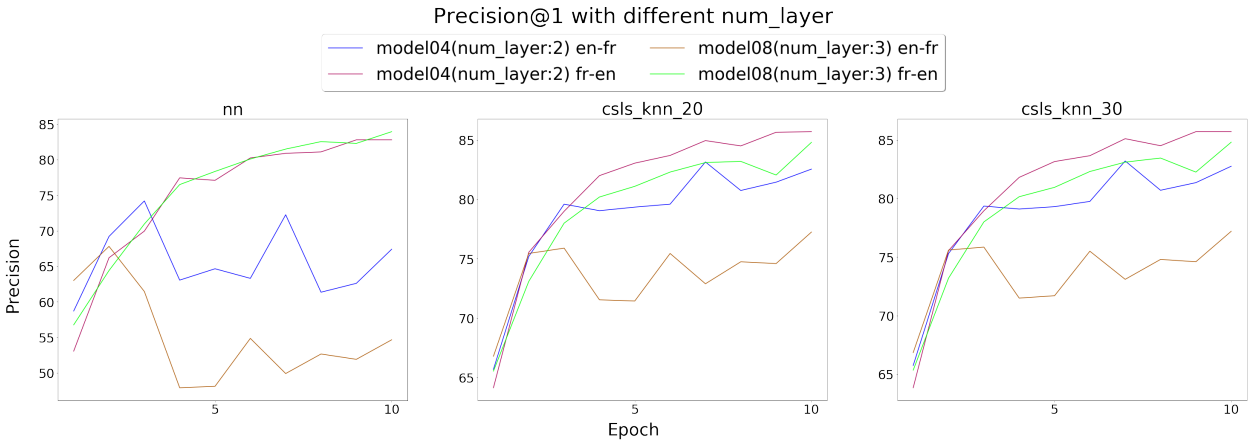

Figure 4.3: The Impact of the number of layers. model08 has 3 Transformers layers but model04 has only 2.

The same as the embedding size shown in Figure 4.3, as the number of layers increases, the model complexity goes up as well. This experiment supports that the difference between p@1 on 'en-fr' and 'fr-en' is caused by overfitting as well.

4.4.3 Sentence Alignment Loss

In this section, we will discuss the impact of sentence alignment loss when other factors are the same. Two models that appeared on one graph(experiment) are under the same setting except for the usage of sentence alignment loss. The settings like the usage of the fully-connected layer at the end and the vector used to represent sentences are under control while the usage of sentence alignment loss is the factor to be tested, that is to say, one of them optimizes token losses with sentence loss while the rest one is trained without sentence alignment loss.

Fine-tune Weights on Sentence alignment loss

From the graphs above, it can be observed that the performance of the models on English to French sentence retrieval always decreases after several training-epochs. It is appeared to be overfitting since this decrease hasn't shown in the results (Figure 4.2) of the model with a smaller embedding size (model07). The conclusion of overfitting can also be proven by an earlier appearance of the decrease phenomenon that happened in model08 who uses 3 layers Transformers, shown in Figure 4.3. In order to mitigate the p@1 decrease appeared in 'en-fr' direction, we try to rescale the weight on the sentence alignment loss.

From Figure 4.4, we can see re-setting a higher weight on the sentence alignment loss is shown to correct the decrease that appeared at epoch 3. In view of using CSLS, sufficient supervision on sentence-level helps the p@1 of the sentence retrieving task. On the other hand, sentence alignment loss brings hubness to some extent. Of course, the weight on sentence alignment loss should not be as high as possible according to the results we have because model04_10 is not the best model here.

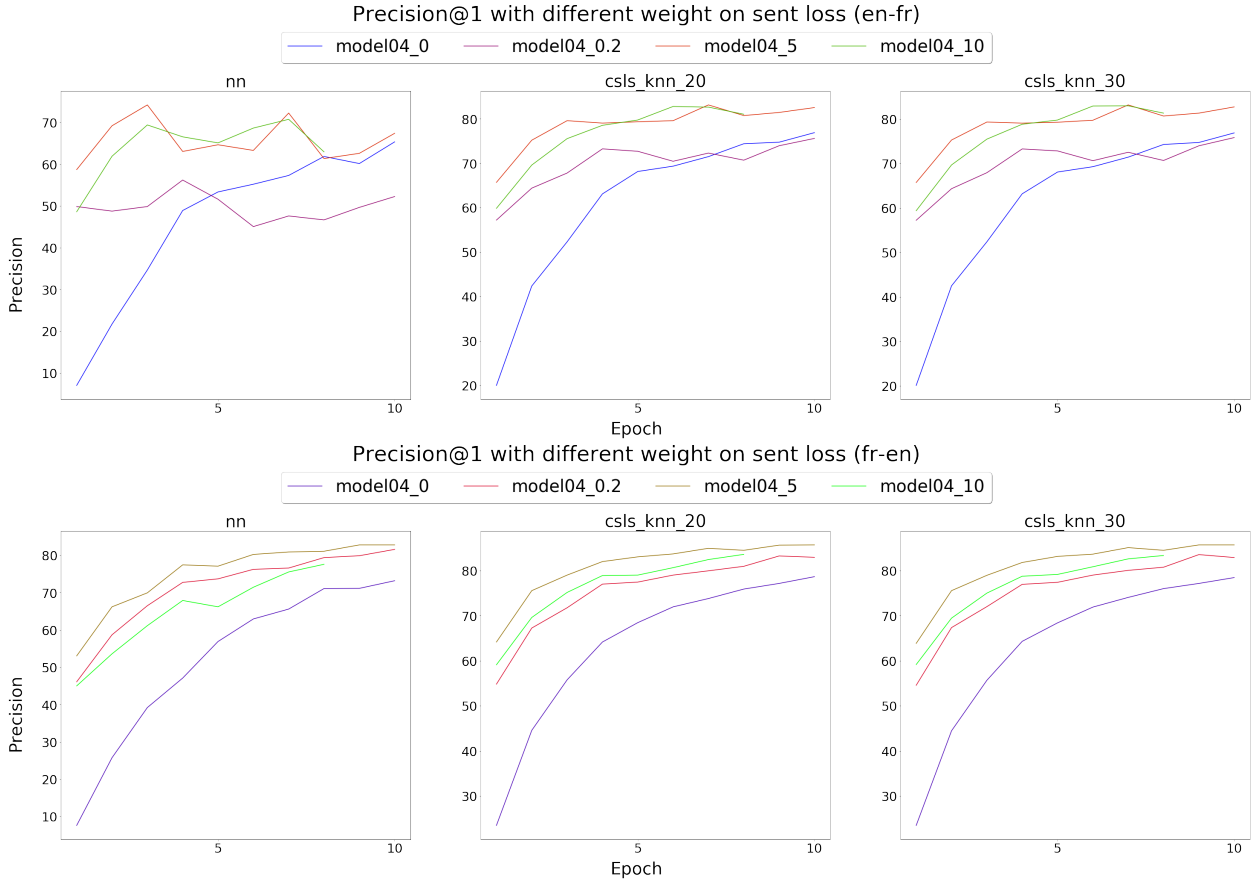

Figure 4.4: Performance under different loss weights. All models follow the setting of having an FC layer and using the pooled_out. The suffix of the model name is the weight of sentence alignment loss, e.g. model04_0.2 means the proportion of sentence and token losses is 0.2:1.

The effect of sentence alignment loss

Shown in Figure 4.5, we could see the same pattern on all of the experiments, where the models optimizing sentence alignment loss do have a better beginning than the models not optimizing it. However, a considerable increase occurred on the p@1 obtained from models without sentence alignment loss and they seem to catch up with the models with sentence loss later gradually.

The results make sense because optimizing the sentence loss forces the model to align sentences from start. When using sentence-level loss, the correct sentence pairs are forced to be closer and then the p@1 are high at the beginning. However, if we don't use the sentence alignment loss and don't add such strong supervision on sentence-level, the model can also learn to align sentences by aligning words via cross-lingual loss. That is to say, aligning words will eventually affect sentences but it needs time.

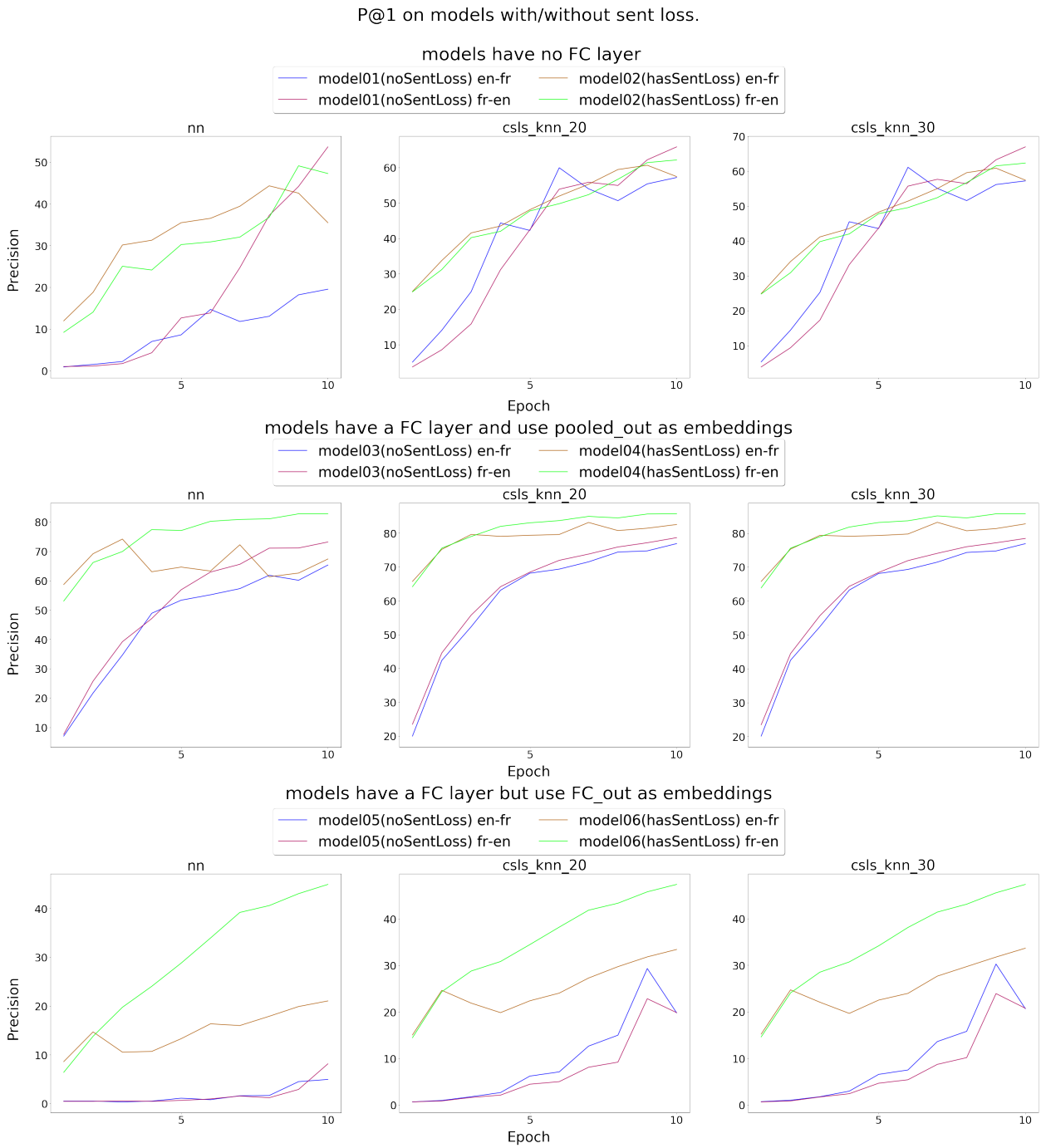

Figure 4.5: The Impact of the sentence alignment loss. The top figure are models without fully-connected layer. The middle one are models with FC layer and they use pooled_out as sentence embeddings. The last one are models with FC layer but they use output of FC layer as embeddings. model01, model03 and model05 optimize sentence alignment loss while the rest three don't optimize it.

Chapter 5

Conclusion and future work

In this report, we introduce a cross-lingual sentence embedding model, which extends the Sentence-BERT algorithm (Reimers and Gurevych, 2019) and inherits training objectives from Sabet et al., 2020. Besides, we propose a new sentence alignment loss which is proven to have a positive impact on our Cross-lingual sentence retrieval task. We use a signal language-agnostic Transformers encoder for English and French, which is trained on publicly available parallel corpora and applied to different downstream tasks without any fine-tuning. The introduced model outperforms or is on par with its competitors on the Cross-lingual sentence retrieval task and Multi-lingual document classification task. However, it performs worse on the Cross-lingual natural language inference task.

In the future, since the models with the sentence loss (3.4) perform worse on our evaluation tasks, we should try different numbers of negative samples and see the impact on sentence embeddings. The weight on positive and negative samples is also a factor to be decided.

We used the Mean-pooling layer as the pooling approach through all experiments in our paper. As mentioned in 4.4.1, replacing the Mean-Pooling layer by the Max-Pooling layer or other pooling approaches will be a nice try in the future.

The reason why different performances are observed in two directions (retrieving translated English sentence given French sentence and retrieving translated French sentence given English sentence) is unclear. Investigations into how the sentence alignment loss effects the two directions differently can be interesting future work. Data visualization on the embedding space should be done to see the different behaviors of English sentences and French sentences and check the hubness problem. Moreover, the tokenizer can be a cause as well.

=

Bibliography

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*, 2019.

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 2019.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018a.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*, 2018b.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. Trans-gram, fast cross-lingual word-embeddings. *arXiv preprint arXiv:1601.02502*, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014.

Miquel Esplà-Gomis, Mikel L Forcada, Gema Ramírez-Sánchez, and Hieu Hoang. Paracrawl: Web-scale parallel corpora for the languages of the eu. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 118–119, 2019.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014.

Karl Moritz Hermann and Phil Blunsom. Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*, 2013.

Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*, 2014.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

Martin Josifoski, Ivan S Paskov, Hristo S Paskov, Martin Jaggi, and Robert West. Crosslingual document embedding as reduced-rank ridge regression. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 744–752. ACM, 2019.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. Loss in translation: Learning bilingual word mapping with a retrieval criterion. *arXiv preprint arXiv:1804.07745*, 2018.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, 2012.

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

Omer Levy, Anders Søgaard, and Yoav Goldberg. A strong baseline for learning cross-lingual word embeddings from sentence alignments. *arXiv preprint arXiv:1608.05426*, 2016.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Aitor Ormazabal, Mikel Artetxe, Gorka Labaka, Aitor Soroa, and Eneko Agirre. Analyzing the limitations of cross-lingual word embedding mappings. *arXiv preprint arXiv:1906.05407*, 2019.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Janarthanan Rajendran, Mitesh M Khapra, Sarath Chandar, and Balaraman Ravindran. Bridge correlational neural networks for multilingual multimodal representation learning. *arXiv preprint arXiv:1510.03519*, 2015.

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

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*, 2017.

Ali Sabet, Prakhar Gupta, Jean-Baptiste Cordonnier, Robert West, and Martin Jaggi. Robust cross-lingual embeddings from parallel sentences, 2020. URL `https://openreview.net/forum?id=SJlJSaEFwS`.

Holger Schwenk and Xian Li. A corpus for multilingual document classification in eight languages. *arXiv preprint arXiv:1805.09821*, 2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*, 2018.

Hubert Soyer, Goran Topić, Pontus Stenetorp, and Akiko Aizawa. Crovewa: Crosslingual vector-based writing assistance. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 91–95, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Mei Yang and Katrin Kirchhoff. Phrase-based backoff models for machine translation of highly inflected languages. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, 2017.