MULTILINGUAL DOCUMENT EMBEDDING WITH SEQUENTIAL NEURAL NETWORK MODELS

by

WEI LI

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

December 2021, Hong Kong

Authorization

WEI LI

MULTILINGUAL DOCUMENT EMBEDDING WITH SEQUENTIAL NEURAL NETWORK MODELS

by

WEI LI

This is to certify that I have examined the above M.Phil. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

Brian Mak

Prof. Brian Mak, Thesis Supervisor

[illegible]

Prof. Dit-Yan YEUNG, Head of Department

Department of Computer Science and Engineering

31 December 2021

ACKNOWLEDGMENTS

I feel grateful to my family, supervisor, and colleagues when I finish this thesis. The past few years have been difficult for me both physically and mentally as I encountered health problems and depression. During this hardship, I feel the warmth of the people around me. I feel incredibly grateful to the doctors and nurses in Tseung Kwan O hospital as they took good care of me when I was hospitalized there. I should also thank my colleague YingKe Zhu's help during that time. Finally, I feel Hongkong has a wonderful hospital system and professional and caring medical staff. Thanks to the hard-working people in Hongkong, I can immediately benefit from this superb system that would take decades to build in any other city.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

MULTILINGUAL DOCUMENT EMBEDDING WITH SEQUENTIAL NEURAL NETWORK MODELS

by

WEI LI

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

One of the current state-of-the-art multilingual document embedding model LASER is based on the bidirectional LSTM (BiLSTM) neural machine translation (NMT) model. This paper presents a **T**ransformer-based **M**ultilingual sentence/**D**ocument **E**mbedding model, T-MDE, which makes two significant improvements. Firstly, the BiLSTM encoder is replaced by the attention-based transformer structure with an novel information bottleneck design. The new model structure is more capable of learning sequential patterns in longer texts. Moreover, it is faster both in training and embedding generation. Secondly, we augment the NMT translation loss function with an carefully designed distance constraint loss term. It will further brings the embeddings of parallel sentences close together in the vector space. We call the T-MDE model trained with distance constraint, cT-MDE. Our T-MDE model significantly outperforms BiLSTM-based LASER in the cross-lingual document classification tasks.

CHAPTER 1

INTRODUCTION

1.1 Distributed representation of text and document embedding

Document vectorization/embedding plays a vital role in natural language processing (NLP). For example, it is usually necessary to transform text documents of variable lengths into vectors of the same length so that they can be classified or compared, as most classifiers only work on inputs of a fixed length.

Perhaps the most popular document vector is the *term frequency-inverse document frequency* (TF-IDF) feature vector [42]. It usually offers a decent baseline performance in many tasks, albeit carefully engineered, task-specific features may give better performance in a particular task [21, 22, 52, 47, 14, 29, 18, 40, 51]. Term-frequency-based document vectorization makes two assumptions [6, 28]: (a) occurrences of each term are mutually independent, and (b) a document is treated as a "bag of words" and different permutations of the same set of words are considered to be the same. These assumptions suffer from a major drawback: it ignores sequential patterns in a document, which can be important in some NLP tasks such as genre classification. For example, 'Wall' and 'Street' in the named entity 'Wall Street' are treated as independent words in a TF-IDF vector. Using an n-gram TF-IDF vector may alleviate the problem to some extent, but it is still hard to capture long-distance or high-level abstract sequential patterns. Moreover, (n-gram) TF-IDF vectors cannot capture syntactic or semantic relationship/similarity between words, paragraphs, and documents. For instance, 'dog', 'cat', and 'move' are treated equally as independent words, though 'dog' and 'cat' are semantically related and they are both pets.

Another notable document vectorization is the *paragraph vector* that learns from a distributed memory model (PV-DM), which is a succinct distributed representation of sentences or paragraphs [28, 11, 1]. PV-DM performs significantly better than the bag-of-words model in many NLP tasks. Moreover, skip-thought vectors [23] that are derived from recurrent encoder-decoder models also show better performance against the bag-of-words model.

1.2 Multilingual document embedding: task, challenge and application

A cross-lingual (or multi-lingual) text embedding/encoding model can project texts from different languages into a common vector space and produces a language-independent representation of the words [24, 10, 7, 5] or sentences/documents [20, 41, 36, 5]. Multilingual data for a specific task or service are scarce. Traditionally, a way of enabling cross-lingual knowledge sharing between resource-rich and resource-scarce languages is to use a multilingual translation model to translate all task-related data of different languages into a single pivot language. Then, a simple classification or information extraction model can use the train/test data from different language sources through this translation. The multilingual text embedding model serves the same purpose as the aforementioned translation model. However, instead of translating various languages into a pivot language, the multilingual text embedding model would 'translate' the texts of various languages into vectors in a common vector space. The multilingual embedding model would be faster than a translator of comparable quality (as it does not require a decoder when producing embedding vectors), and the fixed-size vector it produces would be a better input for comparison, indexing and clustering compared to text feature [20, 7].

In the monolingual situation, the embedding vector will be an enhanced replacement of the traditional TF-IDF feature. In a multilingual situation, the embedding model would also function as an all-to-one translation model that can 'translate' input documents of various languages into one common vector space. Ideally, texts of similar content from different languages should be projected close together (e.g., the English translation and the German translation of the Bible should be projected close together), enabling cross-lingual knowledge sharing.

Unlike translation models whose performance could be directly measured by the BLEU score, the quality of a document embedding model is measured by the performance of its embedding vectors in downstream tasks. As a common practice, the evaluation would use some 'downstream' task (such as cross-lingual topic classification on the news) to test whether the vector representation from the embedding model truly captures the original text's content and whether the knowledge sharing between languages is strong enough. In such a setting, the embedding model is responsible for 'translating' texts of different languages into vectors of the same vector space. A classifier is henceforth used to train and test these vectors after 'translating' texts (of training, testing and validation data) from a particular ‘downstream’ task into vectors using the embedding model.

Suppose such vector representation can indeed capture the various information and characteristics of the original document while at the same time also correctly representing the cross-lingual relationship and cross-lingual content similarity between texts of different languages. We can certainly use this document vector as the only input feature to train a classifier and achieve high accuracy in a cross-lingual setup. Moreover, if we assume the classifier has the same structure and training setting for a particular ‘downstream’ task for a different type of embedding models (e.g., the T-MDE and baseline models), the more capable the embedding model is, the higher accuracy the classifier will be able to obtain on the test set.

For example, in the widely used multilingual document classification task (MLDoc), there are four different types of news articles in eight languages: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets). In evaluation, the system is given 1000 documents in language ‘X’ as training data (consisting of the four types of news mentioned above). The system is required to classify another 5000 documents in language ‘Y’ (where Y is different from X). In order to complete the task, the text embedding model is trained on the task-unrelated Europarl corpus (a commonly available parallel corpus for training translation models). Then, the text embedding would ‘translate’ all MLDoc training/testing data into vectors. If the cross-lingual document embedding model can successfully embed news articles of both language X and Y into vectors of the same vector space, a simple classifier (e.g., SVM, MLP) trained on the embeddings of language ‘X’ (e.g., English) will be able to classify embeddings from language ‘Y’ (e.g., German, French.).

In real-world applications, task-unrelated and unlabeled parallel data (such as the Europarl corpus) are commonly available and most accessible, while annotated data for a particular task and language is very scarce. If a company has a text classification service for its English user base and wants to expand its service to German, then a multilingual embedding model can enable the company to skip the expensive procedure of collecting and annotating the German task-related data by simply projecting German texts of the related task using the embedding model.

1.3 The pre-training-without-fine-tuning setup

The ‘pre-training-without-fine-tuning’ setup is commonly used in training and evaluating text embedding models. Let us consider an embedding model and a classifier as a system. The embedding model (which translates texts of various languages into vectors in the same vector spacer) is the part that is pre-trained only on the commonly available parallel corpus (e.g., the Europarl corpus) regardless of the downstream tasks. The same embedding model is used for different downstream tasks (such as news classification like MLDoc or cross-lingual natural language inference like XNLI). No task-related data (e.g., MLDoc, XNLI) is used to train/fine-tune the embedding models. Only the classifier is trained on task-related data for different downstream tasks and will have different parameters from task to task.

The rationale behind this ‘pre-training-without-fine-tuning’ setup lies in two folds. Firstly, it is inconvenient to adapt or fine-tune an embedding model for each downstream task (especially for new tasks). Moreover, it is often necessary to have a predetermined general-purpose embedding vector that can be referenced and compared across different tasks. For example, fine-tuning the embedding model with MLDoc data can undoubtedly enhance its performance in the MLDoc experiment. However, embedding models fine-tuned for different tasks will produce different embeddings for the same input. It should be noted that this ‘pre-training-without-fine-tuning’ setup is different from BERT, where the embedding model and the classifier are integrated into the same model and fine-tuned together for each different task.

Therefore, the model presented in this thesis and its primary baselines are commonly referred to as the embedding model, while BERT and its variant are commonly referred to as the ‘pre-training-and-fine-tuning’ framework. The difference is that embedding models in this setup can produce a predetermined general-purpose embedding vector that can be referenced and compared across different tasks. In this case, only the classifier is re-trained for any unforeseen new task, while the same embedding model does not need to be fine-tuned. A typical embedding model is guaranteed to produce the same embedding vector for the same text input regardless of tasks.

1.4 Transformer-based multilingual sentence embedding model

Language-Agnostic SEntence Representations (LASER) [43, 4] and XLM-R [26, 8] are two of the current state-of-the-art (SOTA) multilingual sentence/document embedding models. LASER is designed after the neural machine translation (NMT) model. It uses a bidirectional long short-term memory (Bi-LSTM) encoder and a long short-term memory (LSTM) decoder, and the sentence embedding vector is obtained from max-pooling the encoder outputs. XLM-R, on the other hand, is a transformer-based model that employs a token masking and sentence entailment classification framework. Inspired by LASER, BERT, and XLNet [12, 54], this thesis introduces a transformer-based multilingual sentence embedding model, which I call *T-MDE*, and compares its performance with other SOTA document embedding models.

In T-MDE, a bottleneck transformer model is developed for multilingual document embedding. Compared to multilingual BERT [12, 13] (mBERT) and XLM-R, T-MDE adopts the multilingual NMT framework. More importantly, T-MDE has a self-attention-based *information bottleneck* between the encoder and decoder that compiles the entire encoder's information into a single sentence embedding vector. More specifically, T-MDE's decoder can only query the sentence embedding instead of the entire encoder output. I find this *information bottleneck* would help improve the quality of document embedding immensely when task-related fine-tuning is not allowed. T-MDE inherits LASER's strength in dealing with many languages in multilingual situations. As a result, the new transformer-based model performs better and runs faster than the BiLSTM-based baseline owing to the new transformer structure. This is because (1) the absence of recurrent structures in the transformer's encoder renders parallel computations possible, as a successive computational state does not depend on its previous states; (2) the self-attention mechanism in the transformer allows learning from a much broader context than the LASER's BiLSTM encoder as every state can attend to every state in the input sequence.

In this work, a novel loss function has also been developed that combines a specially designed distance constraint loss function for multilingual document embedding with the traditional translation loss. This work also discusses its effect on both LASER and T-MDE architectures. This work would label the T-MDE model trained with this new loss function as *cT-MDE*, whereas the LASER model trained with this new loss function as *cLASER*. The distance constraint loss term significantly raises the performance in multilingual settings. Moreover, it can deal with the bilingual setting, which the traditional multilingual NMT setting is not designed for and, thus, does not give a good performance.

Note that in this thesis, we will sometimes name the 'training' phase of LASER and T-MDE as 'pre-training,' albeit there is no fine-tuning phase for embedding models like LASER and T-MDE.

This naming is to clarify the difference between the training of the embedding model (e.g., LASER, T-MDE) on the task-unrelated parallel corpus (e.g., Europarl corpus) and the training of the simple FFN classifier with task-related data (e.g., MLDoc, XNLI). The 'training' phase for embedding models (e.g., LASER, T-MDE) and the 'pre-training' phase of the currently popular BERT-related models are the same. Moreover, the training of the simple FFN classifier with task-related data can be compared with the 'fine-tuning' phase of BERT. If both the training of the embedding model itself and the training of the classifier are called 'training,' it will generate quite a lot of confusion. Therefore, the 'training' of LASER and TME would sometimes be named 'pre-training' to clarify that it is the training phase of the embedding model, not the training phase of the FFN classifier. Both T-MDE and our LASER implementations are pre-trained only on the publicly available Europarl v7 parallel corpus. Here, the commonly used cross-lingual classification task on RCV1/RCV2 (MLDoc [44]) is taken as the downstream task. All train/test data in MLDoc are converted to document vectors using T-MDE, LASER and XLM-R. Their cross-lingual classification accuracy is examined on a simple (two-layer) FFN classifier. As T-MDE and other embedding model baselines are trained with the same training data under a uniform training setting, a fair comparison can be made between the capabilities of these different model structures.

1.5 Thesis summary and outline

This thesis has been divided into four chapters. In chapter 2, preliminaries and related works are introduced. In the beginning part of the first section, the basic model structure of the recurrent neural network and long short-term memory are described in detail. At the end of the first section, LASER, the primary baseline with Bi-LSTM auto-encoder structure, is introduced. The second section of Chapter 2 introduces the transformer structure used in T-MDE. Moreover, BERT and the pre-training and fine-tuning framework are also described. Finally, XLM-R, another baseline based on the transformer structure, is introduced.

Chapter 3 introduces preliminary research on the neural machine translation model and document embedding. This chapter investigates a cross-lingual document embedding method that improves the current *$\underline{N}$eural machine $\underline{T}$ranslation framework based $\underline{D}$ocument $\underline{V}$ector* (NTDV or simply NV). NV is developed with a self-attention mechanism under the neural machine translation (NMT) framework. A distance constraint is further added to the training objective of NV so that the two embeddings of a parallel document are required to be as close as possible. The new method will be called *constrained NV* (cNV).

cNV requires either the assistance of translator or distance constraint training to achieve better performance and is slower in training and decoding due to the recurrent structure. Therefore, the main theme of this thesis–the T-MDE (and cT-MDE) model, which draws inspiration from cNV and LASER, is introduced in Chapter 4. The GRUs encoder in cNV is replaced by the attention-based transformer structure with a novel information bottleneck design. The new model structure can learn sequential patterns in longer texts. It is faster both in training and embedding generation. Moreover, cT-MDE adopts a more carefully designed distance constraint loss term than cNV, enhancing performance and numerical stability. Finally, the T-MDE model adopts LASER's multilingual translation training framework with a shared encoder and decoder. This design proves to be simpler and more efficient than NV's complicated framework with multiple encoders and decoders.

Chapter 5 concludes the thesis and offers insights into future research multilingual document embedding.

CHAPTER 2

PRELIMINARIES AND RELATED WORKS

2.1 Long-short term memory network and text embedding

2.1.1 Recurrent neural network

Sequence to sequence prediction has always been an important task in NLP systems. For instance, POS-tagging, name entity recognition, and language modeling are all typical sequence to sequence prediction tasks. However, the feed-forward neural network always has difficulty when dealing with sequence to sequence prediction. Under such circumstances, the invention of the recurrent neural network (RNN) is a revolutionary improvement for the sequence to sequence prediction tasks.

As indicated by the name, RNN is distinguished by its recurrent connection in its network structure. When a traditional feed-forward neural network predicts a temporal sequence, it assumes that previous hidden states and current output are independent of each other. On the other hand, the recurrent connection in RNN assumes the current output depends on the previous hidden states within the sequence. Therefore, the recurrent connection between hidden states of different time frames serves as 'memory' between prior events and current events. In particular, Let x_t, y_t, h_t be the input, output and hidden state at time frame t. A most commonly used single layer RNN (Elman networks) would have:

$$h_t = \sigma_h(\mathbf{W_h} x_t + \mathbf{U_h} h_{t-1} + b_h\ ), \tag{2.1}$$

where σ_h is the activation function at hidden layer (usually use sigmoid or tangent function), $\mathbf{W_h}$ is the projection matrix from the input layer to hidden layer, $\mathbf{U_h}$ is the projection matrix from the previous state to the current state. b_h is the bias at the hidden layer. Then the current moment output is:

$$y_t = \sigma_y(\mathbf{W_y} h_t + b_y\ ), \tag{2.2}$$

where σ_y is the activation function (usually use Sigmoid or Tangent function) at the output layer, $\mathbf{W_y}$ is the projection matrix between hidden states and output, and b_y is the bias term at the output layer. The above equations indicate that the RNN model can be infinitely unrolled along the time

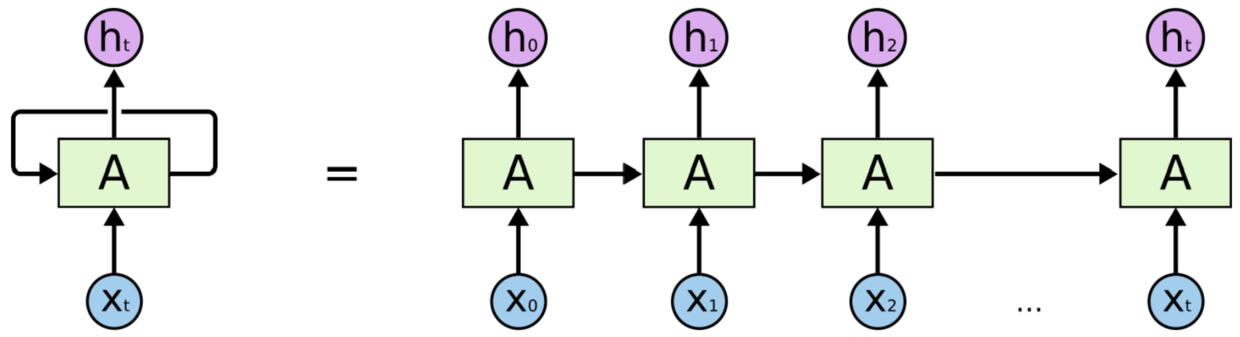


Figure 2.1: Model architecture of RNN[38].

axis. Thus the 'memory' accumulated from $\mathbf{h_0}$ can have an impact to $\mathbf{h_{1000}}$ even after 1000 time steps.

2.1.2 Long short-term memory network

The most common problem over the traditional RNN is the gradient explosion and vanishing problem. From Equation 2.1 we can infer that if we unroll the RNN network along the time axis, the hidden states will be multiplied repeatedly by the same $\mathbf{U_h}$. Thus, its numerical value will become very small or very big after many steps. This instability will, in effect, make RNN impractical in processing very long sequences. Therefore, algorithm modifications tailored for RNN's training/updating, such as gradient clipping, are developed to overcome the problem. However, the overall 'memory' capacity of RNN is still quite limited as repeated multiplication of $\mathbf{U_h}$ is the only mechanism that emulates 'memory' of the past events. Under such circumstances, the long short-term memory network is developed to overcome the shortcoming of traditional RNN.

The key to LSTMs is the cell state. In the RNN model, historical information is represented by past hidden states unrolled along the time axis. The historical information in an LSTM model is represented by the past cell states unrolled along the 'time' axis. In the LSTM model, the transfer of historical information from the left cell state (previous event) to the right cell state (future event) is controlled by various gates:'forget gate', 'input gate' and 'output gate'. These gates would enable the LSTM to remove or add information to the cell state $\mathbf{c_t}$ according to the current circumstances and carefully regulate historical information passed down the time axis.

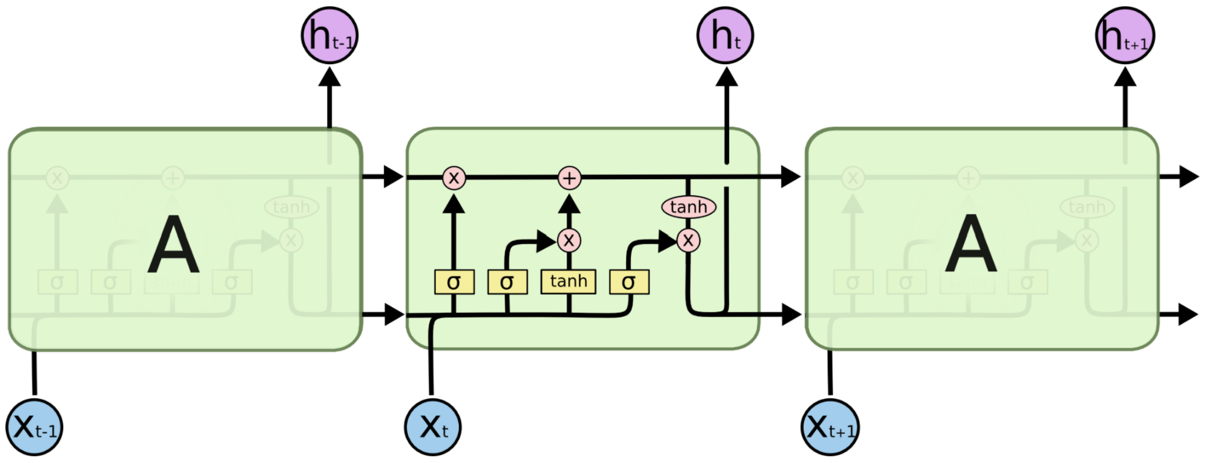


Figure 2.2: Model architecture of LSTM[38].

In particular, in the LSTM cell, the hidden output $\mathbf{h_t}$ is governed by:

$$\mathbf{f_t} = \sigma(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + \mathbf{b_f}) \tag{2.3}$$

$$\mathbf{i_t} = \sigma(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{b_i}) \tag{2.4}$$

$$\mathbf{o_t} = \sigma(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{b_o}) \tag{2.5}$$

$$\tilde{\mathbf{c}}_\mathbf{t} = \tanh(\mathbf{W_c x_t} + \mathbf{U_c h_{t-1}} + \mathbf{b_c}) \tag{2.6}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \tilde{\mathbf{c}}_\mathbf{t}) \tag{2.7}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{c_t})) \tag{2.8}$$

where σ is the sigmoid function, $\mathbf{x_t}$ is the current input, $\mathbf{h_{t-1}}$ is previous hidden state and $\mathbf{f_t}$ is the forget gate's activation vector that control what information to forget. $\mathbf{b_f}$, $\mathbf{W_f}$, $\mathbf{U_f}$ are bias and projection matrices of the forget gate layer. $\mathbf{i_t}$ is the input gate's activation vector and $\mathbf{b_i}$, $\mathbf{W_i}$, $\mathbf{U_i}$ are bias and projection matrices of the input gate layer. $\mathbf{o_t}$ is the output gate's activation vector and $\mathbf{b_o}$, $\mathbf{W_o}$, $\mathbf{U_o}$ are bias and projection matrices of the output gate layer. $\tilde{\mathbf{c}}_\mathbf{t}$ is the cell input modulation vector and $\mathbf{b_c}$, $\mathbf{W_c}$, $\mathbf{U_c}$ are bias and projection matrices of the cell update layer. $\odot$ is the Hadamard product.

More specifically, the forget gate layer in the LSTM model decides what historical information the model should 'forget' (i.e., not being passed along the time axis). The forget gate layer uses the previous hidden state $\mathbf{h_{t-1}}$ and current input $\mathbf{x_t}$ to decide the forget gate's activation vector $\mathbf{f_t}$ (equation 2.3). If $\mathbf{f_t}$ has a numerical value of zero on one dimension, then the previous cell state $\mathbf{c_{t-1}}$ on that dimension will be forgotten. If $\mathbf{f_t}$ has a numerical value of one on one dimension, then

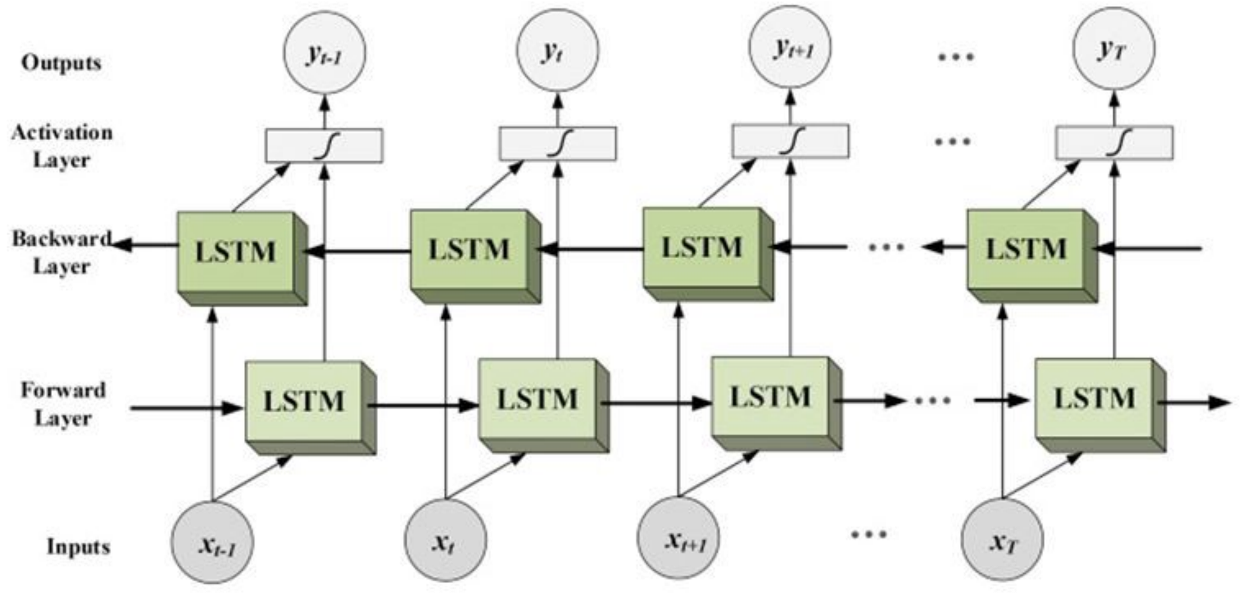


Figure 2.3: Model architecture of bidirectional LSTM[37].

the previous cell state $\mathbf{c_{t-1}}$ on that dimension will be remembered by the cell on the right without any loss (equation 2.7). This mechanism is more powerful than the recurrent projection matrix $\mathbf{U_h}$ in the RNN model, as the forget gate in the LSTM model can decide to copy all the information from the past and eliminate the gradient vanishing and gradient explosion problem.

The input gate layer controls which new information will be stored in the cell state. The input gate layer uses the previous hidden state $\mathbf{h_{t-1}}$ and current input $\mathbf{x_t}$ to decide the input gate's activation vector $\mathbf{i_t}$ (equation 2.4). The cell input modulation vector $\tilde{\mathbf{c}}_\mathbf{t}$ is used to modulate the information flowing from the input gate to the cell state by adding non-linearity to the information and making the information zero-mean (equation 2.7). Zero-mean input has faster convergence and can reduce the learning time. The current cell state is determined by information from the previous cell state regulated by the forget gate and the information from the current input regulated by the $\tilde{\mathbf{c}}_\mathbf{t}$ (please see equation 2.7).

The output gate layer controls information flowing from the cell state to the hidden state (the output of the LSTM cell). The cell state is put through the hyperbolic tangent function (in order to produce values of $(-1, 1)$) and multiplies it by the output gate's activation vector $\mathbf{o_t}$ (equation 2.8).

While in many occasions, future events (states on the right side of the current time frame) would also help determine the output of a given sequence, unidirectional recurrent neural networks only

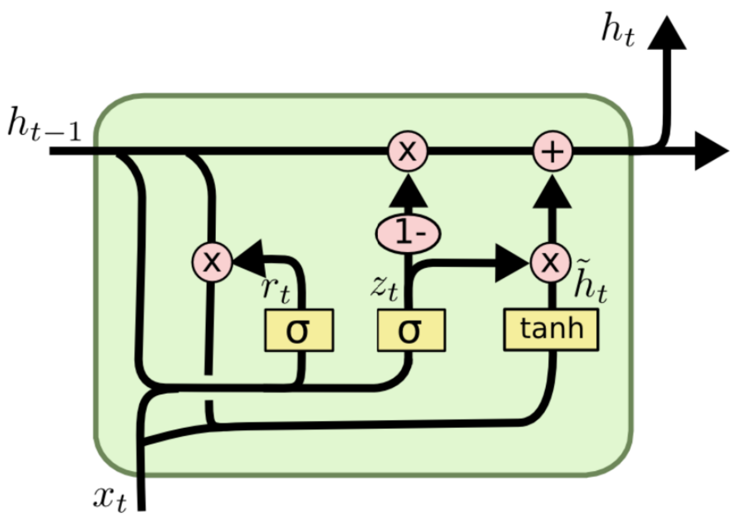


Figure 2.4: Model architecture of GRU[38].

account for previous events (states on the left side of the current time frame). Therefore, bidirectional LSTM(Bi-LSTM) is developed to solve this problem (see figure 2.3). In a BI-LSTM, two LSTM networks are put together. The input sequence is fed in with a left-to-right order for one network and a reverse right-to-left order for another. The outputs of the two networks are usually concatenated at each time step. Thus, combining these two reversely ordered LSTM networks forms a Bi-LSTM layer, whose outputs are conditioned on both 'past' and 'future' information.

2.1.3 Gated recurrent units

Although the LSTM model has significantly improved performance in many applications, its complicated structure would significantly increase the number of model parameters and decrease the training speed. Therefore, much effort has been put into simplifying the LSTM model. The gated recurrent units (GRU) can be seen as a simplified variant of LSTM. The GRU (also called GRUs) has a simplified cell structure, faster training speed, and equivalent performance to the traditional LSTM model.

In the GRU network, the forget gate, input gate, and output gate are simplified into an update gate and a reset gate. Instead of using an intermediate cell state $\mathbf{c_t}$ as 'memory' passes along the time axis like LSTM, GRU use hidden state $\mathbf{h_t}$ directly as 'memory'. In particular, in a GRU cell, the hidden output h_t is governed by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{2.9}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2.10}$$

$$\tilde{h_t} = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.11}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{2.12}$$

where z_t is the update gate vector; W_z, U_z, b_z are the projection matrices and bias of the update gate; r_t is the rest gate vector; W_r, U_r, b_r are the projection matrices and bias of the reset gate; $\tilde{h_t}$ is the candidate activation vector.

The update gate vector z_t in GRU helps to determine how much the current hidden state h_t is affected by the candidate activation vector $\tilde{h_t}$ and the previous hidden state h_{t-1}. If z_t has a numerical value of zero on one particular dimension, then the current hidden state h_t on that dimension will be a direct copy of the previous hidden state h_{t-1}. If z_t has a numerical value of one on a particular dimension, then the current hidden state h_t on that dimension will be entirely coming from the candidate activation vector $\tilde{h_t}$. The reset gate vector r_t in GRU helps to determine how much of the past information to remember. Sigmoid is used to produce both z_t and r_t with numerical value between $(0, 1)$.

GRU has proven to be a simple, fast, and reliable replacement for traditional LSTM in most NLP applications such as machine translation, document classification and sequential tagging. It can achieve LSTM's performance with less training time and the number of parameters [15]. Therefore, this thesis will put GRU and LSTM methods in the same category. When an LSTM model can achieve something or has some characteristics in document embeddings, its GRU variant will be in a very similar situation. For example, when comparing LSTM-based document embedding model with a transformer-based model on performance and speed, the same conclusion can be drawn on the GRU variant of that LSTM model in most cases.

Note that the training/testing speedup in GRU compared with its LSTM variant is linear. A GRU-based encoder still has the recurrent mechanism and the conditional dependency between past and current events. Processing time on a GPU device for an input sequence of length N with a GRU encoder is still O(N). In comparison, processing time on a GPU device for the same input sequence with a transformer-based encoder is O(1). We will explain it in more detail in the transformer-related section.

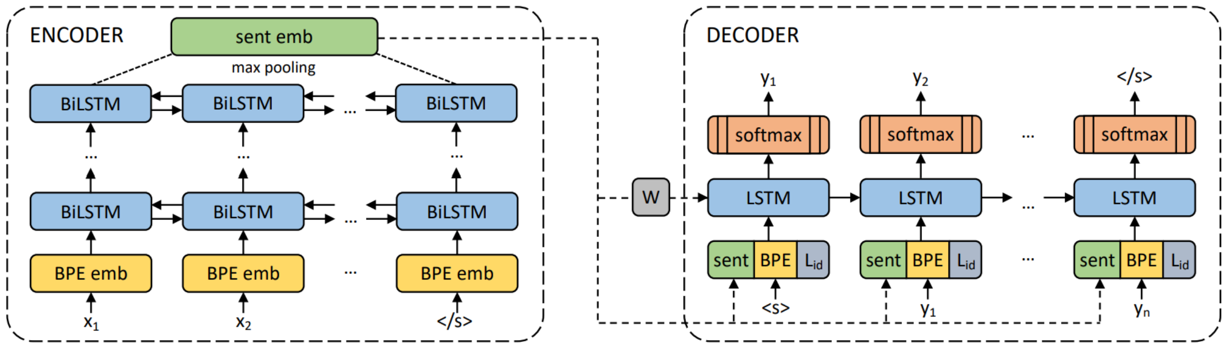


Figure 2.5: Architecture of LASER for multilingual sentence embeddings[4]

2.1.4 LASER, bidirectional LSTM auto-encoder for document embedding

LSTM model is very suitable for producing text embedding. Firstly, it can capture comparatively more distant patterns in a document that are not limited by the size of the input context window. Thus, the model parameters of an LSTM-LM can encapsulate the different grammars and styles in its training documents. Secondly, the hidden layer(s) of an LSTM provides a distributed representation of the input words in a continuous space to capture the semantic and syntactic relationship among words.

LASER is the current state-of-the-art multilingual sentence embedding model [43, 4]; it is designed after a typical LSTM-based multilingual neural machine translation (NMT) model. It uses a bidirectional-LSTM (BiLSTM) encoder and an LSTM decoder, and the sentence embedding vector is obtained from max-pooling the encoder outputs. This sentence embedding is used to initialize the first-time step of the decoder LSTM. For every decoder side input, the inputting vector is the concatenation of sentence embedding, ground truth at the previous time frame, and language id (indicating the output language)

For example, if a five languages subset (English, German, Spanish, French, Italian) of the Europarl corpus is used for LASER training, then the input text can be drawn from any language in the training set while the target language is one of the two pivot languages, which are English and Spanish in this example. The training language pair will be de-en,de-es,en-es,es-en,fr-en,fr-es,it-en,it-es [1]. For example, in the 'de-es' training pass, the German sentence will be processed through the five layers of the shared Bi-LSTM encoder. It will be further translated into the corresponding

[1]de: German, en: English, es: Spanish, Fr: French, it: Italian

English output through the attention mechanism and the single-layer decoder. On the decoder side, the decoder input (the English ground truth of the previous token) **Y** would first concatenate with a language embedding (E_l) (indicating whether the current target output is English or Spanish language). The **Y** would also be concatenated with the sentence embedding vector obtained from max-pooling the encoder outputs. Thus, the current English output depends on both the sentence embedding and the previous ground truth.

In LASER, the encoder would not have language-related information as LASER uses a language-independent dictionary, and the encoder is fully shared among all source languages. As a result, the sentence embedding of similar content (e.g., French, German, or Italian translation of similar content) would get closer to each other as they are encouraged to produce the similar English or Spanish translation target at the decoder. The pivot language effectively serves as an anchor for text embedding of different languages to establish relationships if they have similar contents. This trait (uniform dictionary, fully shared encoder, and pivot languages) makes LASER exceptionally capable when dealing with a large number of languages in multilingual situations [43, 4]. The training mentioned above does not require a completely parallel reference for a sentence in every language. There will be a similarity between English target outputs in a highly abstract manner even if the fr-en and de-en datasets do not overlap.

As a document/sentence embedding model, LASER takes a pre-training-without-fine-tuning framework. LASER$_{2018}$ is the LASER model [44] trained with the Europarl corpus. The latest published LASER model (LASER_5$_{2019}$) [4] was trained on a large 93-language corpus. It combines Europarl, United Nations, OpenSubtitles2018, Global Voices, Tanzil, and Tatoeba corpora, consisting of 93 languages. Training the LASER model with the 93-language corpus would take five days using 16 V100 GPUs.

After the training on the task-unrelated parallel corpus, the effectiveness of the LASER model is tested on several cross-lingual tasks that require the multilingual embedding model to establish knowledge sharing between different languages. The embedding quality of LASER$_{2018}$ is tested by the cross-lingual document classification (MLDoc) task. The effectiveness of LASER_5$_{2019}$ is tested on five tasks: the cross-lingual document classification (MLDoc), WikiMatrix, Bitext mining (BUCC), Cross-lingual NLI (XNLI), and Multilingual similarity search [44, 4]. The MLDoc, XNLI and BUCC tasks are commonly cited downstream tasks in cross-lingual document embedding evaluation. The accuracy of a simple FFN classifier on these tasks reflects how well document embeddings of various languages are placed in a common vector space. In this thesis, MLDoc and XNLI are used to evaluate the LASER baselines and the T-MDE models, considering the availability of the data.

2.2 Transformer and text embedding

2.2.1 Transformer

In recent years, the attention-based transformer model has become very popular for sequence-to-sequence NLP tasks (e.g., translation, language modeling, POS-tagging, dialog generation). Compared to the traditional LSTM-based architecture, the transformer structure enjoys two significant advantages. Firstly, the absence of recurrent structure makes computational parallelism possible in the transformer encoder, as the successive computational state does not depend on the previous states. Secondly, the self-attention mechanism allows learning from a much broader context than LSTM as every state can attend to every state in the input sequence. In most sequence-to-sequence related tasks such as machine translation sentence parsing, the transformer-based structures would achieve comparable or superior performance than LSTM/GRU-based models while also enjoying a significantly faster training speed.

In the original transformer paper [50] a multi-head attention mechanism $\mathcal{F}_{MH}$ is adopted to replace the recurrent structure in the LSTM model to represent the memory and connection between past and future events:

$$\mathbf{U}_i = \mathrm{softmax}\left(\frac{(\mathbf{QW}_i^Q)(\mathbf{KW}_i^K)^T}{\sqrt{d_h}}\right)(\mathbf{VW}_i^V) \tag{2.13}$$

$$\mathcal{F}_{MH}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathbf{U}_1,\dots,\mathbf{U}_r)\mathbf{W}^O \tag{2.14}$$

where U_i is the i-th attention head; r is the number of attention heads; d_h is the size of the encoder's hidden layers; Q, K, V indicate the role of *query, key* and *value*; **Q**, **K**, **V** are matrices constructed by sequentially arranging the vectors of *query*, *key* and *value*; $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ are projection matrices associated with the i-th attention head; $\mathbf{W}^O$ is the projection matrix for the concatenated attentions from various heads. $\mathbf{U}_i$ should be a dimension of $F \times T$, where T is the length of the sequence F is the feature dimension (i.e., the width of the transformer). Concat is the concatenation of matrices along the feature dimension.

In the transformer encoder, the queries, keys and values are the same and are the outputs of the previous layers. The outputs from every position will attend on every other position and itself (and therefore named self-attention). Between the encoder and decoder, the keys and values come from the encoder outputs, while the queries come from the previous decoder layer. Thus the decoder would select the most relevant information from the encoder on the condition of the output from the previous decoder layer. The first decoder layer input is either the previous predictions or right-shifted ground truth. The transformer decoder also uses the self-attention mechanism as in the encoder. Moreover, a mask is added to the future positions on the decoder input to prevent future information from flowing in and make the decoder auto-regressive.

As can be inferred from the above equations, the self-attention mechanism can establish the connection between past, present, and future events and produce sequential outputs without any recurrent structure. Thus, a transformer encoder layer can combine information from t_0 to t_N and generate the output sequence simultaneously. Therefore in a device where computation parallelism is sufficiently supported, when the text sequence has a length of N, the training/testing speed of an LSTM encoder layer is O(N), while the training/testing speed of a transformer encoder layer is O(1).

On a GPU device, a transformer layer can generate the entire output sequence simultaneously, while an LSTM layer can only generate a single output at a time due to the strict dependency between the current event and the past event. In the current GPU device, a large tensor operation in which every element can be computed simultaneously will be solved much faster than a sequence of smaller tensor operations that have to be computed in a strictly sequential manner.

2.2.2 BERT, pre-training and fine-tuning framework of deep bidirectional transformers for language understanding

The state-of-the-art pre-training and fine-tuning models like BERT and XLNet based on the transformer encoder[12, 54] also gain much popularity as it is adaptable to a variety of language understanding tasks. The Bidirectional Encoder Representations from Transformers (BERT) is invented to perform various language understanding tasks. BERT adopts a simple but effective pre-training and fine-tuning framework. Thanks to the self-attention mechanism in the transformer encoder, the BERT encoder can be jointly conditioned on the entire context in all layers and produce deep bidirectional representations from the inputting text sequence.

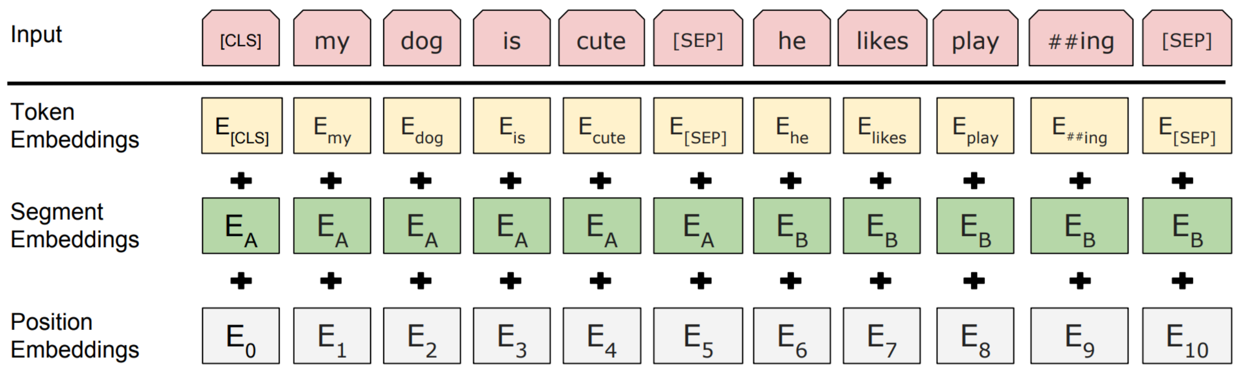

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.6: The BERT input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings[12].

BERT adopts essentially the same transformer encoder used in the transformer NMT model. Unlike the transformer NMT model, which takes one sentence or one text sequence as input, BERT would randomly sample two sentences/sequences from a broader context and concatenate the two sentences into an input sequence. It should be noted that the two sentences sampled can be either consecutive or completely unrelated. Moreover, left-padding is eliminated, and a special 'CLS_TAG' token is inserted in the first position of the concatenated input; a special token of separation is also inserted between the two concatenated sentences. Since the 'CLS_TAG' would always appear as the first input token for every sentence pair, it makes the first position of the encoder a unique position.

The BERT model would handle two tasks in the pre-training phase. Firstly, BERT will randomly mask the input text and the BERT encoder is trained to predict the masked tokens/words given the surrounding context. In this situation, the BERT encoder effectively becomes a masked language model attempting to fill in the blank in the sentence. Moreover, BERT also deals with a next sentence prediction (NSP) task during pre-training. The first position encoder output will be taken into an FFN layer to produce a binary NSP label. This binary NSP label would infer the relationship between the two input sentences concatenated together whether they are consecutive or unrelated. In the transformer encoder, the 'CLS_TAG' on the first position encoder input would attend not only on itself but also on inputs from every other position. The first position output on the final encoder layer is taken into the FFN layer to perform the NSP or any subsequent classification task. This setup will encourage the BERT encoder to gradually concentrate inter-sentence or contextual information through self-attention layers into the first position encoder output.

In summary, the language modeling task in the pre-training phase would enable the BERT

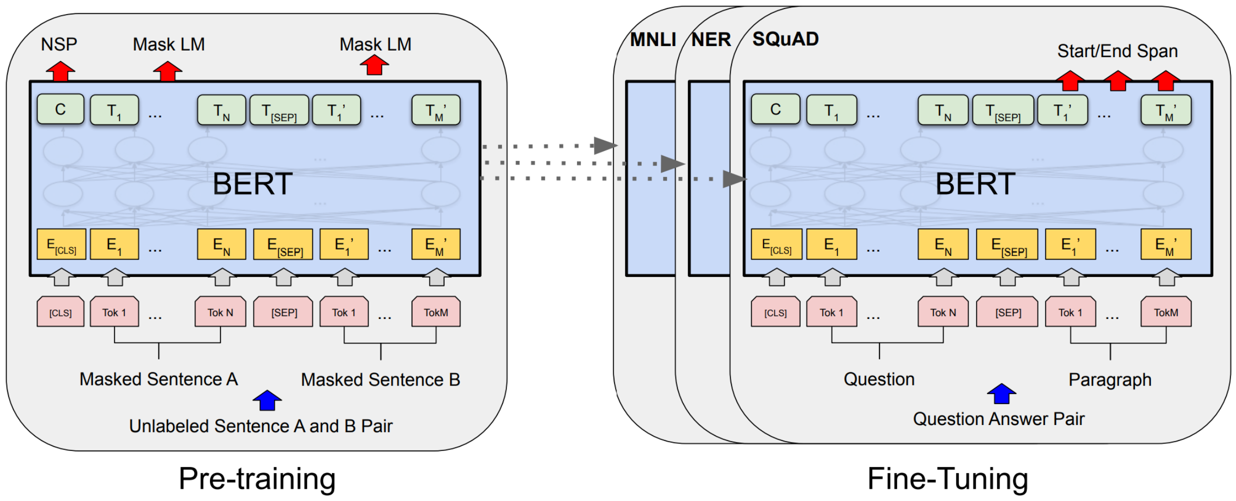


Figure 2.7: Overall pre-training and fine-tuning procedures for BERT[12]

encoder to learn syntactic and semantic information within a sentence. In contrast, the next sentence prediction task would enable the BERT encoder to learn inter-sentence relationships and concentrate such information on the BERT encoder's first position output. This first position encoder output as a vector can also be used for other downstream language understanding tasks in the fine-tuning phase. The benefit of such a pre-training setup is that the most commonly available raw texts crawled from the internet can be utilized to train the masked language model and imprint basic semantic or syntactic knowledge (such as grammar and semantic relationship between texts) in the BERT encoder. This is especially useful because the task-related data for a particular downstream task is often quite small. The small amount of task-related data would not be sufficient to train the large transformer encoder used in BERT. Therefore, BERT would utilize a large amount of unlabeled text to pre-train model parameters in the encoder so that the BERT encoder can already produce a deep representation of texts for various tasks without even seeing the task-related data.

During the fine-tuning stage, task-specific data can be used to optimize the model towards a particular task. Like the NSP task, the BERT encoder can be concatenated to another FFN layer suitable for a classification task. For example, if the first position encoder output is put into an FFN layer and the entire model is fine-tuned with News classification data, the abstracted feature and helpful information for the News classification task will be concentrated on the first encoder output. A decoder can also be attached to the pre-trained BERT encoder if the task at hand is text generation or machine translation. The combined model is fine-tuned on the task-related data.

2.2.3 XLM-RoBERTa, unsupervised cross-lingual representation learning at scale

XLM-RoBERTa (XLM-R) adopts a similar structure as mBERT (a multilingual adaptation of BERT); it also makes several important improvements. XLM-R supports more languages (100 languages) and is trained with two terabytes of publicly available CommonCrawl data [8]. With the scaling of supporting language, each language can have unbalanced performance. A poorly trained model will have much inferior performance in low-resource languages. XLM-R is tuned to offset this problem by upsampling low-resource languages during training and vocabulary construction. In order to achieve the state-of-the-art performance in 100 languages, XLM-R also adopts a larger shared vocabulary and has 550 million parameters to boost model capacity.

XLM-R has two objectives in the pre-training. The masked language modeling (MLM) objective is the same as the BERT, in which the model will predict a masked token based on its surrounding context. The translation language modeling (TLM) objective in XLM-R and mBERT is different from the monolingual BERT. In TLM, XLM-R will take a pair of parallel sentences (e.g., English and French translation). The model will predict the masked English word by referring to both its English context and the French translation in TLM. The prediction is made by attending to both the English and French sentences, thus aligning the English and French representations and enabling cross-lingual knowledge sharing.

The pre-trained XLM-R model is fine-tuned and evaluated on various downstream tasks such as cross-lingual natural language inference (XNLI), named entity recognition (NER) and question and answering (QA). XLM-R achieves the state-of-the-art performance in both cross-lingual and monolingual situations.

In this thesis, XLM-R is one of our primary baselines due to its superior performance than mBERT and other transformer-related multilingual models. XLM-R is a pre-training fine-tuning model; it can also produce text embedding without fine-tuning. Therefore, the pre-trained XLM-R is downloaded and used as a document embedding model and is compared with LASER and our T-MDE.

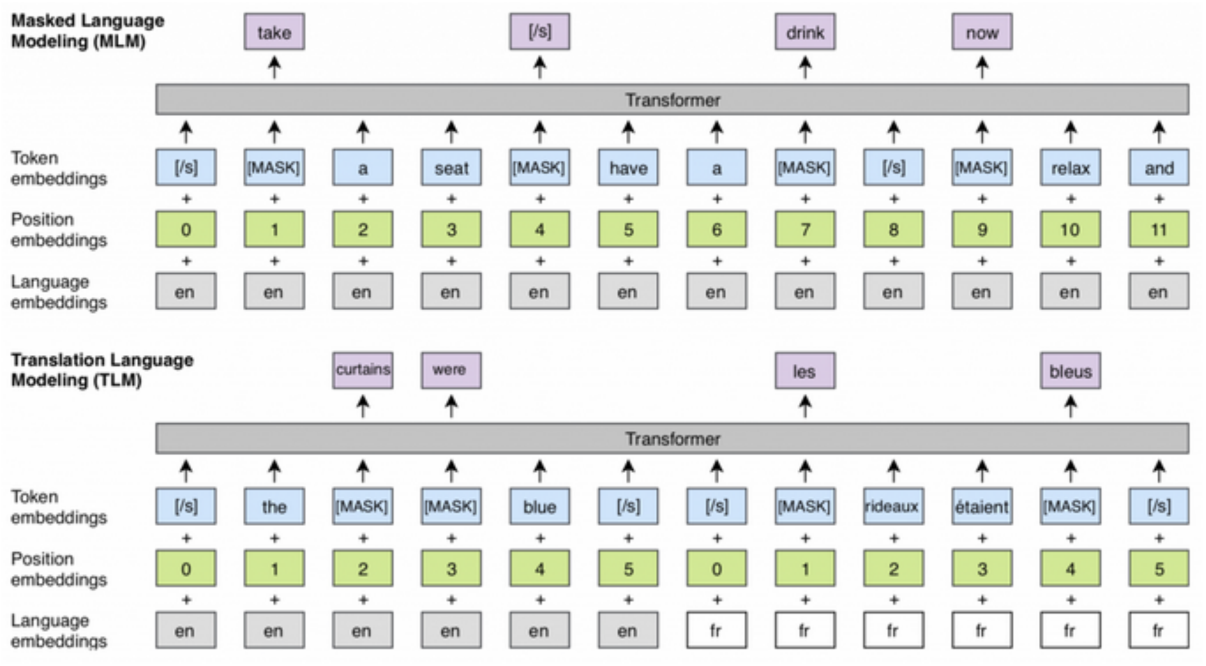


Figure 2.8: The MLM and TML objectives in XLM-R [4]

2.3 Summary

This section first introduces the LSTM model and the baseline model LASER built on LSTM. Then, the transformer model, which has improved performance and training speed over LSTM, is introduced. The currently popular pre-training-fine-tuning model, BERT, built on a transformer encoder, is also introduced in the following section. Finally, another baseline model, XLM-R, further improves BERT towards multilingual situations.

It should be noted that in this thesis, we do not adopt the task-unrelated pre-training and task-specific fine-tuning frameworks like BERT or XLNet [12, 54]. Although the pre-training and fine-tuning framework may produce better performance when its encoder and classifier are optimized jointly in a single network, it is much easier to compare task-agnostic pre-trained embedding vectors across tasks. Moreover, the pre-training of the embedding model can be done offline. Afterward, such a pre-trained embedding model can be plugged swiftly into many different online applications, such as comparison, clustering, or indexing, with a very lightweight online classifier.

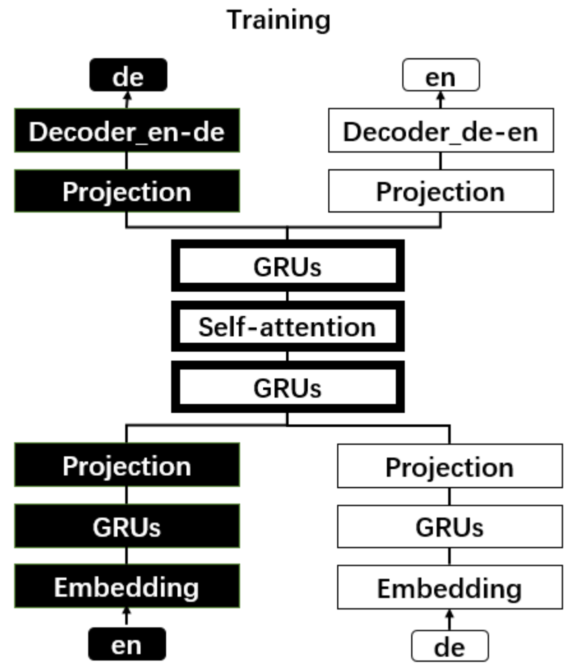
CHAPTER 3

NMT-BASED CROSS-LINGUAL DOCUMENT EMBEDDINGS

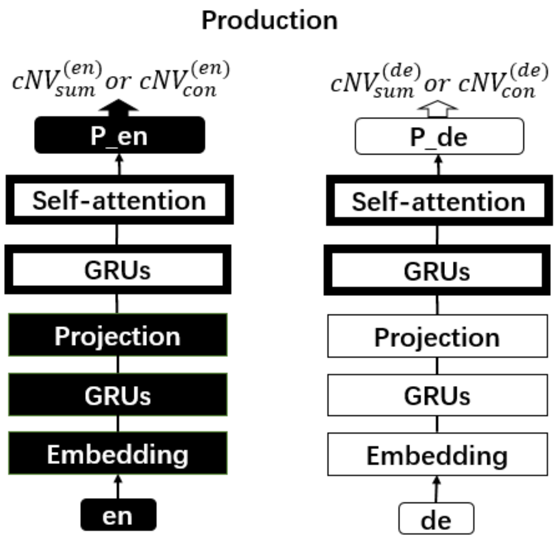
3.1 Introduction

Cross-lingual embedding of texts from different languages to a unified space will enable comparison and knowledge-sharing between languages [20, 7, 36, 5, 17, 19, 3, 44, 46]. With cross-lingual embedding, data in resource-rich languages can be used to help understand inputs from resource-scarce languages. Prior works can be categorized into cross-lingual word embedding [24, 10, 7, 5] and cross-lingual text (document/sentence) embedding [20, 41, 36, 5]. This chapter first introduces a neural machine translation (NMT) based multilingual document/sentence vectorization model (NTDV). A distance constraint loss to minimize the distance between two embedding vectors produced from each parallel document is further introduced to NTDV training. The NTDV model with distance constraint training is designated as cNV. The basic NTDV model without distance constraint training is designated as NV. The NV model cannot produce competent multi-lingual document embedding and would only serve as a reference point to examine the effectiveness of the distance constraint training.

The central theme of the thesis is the transformer-based document embedding (T-MDE). T-MDE uses a shared encoder and decoder structure. In this chapter, NTDV provides an alternative perspective on how a framework based on independent encoders and decoders would work. In combination with the next chapter on T-MDE, this chapter would provide the reader an opportunity to examine the advantage and disadvantages of shared and independent structure in multilingual document embedding models. This chapter is edited and converted from my previous paper: NMT-based Cross-lingual Document Embeddings[32].



(a) In training, English inputs pass through the English encoder and decoder (NMT$_{\mathtt{en}\rightarrow\mathtt{de}}$), while German inputs pass through the German encoder and decoder (NMT$_{\mathtt{de}\rightarrow\mathtt{en}}$.)



(b) In production, the encoder part of the model is used. The English encoder is used to embed English text; the German encoder is used to embed German text.

Figure 3.1: NTDV model in training and production modes.

3.2 Model architecture

3.2.1 The attention-based NMT model

NTDV model is built on the NMT framework, which consists of an encoder-decoder structure with an attention mechanism [45]. NTDV shares many similarities with a bilingual NMT model. It has two NMT models ($\text{NMT}_{a\to b}$ and $\text{NMT}_{b\to a}$) in opposite directions, where $a \to b$ indicates the translation from source language a to target language b. Shared layers are then inserted between the encoders and decoders.

Fig. 3.1(a) shows the NTDV model in the training mode with a being English (en) and b being German (de). $\text{NMT}_{a\to b}$ and $\text{NMT}_{b\to a}$ are first trained independently. Then their parameters are copied into the NTDV model and are fixed in subsequent model training, in which only the parameters of the shared layers are updated. Given a pair of English and German sentences, the English sentence will be processed through the encoder of $\text{NMT}_{en\to de}$, the shared layers, and its decoder resulting in the translation loss of $l_{mt}^{en\to de}$. Similarly, the German sentence in the pair will be processed through the encoder of $\text{NMT}_{de\to en}$, the same shared layers, and its decoder resulting in the translation loss of $l_{mt}^{de\to en}$.

Previous researches show that the information from the encoder helps decide the content of the translated text — the *adequacy* [49, 16]. The NTDV model uses a multi-head self-attentive mechanism to summarize the adequacy information, taking advantage of the information extraction ability of the self-attention layer. This self-attention layer focuses only on the overall sequential pattern, in contrast to the self-attention mechanism in the transformer network that puts more weight on the time-sensitive information [50]. It summarizes the input sequence to a fixed-length matrix (which becomes the document representation after some post-processing). Let z_i be the hidden state of the shared GRU layer. The context vector $\mathbf{p}_i$ produced by the i-th attention head is given by:

$$\mathbf{U}_i = \mathrm{softmax}\left(\frac{(\mathbf{QW}_i^Q)(\mathbf{KW}_i^K)^T}{\sqrt{d_h}}\right)(\mathbf{VW}_i^V) \tag{3.1}$$

$$\mathbf{p}_i = \mathrm{Mean}_t(\mathbf{U}_i) \tag{3.2}$$

where U_i is the i-th attention head; d_h is the size of the encoder's hidden layers; Q, K, V indicate the role of *query, key* and *value*; **Q**, **K**, **V** are matrices constructed by sequentially arranging the vectors of *query*, *key* and *value*; $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ are projection matrices associated with the i-th attention head; $\mathbf{U}_i$ has a dimension of $F \times T$, where T is the length of the sequence F is the feature dimension (i.e., the width of the transformer). $Mean_t$ is a mean-pooling function along the time axis (dimension T).

Fig. 3.1(b) shows the production of various cNV vectors. NTDV uses independent encoders and decoders for different languages; English documents should be put into the English encoder and German documents should be put into the German encoder. Let there be r attention heads, and $\mathbf{P}^{(y)}$ be the context matrix $\mathbf{P}$ from language y's input. The i-th column vector represents the context vector from the i-th head. Different document/sentence vectors can be extracted by summing or concatenating the column vectors in $\mathbf{P}^{(y)}$ as follows:

$$\text{cNV}_{sum}^{(y)} = \sum_{i=1}^{r} \mathbf{p}_i^{(y)} \tag{3.3}$$

$$\text{cNV}_{con}^{(y)} = concat(\mathbf{p}_1^{(y)}, \mathbf{p}_2^{(y)}, \ldots, \mathbf{p}_r^{(y)}) \tag{3.4}$$

$$\tag{3.5}$$

where $\mathbf{p}_i^{(y)}$ is the i-th column of $\mathbf{P}^{(y)}$ and $concat()$ is vector concatenation. Note that the above cNV equations also apply to NV, since they use the same method except for distance constraint training.

3.2.2 Training with a distance constraint

In order to make cross-lingual classification more accurate, $\text{cNV}_{con}^{(en)}$ and $\text{cNV}_{sum}^{(en)}$ should be as close as possible to $\text{cNV}_{con}^{(de)}$ and $\text{cNV}_{sum}^{(de)}$ if they are produced from a pair of parallel text (same text of different translations). To do that, a distance cost is added to the training cost function. The idea is to minimize the distance between the embeddings of related sentence pairs $\{\mathbf{P}^{(a)}, \mathbf{P}^{(b)}\}$ while maximizing the distance between the embeddings of unrelated sentence pairs $\{\mathbf{P}^{(a)}, \mathbf{P}^{(b_j)}\}$, where $\mathbf{P}^{(b_j)}$ is the context matrix obtained from a randomly sampled jth sentence [20]. The distance cost with one negative sample is:

$$l_{d_j} = \max(0, \alpha + \alpha \frac{k_{d_j}}{2 * v_{norm} + 1}) \,, \tag{3.6}$$

where

$$k_{d_j} = |\mathbf{P}^{(a)} - \mathbf{P}^{(b)}|^2 - |\mathbf{P}^{(a)} - \mathbf{P}^{(b_j)}|^2 \,, \tag{3.7}$$

α is the margin and $|\cdot|$ is the Frobenius norm of a matrix; ν_{norm} is the average Frobenius norm of vectors in the training batch; '+1' is to prevent division by zero. Finally, the total cost of the model is the sum of the distance costs from N_s samplings plus the translation cost:

$$l_{com} = \beta \sum_{j=1}^{N_s} l_{d_j} + (1-\beta)(l_{mt}^{a\to b} + l_{mt}^{b\to a}) \,, \tag{3.8}$$

where $l_{mt}^{a\to b}$ and $l_{mt}^{b\to a}$ are the cross-entropy losses of the outputs from $\text{NMT}_{a\to b}$ and $\text{NMT}_{b\to a}$, respectively. $0 \leqslant \beta \leqslant 1$ is a weight to balance the translation cost and the distance cost.

This novel cost function ensures that the produced document vectors contain ample semantic information and are, at the same time, very similar for parallel documents in a language pair. The cNV would enjoy much higher accuracy than NV in the cross-lingual document classification task thanks to the distance constraint training.

3.3 Experiments

Table 3.1: *The setting of various vectors in the cNV model.*

Feature	Dimension	r	d_h
$\text{cNV}_{sum}^{(y)}$	1024	4	1024
$\text{NV}_{sum}^{(y)}$	1024	4	1024
$\text{cNV}_{con}^{(y)}$	4096	4	1024
$\text{NV}_{con}^{(y)}$	4096	4	1024

Our embedding model was trained on the Europarl v7 parallel corpus [25]. Europarl parallel corpus is widely used in statistical machine translation; it includes parallel text in 11 languages collected from the European parliament. Each language roughly has 1 million sentences (or 20-30 million words). The training of the NTDV models in this chapter used the English and German language subsets of Europarl. For training the attention-based NMT models and cNV, we adopted the same settings as in [31] with the following additional settings due to the distance constraint: $d_h = 1024$, $r = 4$, $N_s = 20$, $\beta = 0.5$, $\alpha = 2\sqrt{d_h}$. In training the attention-based NMT models, we used the default setting with a mini-batch size of 80, a vocabulary size of 85,000, a maximum sentence length of 50, a word embedding size of 500, and a hidden layer size of 1024. The models were trained with the Adam optimizer using the cross-entropy loss. Model parameters of cNV were copied from the pair of pre-trained NMT models. The three newly added shared layers were further trained for 5 epochs with the original NMT parameters fixed. Note that if third-party pre-trained NMT models are available, we only need to fine-tune the three shared layers with 5 epochs. Here we trained the NMTs from scratch for fair comparisons.

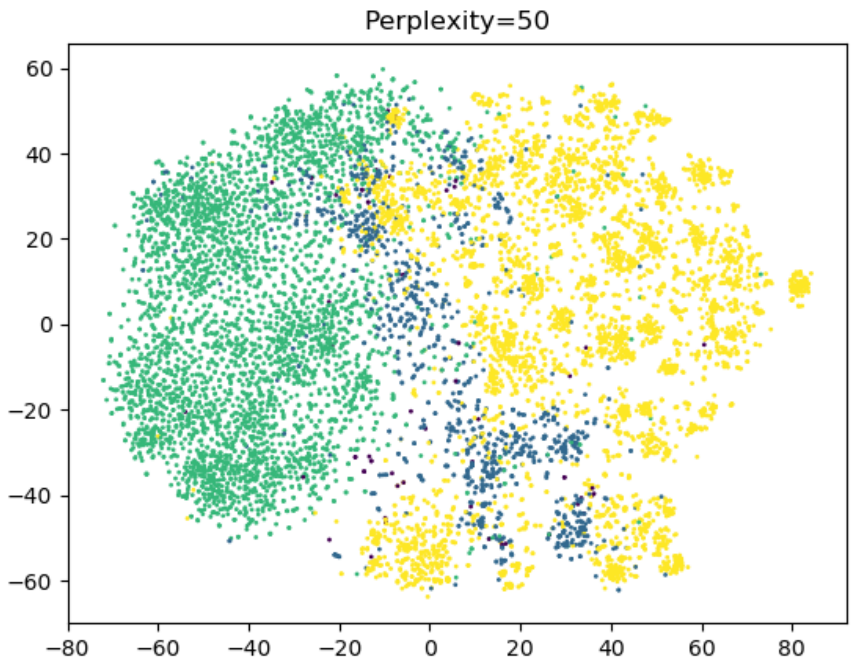
The cross-lingual document classification task on RCV1/RCV2 was used to evaluate the effectiveness of the NTDV models[30, 24]. In this task, about 1K English documents of four categories were converted to word/document embeddings and were given to classify the category of 5K German documents in the test set, and vice versa. If an SVM classifier (we used the same setting as in [31]) is trained on embeddings from 1K English data and can accurately classify embeddings from 5K German data, then it indirectly proves that English and German texts are projected into the same vector space. Note that the 1K training set here was used to train the SVM classifier, whereas the NTDV models were pre-trained on the task-independent Europarl corpus.

Six-fold cross-validation was further used to examine the statistical significance of the results. The six sets were constructed from the original training set plus the data sampled from test set (English from the de→en test set and German from en→de test set). The single sample t-test was conducted and differences between any two results are significant at the 99% confidence level when $p < 0.01$.

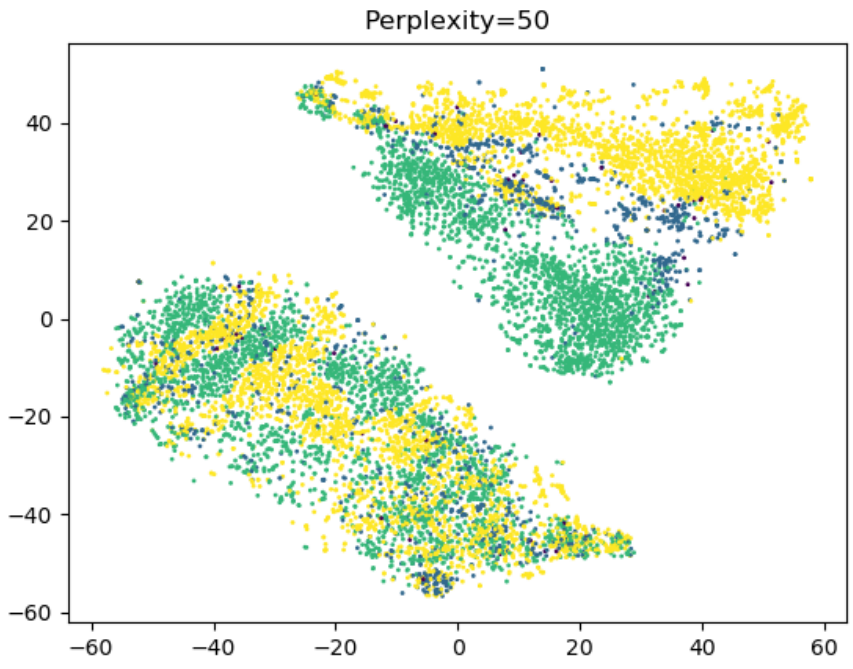
3.4 Results

Table 3.2 presents the performance of various embedding methods on the cross-lingual document classification task. MT_{bs} is the machine translation baseline from [24]. NMT_{bs} is the NMT model on which the cNV model was built. In NMT_{bs}, the classifier was trained with the term frequency/inverse document frequency (TF/IDF) of the most frequent 50000-word features. Among all the methods, $\text{cNV}_{con}^{(y)}$ is not significantly worse than the 'Unsup' method ($p = 0.143811$) in en→de. $\text{cNV}_{con}^{(y)}$ is significantly better than the 'BRAVE' method ($p = 0.000169$) in de→en.

Without the distance constraint, the English embedding and its corresponding German embedding of the same document are not guaranteed to lie in the same vector space and cannot be compared.

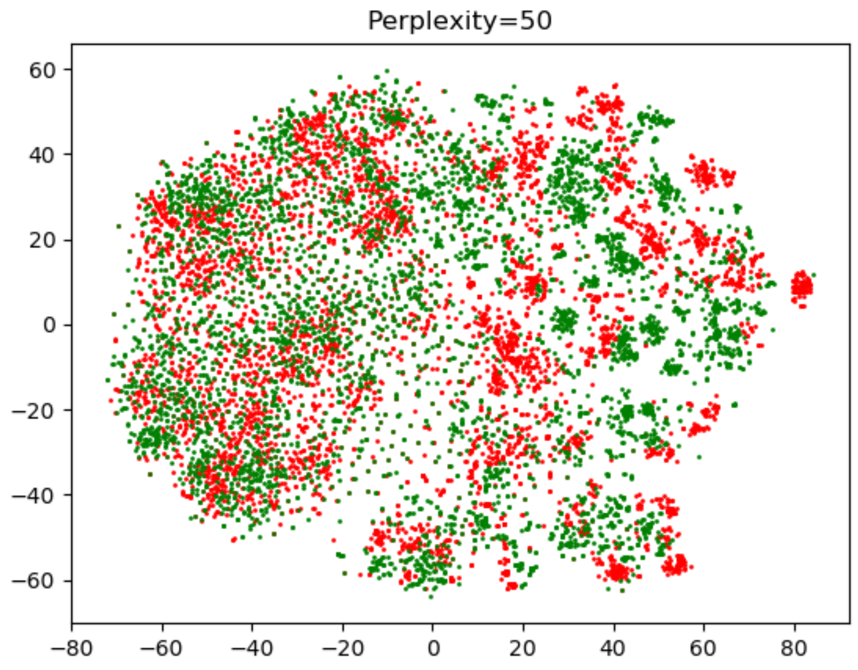


(a) $\text{cNV}_{sum}^{(en)}$; $\text{cNV}_{sum}^{(de)}$

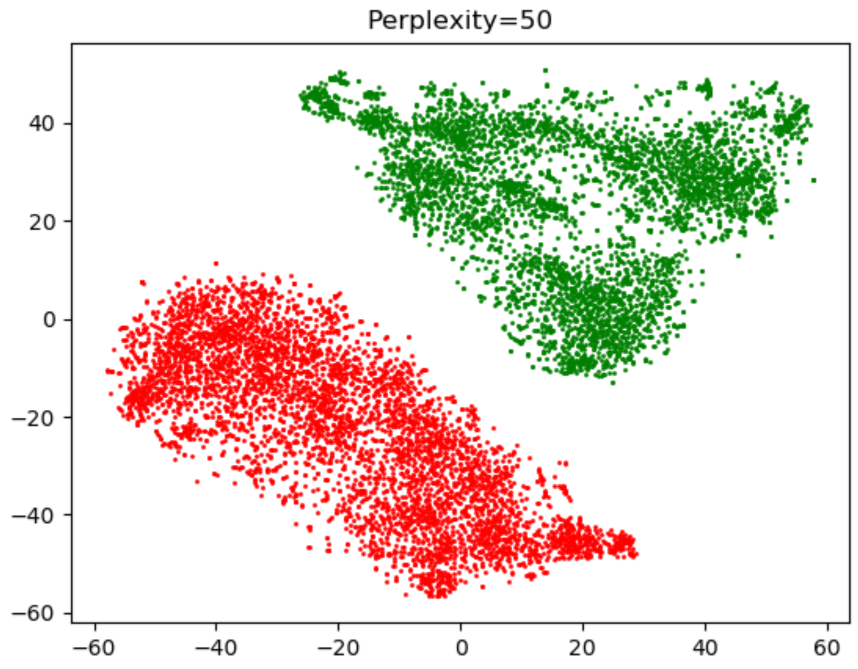


(b) $\text{NV}_{sum}^{(en)}$; $\text{NV}_{sum}^{(de)}$

Figure 3.2: The distribution of embeddings from the four document classes in the test set with and without distance constraint training.



(a) $\text{cNV}_{sum}^{(en)}$; $\text{cNV}_{sum}^{(de)}$



(b) $\text{NV}_{sum}^{(en)}$; $\text{NV}_{sum}^{(de)}$

Figure 3.3: the distribution of English and German embeddings in the test set with and without distance constraint training.

Table 3.2: *Classification accuracy (%) on RCV1/RCV2*

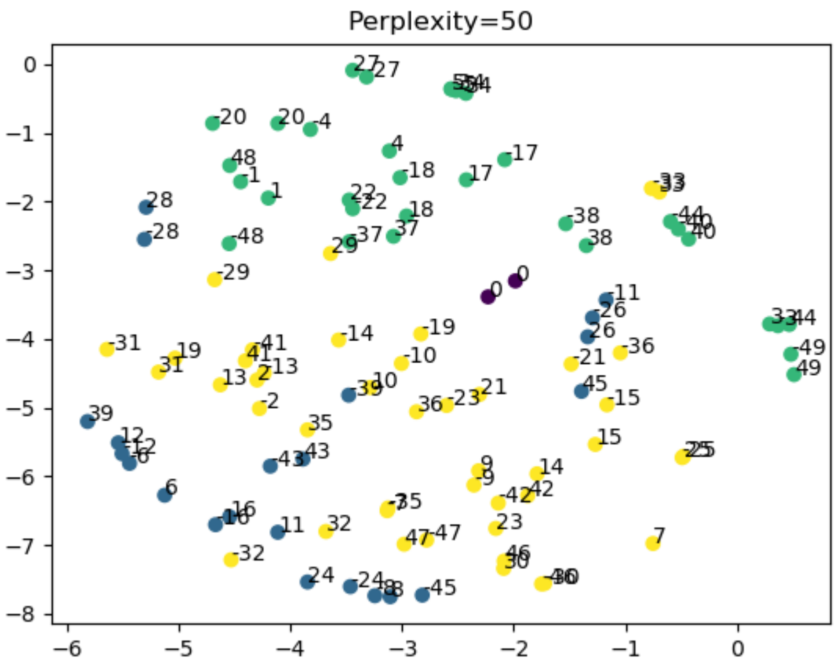
Method	en→de	de→en
$MT_{\mathtt{bs}}$ [24]	68.1	67.4
$NMT_{\mathtt{bs}}$	92.9	70.8
BAE [7]	91.8	74.2
ADD [20]	86.4	74.7
BI [20]	86.1	79.0
BRAVE [36]	89.7	80.1
MultiVec [5]	88.2	79.1
Unsup [34]	90.7	80.0
$NV^{(\mathtt{y})}_{\mathtt{sum}}$	35.6	41.9
$cNV^{(\mathtt{y})}_{\mathtt{sum}}$	88.9	79.1
$cNV^{(\mathtt{y})}_{\mathtt{con}}$	**89.8**	**81.0**

To further illustrate this, we visualize the document embeddings of different languages using their t-SNE projections in Fig. 3.2, 3.3 and 3.4.

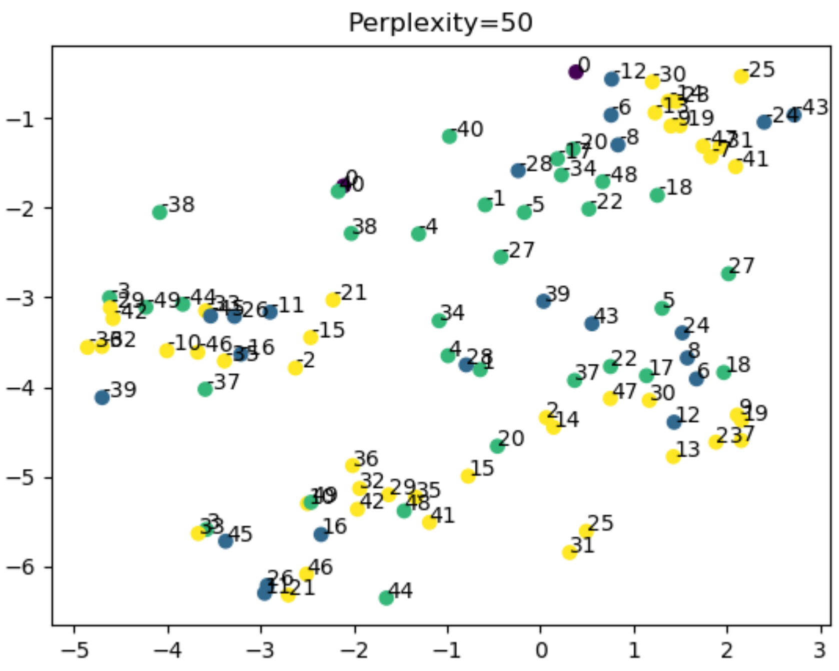
Fig. 3.2 shows the NV/cNV embeddings of German documents in the $\mathtt{en} \rightarrow \mathtt{de}$ test set. We also translated the same documents into a mirrored English set with the NMT to show the distribution of the same documents in a different language. In Fig. 3.2 , the 4 types of documents are plotted in 4 different colors (red: Corporate/Industrial, blue: Economics, green: Government/Social, yellow: Market). The document embeddings are clearly divided into 4 clusters, but only when they were trained with distance constraint then their German embeddings and English embeddings (from the translation of the German texts) are indistinguishable if they belong to the same type.

On the other hand, in Fig. 3.3, English embeddings are shown in red while their German counterparts of the same documents are shown in green regardless of their document types. It is clear that without the distance constraint training, the NV document embeddings are segregated by the languages into two major clusters. These figures show that simply sharing layers in the NV model without distance constraint training is insufficient to produce similar embeddings from two languages and cannot be compared.

Fig. 3.4 shows the t-SNE projections of 50 English-German document embedding pairs. The positive and negative indices refer to the English and German embeddings produced from the same indexed document in the German test set, respectively. For example, '-2' is the German embedding of the 2nd document in the test set, whereas '2' is the English embedding of the same 2nd German



(a) $\mathrm{cNV}_{sum}^{(en)}$; $\mathrm{cNV}_{sum}^{(de)}$



(b) $\mathrm{NV}_{sum}^{(en)}$; $\mathrm{NV}_{sum}^{(de)}$

Figure 3.4: English (+ve) and German (-ve) embedding pairs of 50 randomly sampled documents.

document after it is translated to English. It shows that cNV model, trained with the distance constraint, effectively pairs the German and English embeddings from two parts of the model close together, regardless of their different encoders and language inputs. On the other hand, the same document's English and German NV embeddings are far apart.

3.5 Summary

The NTDV models in this chapter are preliminary research before the T-MDE model in the next chapter, which is the central theme of this thesis. Compared to T-MDE, NTDV uses independent encoder and decoder structures. In the early stage of this study, it is not entirely clear whether a shared or an independent encoder/decoder structure should be adopted in the document embedding model. After completing our research on both cNV and cT-MDE models, a few retrospective conclusions can be made: Firstly, the use of independent encoders/decoders for different languages in the NTDV model makes it difficult to scale up to many languages, so it is only used in the bilingual situations in this research. Secondly, a shared encoder and decoder in the T-MDE model renders embeddings from all the languages in the training curriculum to get close with the embedding from the pivot language. On the other hand, the independent encoder and decoder would not be able to do that. Therefore, cNV would have to rely on distance constraint training to achieve the same effect. Finally, the independent encoder and decoder structure in NTDV can be easily adapted from a pair of existing bilingual NMT models, which have been much more abundant in the past years.

In this thesis, NTDV provides a perspective on how an alternative framework would work. Combined with the next chapter, this chapter exhibits the advantages and disadvantages of choosing an independent encoder/decoder structure. In retrospect, the shared encoder/decoder would provide many benefits in simplicity and performance. In the next chapter, the T-MDE model will use a shared encoder and decoder structure to embed many languages in a single model.

CHAPTER 4

T-MDE: TRANSFORMER BASED MULTILINGUAL DOCUMENT EMBEDDING

4.1 Introduction

Multilingual data for a specific task or service are scarce. Thus, transferring task/user-related knowledge from resource-rich languages to resource-scarce languages is vital for NLP applications when they extend to a new language. A traditional way of enabling cross-lingual knowledge sharing is to use a translation model to translate all task-related data of different languages into a single pivot language. Then, we can use train/test data from different language sources on a single classification model through this pivot language.

A cross- or multi-lingual text embedding model can project texts from different languages into a common vector space and produces a language-independent representation of words [24, 10, 7, 5] or sentences/documents [20, 41, 36, 5]. This embedding model would be faster than a translator of comparable quality (as it does not require a decoder when producing embedding vectors), and the fixed-sized vector it produces would be a better input for comparison, indexing and clustering compared to text features [20, 7].

LASER [43, 4] and XLM-R [26, 8] are two of the current state-of-the-art (SOTA) multilingual sentence/document embedding models. LASER is designed after multilingual neural machine translation (NMT). It uses a bidirectional-LSTM (BiLSTM) encoder and an LSTM decoder, and the sentence embedding vector is obtained from max-pooling the encoder outputs. XLM-R, on the other hand, is a transformer-based model that employs a token masking and sentence entailment classification framework. Inspired by LASER, BERT and XLNet [12, 54], this thesis introduces a transformer-based multilingual sentence embedding model, which we call Transformer Based Multilingual Document Embedding (*T-MDE*), and compares its performance with other SOTA document embedding models.

The first contribution of the T-MDE model is that it develops a bottleneck transformer structure for multi-lingual document embedding. T-MDE inherits LASER's strength in dealing with a large number of languages in multilingual situations. At the same time, the new transformer-based model performs better and runs faster than the BiLSTM-based baseline owing to the new transformer structure. LASER also has a information bottleneck, however, it is LSTM and max-pooling based rather than self-attention based.

Compared to multilingual BERT [12, 13] (mBERT) and XLM-R, T-MDE adopts the multilingual NMT framework. More importantly, T-MDE's decoder can only obtain the sentence embedding instead of the entire encoder output. This *information bottleneck* design between the encoder and decoder compiles the entire encoder's information into a single sentence embedding vector. We find this *information bottleneck* would help improve the quality of document embedding.

T-MDE's another contribution is to present a novel loss function that combines a distance constraint loss with the traditional translation loss for cross-lingual sentence embedding and discusses its effect on both LASER and T-MDE architectures. We would label the T-MDE model trained with this new loss function as *cT-MDE*, whereas the LASER model trained with this new loss function as *cLASER*. The distance constraint loss term significantly raises the performance in multilingual settings. Moreover, it can deal with the bilingual setting, which the traditional multilingual NMT setting is not designed for and thus does not give a good performance.

In this thesis, both T-MDE and our LASER implementations are pre-trained only on the publicly available Europarl v7 parallel corpus. Unlike translation models whose performance could be directly measured by the BLEU score, the document embedding model's quality is measured by the performance of these embedding vectors in downstream tasks. Here, we take a commonly used classification task on RCV1/RCV2 (MLDoc [44]) as the downstream task. We convert all train/test data in MLDoc to document vectors using T-MDE, LASER and XLM-R and examine their cross-lingual classification accuracies on a classifier. No task-related data (MLDoc) is used to train the embedding models. This chapter is edited and converted from my previous paper: Transformer based Multilingual document Embedding model[33].

4.2 Model architecture

Figure 4.1 illustrates the architecture of our proposed T-MDE model. The model is based on the transformer translation model [50]. It has six encoder layers and one decoder layer. During training, the model uses input from one language (e.g., German) to predict its translation target (e.g., English).

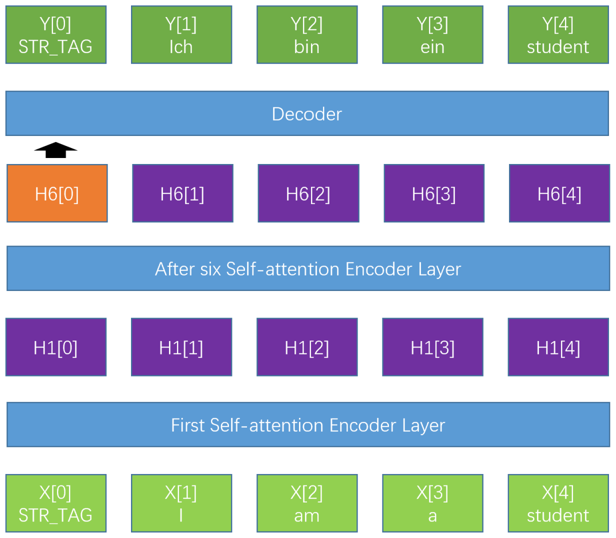


Figure 4.1: Architecture of T-MDE.

During testing, only the encoder and the input language is needed to produce a sentence/document embedding.

Like LASER, T-MDE also uses a shared encoder and a shared decoder for all different language pairs. The source language would cycle through all languages in the training corpus during T-MDE training, whereas the target language alternates only between two pivot languages, English and Spanish. For the five languages we currently use in the Europarl corpus, the training language pairs are de-en, de-es, en-es, es-en, fr-en, fr-es, it-en, and it-es.[1] Driven by the translation loss of the English or Spanish target of the decoder, the embeddings of parallel input sentences written in the 5 languages (i.e., English, Spanish, French, Italian, and German translation of the same sentence) should get closer to each other.[2]

4.2.1 Model details of T-MDE

Unlike LASER, we use the first encoder output as sentence embedding and adopt a BERT-like strategy that eliminates the need for left padding. We insert a special '*STR_TAG*' token to the first position of every sequential text input (which can be a sentence or a paragraph depending on the tasks) so that the model would know that the first token is reserved for generating sentence

[1]The common shorthand notation of the languages is adopted. de: German; en: English; es: Spanish; fr: French; it: Italian.

[2]While a non-pivot language such as French would have two target languages during training, there will be only one target language (Spanish/English) for English/Spanish input.

embedding. Thus, the embedding of *STR_TAG* effectively becomes a learnable query fixed for every input sentence. Due to the nature of multi-head attention, each head will use this first token to query every token in the input sequence to find a weighted average of all their information from a different perspective. After six layers of such summarization and information extraction, the first token of the encoder output, $\mathbf{H}^6[t_0]$, would be the proper sentence embedding $\mathbf{P}$ of the input text. The parameters of the *STR_TAG* embedding and its associated projection matrix are learned to summarize the overall sentence-level information during training better.

The following formulas summarize the multi-head attention function $\mathcal{F}_{MH}$ from the original transformer paper [50]:

$$\mathbf{U}_i = \mathtt{softmax}\left(\frac{(\mathbf{Q}\mathbf{W}_i^Q)(\mathbf{K}\mathbf{W}_i^K)^T}{\sqrt{d_h}}\right)(\mathbf{V}\mathbf{W}_i^V) \tag{4.1}$$

$$\mathcal{F}_{MH}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathbf{U}_1,\ldots,\mathbf{U}_r)\mathbf{W}^O \tag{4.2}$$

where U_i is the i-th attention head; r is the number of attention heads; d_h is the size of the encoder's hidden layers; Q, K, V indicate the role of *query, key* and *value*; $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are matrices constructed by sequentially arranging the vectors of *query*, *key* and *value*; $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ are projection matrices associated with the i-th attention head; $\mathbf{W}^O$ is the projection matrix for the concatenated attentions from various heads. $\mathbf{U}_i$ should be a dimension of $F \times T$, where T is the length of the sequence F is the feature dimension (i.e., the width of the transformer). Concat is the concatenation of matrices along the feature dimension.

Let $\mathrm{NMT}_{a\to b}$ be the translation of a sentence from language a to language b, the current time step be t_i, and the target outputs be $\mathbf{Y}$. On the encoder side, the input token $\mathbf{X}$ is first embedded and then goes through six self-attention encoder layers, where 'key', 'query', and 'value' are all coming from the previous layer's output:

$$\mathbf{H}^0 = \mathsf{FFN}_{emb}(\mathbf{X}) \tag{4.3}$$

$$\mathbf{C}^k = \mathcal{F}_{MH}(\mathbf{H}^k,\mathbf{H}^k,\mathbf{H}^k) \tag{4.4}$$

$$\mathbf{H}^{k+1} = \mathsf{FFN}^k(\mathbf{C}^k) \tag{4.5}$$

where FFN^k is the feed-forward network in the k-th encoder layer. FFN_{emb} is the embedding layer.

On the decoder, the output $\mathbf{J}^k$ (where $k > 0$) from the k-th decoder layer is computed as follows:

$$\mathbf{J}^0 = \mathsf{FFN}(\mathbf{H}^6[0]) \tag{4.6}$$

$$\mathbf{J}^{k+1} = \mathsf{FFN}^k(\mathbf{J}^k) \tag{4.7}$$

where FFN^k is the feed-forward network in the k-th decoder layer. FFN is the feed-forward network between the encoder output and the decoder input.

This $\mathrm{NMT}_{a\to b}$ translation path would result in the translation loss $l_{mt}^{a\to b}$. Similarly, the translation of the same sentence from language b to language a ($\mathrm{NMT}_{b\to a}$) will result in the translation loss of $l_{mt}^{b\to a}$.

4.2.2 Information bottleneck and T-MDE_P

In T-MDE, we only allow the sentence embedding as 'key' and 'value' for the encoder-decoder attention. This *information bottleneck* is the reason why T-MDE is better than mBERT and XLM-R in 'zero-shot' cross-lingual document embedding tasks. In addition, we also develop an improved version of T-MDE with a more effective bottleneck self-attention, it is denoted as T-MDE_P.

Let $\mathbf{Z}$ be an array of learnable vectors (32 in total). At the 4-th layer, we use these learnable vectors to query on the outputs of the 4-th layer $\mathbf{H}^4$. We expect these learnable vectors to adapt to the entire training data and become intelligent filters to extract helpful information from the lower layers..

$$\mathbf{Z}_x^4 = \mathcal{F}_{MH}(\mathbf{Z}, \mathbf{H}^4, \mathbf{H}^4) \tag{4.8}$$

$$\mathbf{Z}_y^4 = \mathsf{FFN}^4(\mathbf{Z}_x^4) \tag{4.9}$$

$$\mathbf{Z}_x^5 = \mathcal{F}_{MH}(\mathbf{Z}_x^4[t_0 : t_{16}], \mathbf{Z}_x^4, \mathbf{Z}_x^4) \tag{4.10}$$

$$\mathbf{Z}_y^5 = \mathsf{FFN}^4(\mathbf{Z}_x^5) \tag{4.11}$$

$$\mathbf{Z}_x^6 = \mathcal{F}_{MH}(\mathbf{Z}_x^5[t_0 : t_8], \mathbf{Z}_x^5, \mathbf{Z}_x^5) \tag{4.12}$$

$$\mathbf{Z}_y^6 = \mathsf{FFN}^4(\mathbf{Z}_x^6) \tag{4.13}$$

where $\mathbf{Z}_x^i[t_0 : t_j]$ is a truncation of $\mathbf{Z}_x^i$ taking only the first j-th elements. At the 5-th and 6-th layers, we cut the query size of self-attention by half ($\mathbf{Z}_x^4$: 32, $\mathbf{Z}_x^5$: 16 $\mathbf{Z}_x^6$: 8) and further compress information like a pyramid. Note that the output produced by the learnable filter $\mathbf{Z}$ and the encoder layer output $\mathbf{H}$ share the same model parameter and weight matrix (i.e., $\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$). Therefore, the number of parameters in T-MDE and T-MDE_P for each layer will be the same. The increase of parameter size due to $\mathbf{Z}$ itself will be marginal as $\mathbf{Z}$ is much smaller than regular weight matrixes. Moreover, due to the very small query length of $\mathbf{Z}$, T-MDE_P will be as fast as the basic T-MDE during training.

In T-MDE_P, the final sentence embedding $\mathbf{P}$ attends all the first position encoder output $\mathbf{H}^i[t_0]$ of the previous layers and $\mathbf{Z}_y^6$. This setup aims to emphasize the critical role of $\mathbf{H}^i[t_0]$, so that valuable information in each layer can be gathered to $\mathbf{H}^i[t_0]$ and are directly summed to the final sentence embedding $\mathbf{P}$. Moreover, we also believe this setup can help the distance constraint training (see the section below) penetrate to lower layers as it offers an incentive to form language-independent representation to $\mathbf{H}^1[t_0]$ and $\mathbf{H}^6[t_0]$ at the same time.

$$\mathbf{D}_y = \text{Concat}(\mathbf{H}^6[t_0], \ldots, \mathbf{H}^1[t_0]) \tag{4.14}$$

$$\mathbf{D}_c = \text{Concat}(\mathbf{D}_y, \mathbf{Z}_y^6) \tag{4.15}$$

$$\mathbf{P}_x = \mathcal{F}_{MH}(\mathbf{H}^6[t_0], \mathbf{D}_c, \mathbf{D}_c) \tag{4.16}$$

$$\mathbf{P} = \mathsf{FFN}^k(\mathbf{P}_x) \tag{4.17}$$

where $\mathbf{D}_y$ is the concatenation of first position encoder outputs $\mathbf{H}^i[t_0]$ from the 6-th to the 1-st layers.

4.2.3 Training with a distance constraint

To further encourage the embeddings of the same sentence but written in different languages to be as close as possible, we propose to add a novel distance constraint loss to the training loss function. This distance constraint loss is inspired and modified from negative sampling in word2vec [35] and the noise-constrastive loss in [20]. The distance constraint loss is computed from the following three terms during the translation of a sentence from language a to language b:

- d_p: distance between the embeddings from a parallel sentence pair,
- d_{n_j}: distance between the embeddings from a non-parallel sentence pair where the unrelated target is obtained from negative sampling, and
- δ_{n_j}: the difference between d_p and d_{n_j} with a margin.

The three terms are defined as follows:

$$d_p = \frac{|\mathbf{P}^{(a)} - \mathbf{P}^{(b)}|^2}{\nu_{norm} + \epsilon} \tag{4.18}$$

$$d_{n_j}^{a \to b} = \frac{|\mathbf{P}^{(a)} - \mathbf{P}^{(b_j)}|^2}{\nu_{norm} + \epsilon} \tag{4.19}$$

$$\delta_{n_j}^{a \to b} = \max(0, \alpha - (d_{n_j}^{a \to b} - d_p)) \tag{4.20}$$

where $\mathbf{P}^{(z)}$ is the sentence embedding from a language z; α is the margin; $|\cdot|^2$ is the Euclidean distance; ν_{norm} is the average Euclidean distance computed from all sentence pairs in a training batch; ϵ is a small value added to prevent division by zero. The notation $a \to b$ indicates that a quantity/variable is produced in the translation of a sentence from language a to language b.

(4.18) is used to minimize the distance between the two embeddings of a parallel sentence pair $\{\mathbf{P}^{(a)}, \mathbf{P}^{(b)}\}$, whereas (4.19) is used to maximize the distance between the two embeddings of a non-parallel sentence pair $\{\mathbf{P}^{(a)}, \mathbf{P}^{(b_j)}\}$, where $\mathbf{P}^{(b_j)}$ is the sentence embedding obtained from a randomly sampled j-th sentence of language b [20]. In both (4.18) and (4.19), the distances are normalized by the averaged distance norm in a training batch. We find that such normalization renders the model training much more stable, resulting in better model performance as well. Finally, $\delta_{n_j}^{a \to b}$ in (4.20) is designed to maximize the marginal distance between embeddings of parallel sentences and non-parallel sentences (α is the margin). Please see the Discussion section for a more detailed explanation of the equation.

Lastly, the total loss of the model for the translation of a sentence from language a to b, $l^{a \to b}$, is the sum of the paired sentence distance loss d_p, averaged deltas δ_{n_j} from N_s negative samplings and the translation cost $l_{mt}^{a \to b}$:

$$l^{a \to b} = \beta(d_p + \frac{\lambda}{N_s} \sum_{j=1}^{N_s} \delta_{n_j}^{a \to b}) + 0.5 * l_{mt}^{a \to b}, \tag{4.21}$$

where $0 \leqslant \beta, \lambda \leqslant 1$ are weights. Our unique distance-constraint-loss formula with the separation of absolute distance (controlled by β, for tighter cluster) and marginal distance (controlled by λ, for better differentiation) is essential for stable performance gain. This novel cost function ensures that the produced sentence embedding vectors contain ample semantic information and are, at the same time, very similar for parallel sentences in a language pair.

4.3 Experimental evaluation

4.3.1 Training

Table 4.1: *Dimensions of various components in our LASER and T-MDE models.*

Model	d_h	d_z	r	d_{fc}	#enc	#dec	#vocab	#param	WPS	WPS_{dist}
$LASER_{2018}$	512×2		-	-	1	1				
LASER_1	512×2	2048	-	-	1	1	50K	209M	14.3K	12.2 K
LASER_6	512×2	2048	-	-	6	1	50K	241M	12.4K	9.6K
XLM-R	768	-	12	3072	12	-	250K	270M	-	-
T-MDE	1024	1024	16	4096	6	1	50K	246M	23.5K	15.2 K
T-MDE_P	1024	1024	16	4096	7	1	50K	259M	21.2K	15.1 K

d_h *is the size of the hidden layer of the encoder, which is also the sentence embedding size;* d_z *is the size of the hidden layer of the decoder.* r *is the number of attention heads,* d_{fc} *is the dimension of FFN layers in the transformer. Since LASER adopts a BiLSTM encoder, which has an LSTM of size 512 for each direction, the embedding size on the encoder side is 512×2. '#enc,' '#dec,' '#vocab' and '#param' are the number of encoder layers, decoder layers, vocabularies and parameters of the model.* WPS *is the training speed (word per second) of the model,* WPS_{dist} *is the training speed when distance constraint loss was used in the training*

We implemented both T-MDE and LASER using fairseq [39].[3] Note that the LASER model in this thesis is re-implemented according to the SOTA version [4]. Both models were trained on the Europarl v7 corpus using the following 5 languages: en, de, fr, es and it, which are also used in the previous literature [44, 25].

For training our LASER and cLASER (LASER structure trained with the proposed additional distance constraint) models, we adopted the same settings as in LASER [4] with the following additional settings due to the distance constraint loss: $N_s = 40$, $\lambda = 1$, $\alpha = 0.5$. While other parameters were decided by heuristic or from related publications, β is the main parameter for deciding the weight between the translation loss and the distance constraint loss. For training cLASER and cT-MDE, we chose the value of β among $\{0.25, 0.5, 1.0\}$ according to the performance of the models sampled at the 10th training epoch on the MLDoc development set; the best value was found to be $\beta = 0.25$.

Both our LASER and T-MDE implementations used the same Byte-Pair Encoding (BPE) dictionary with 50K vocabulary (approximately 10K vocabulary for each language) and went through the same text processing. Their training used the same batch size of 128K (maximum

[3]Our LASER implementation was based on code from [48]. Our cT-MDE code can be found in `https://github.com/ever4244/tfm_laser_0520`.

number of tokens in a batch), the same uniform sampling-based data-loader and training curriculum when languages were alternated. For training T-MDE, we adopted the default setting from the 'transformer_vaswani_wmt_en_de_big' architecture in the fairseq toolkit [39, 50].[4] We also applied layer normalization before each encoder/decoder block as it is claimed that training can be more stable in that way [50]. Ten to twenty epochs were run and the best model was selected according to its performance on the MLDoc development set.

For XLM-R, we made three different attempts. Firstly, we trained an XLM-R$_{\mathtt{europarl}}$ model with the default 'MLM+TLM' setting [27] using the Europarl corpus. Moreover, we downloaded the pre-trained XLM-R$_{\mathtt{base}}$ and XLM-R$_{\mathtt{large}}$ models and tested them with the same zero-shot setting as in LASER and T-MDE. We found that XLM-R$_{\mathtt{base}}$ using the first encoder output has the best performance among the three; therefore, we will cite XLM-R$_{\mathtt{base}}$ as XLM-R from now on. Note that the pre-trained XLM-R model is trained with the 100 language Common Crawl Corpus [27].

4.3.2 Model structure comparison

Table 4.1 shows the model structure of our LASER and T-MDE models. LASER_1 and LASER_6 were implemented according to [4] with 1 and 6 layers of BiLSTM in their encoders, respectively. $\mathtt{d_h}$ is the size of the hidden layer of the encoder, which is also the sentence embedding size; $\mathtt{d_z}$ is the size of the hidden layer of the decoder. $\mathtt{d_{fc}}$ is the dimension of FFN layers in the transformer. Since LASER adopts a BiLSTM encoder, which has an LSTM of size 512 for each direction, the embedding size on the encoder side is 512×2. '#enc,' '#dec,' and '#vocab' are the number of encoder layers, decoder layers, and vocabulary size of the model. LASER$_{2018}$ is the LASER model published in [44]; its performance is used to gauge our LASER implementations.

Albeit T-MDE and T-MDE_P have more parameters ('#param'), they still run faster with higher word-per-second (WPS)[5] and shorter average time per epoch than LASER_6 and even LASER_1. Finally, WPS$_{\mathtt{dist}}$ is the WPS when the models were trained with the additional distance constraint. Although the additional constraint slows down training for 15–35%, WPS$_{\mathtt{dist}}$ for T-MDE is still higher than WPS for LASER_6 even though the two models have about the same number of parameters.

[4] Adam-betas = (0.9, 0.98), lr (learning rate) = 0.0005, lr-scheduler = inverse_sqrt, label-smoothing = 0.1, dropout = 0.3, and weight-decay = 0.0001.

[5] WPS was measured on one Titan RTX GPU with CUDA 10.1, Nvidia-APEX and mixed-precision training (–fp16).

4.3.3 Implementation details

The new Multilingual Document Classification Corpus (MLDoc) dataset [44] for cross-lingual document classification task on RCV1/RCV2 was used to evaluate the quality of the document embedding. The dataset was split into a 1K train set, a 1K development set, and a 4K test set. MLDoc is improved over the older CLDC dataset [30, 24]; it has a uniform class distribution of four classes (‘ECAT’, ‘CCAT’, ‘GCAT’, ‘MCAT’) in eight languages.

Evaluation I: *English-Y cross-lingual document classification.* We first performed English-Y document classification in which the cross-lingual document classifier is a feed-forward network (FFN) that was trained with document embeddings derived from a training set of 1K English-only documents. The hyperparameters were optimized on the English development set. The same English-trained FFN classifier was then used to classify test documents in other 4 languages; i.e., Y = {de, fr, es, it}. Thus, the classifier has never seen languages other than English during training and hyperparameters fine-tuning.[6] If the document embedding model can successfully project different languages into a common vector space, the FFN classifier trained on the embeddings of one language (here, English) would be able to classify the document embeddings of another language (e.g., German) in the test set.

Evaluation II: *X-Y cross-lingual document classification.* Then the same models were evaluated on the general MLDoc cross-lingual document classification task where X and Y can be any language among {en, de, fr, es, it}.

Evaluation III: *Bilingual document classification.*

This experiment is designed to show the effectiveness of the distance constraint training without the interference of pivot languages. In the experiment, a pair of T-MDE and cT-MDE models were trained using exactly the same training procedure as the multilingual models but with only two translation language pairs, en-es and es-en. We took both models at their 10th training epoch and tested them on the MLDoc dataset.

[6] To ensure a fair comparison, we used the same FFN and testing script from the LASER Github `https://github.com/facebookresearch/LASER`.

4.4 Results

4.4.1 Evaluation I

Table 4.2 compares the performance of our T-MDE model with some published results. In the table, multiCCA was trained on dictionaries and some monolingual data, and mBERT was trained on a concatenated Wikipedia corpus, while LASER_5$_{2019}$ was trained with the large 93-language corpus combination. Our T-MDE and LASER$_{2018}$ (which has fewer encoder layers than LASER_5$_{2019}$) were trained with the smaller Europarl corpus. LASER_1 and LASER_6 are our re-implementations of the latest LASER model based on LASER_5$_{2019}$ except that LASER_1 uses 1 encoder layer and LASER_6 uses 6 encoder layers (instead of 5 in LASER_5$_{2019}$), and they were trained with the same Europarl corpus subset as T-MDE and LASER$_{2018}$.

Europarl corpus is the standard pre-training data in the paper which designed the MLDoc experiment. It is difficult for us to replicate the current LASER training data when the processing script is not disclosed.[7] Training on the 93-language corpus is also very costly.[8] Therefore, our re-implemented models were trained with the Europarl instead of using other uniquely sampled and combined corpus collections. It would also be more convenient for other researchers to reference and compare when our models were trained with the standard pre-training data like Europarl.

Table 4.2: *MLDoc English-Y classification accuracy (%) of T-MDE vs. published results.*

	de	**es**	**fr**	**it**	**Avg.**
MultiCCA [44]	81.2	72.5	72.4	69.4	73.9
mBERT [53]	80.2	72.6	72.6	68.9	73.6
LASER_5$_{2019}$ [4]	84.8	77.3	78.0	69.4	77.4
T-MDE*	84.6	73.8	74.9	70.5	76.0
LASER_1*	78.9	71.8	72.3	68.1	72.8
LASER_6*	85.5	62.5	73.5	67.4	72.2
LASER$_{2018}$* [44]	71.8	72.8	66.7	60.7	68.0

*Models labelled with * are trained with the smaller Europarl corpus while the others are trained with more data.*

It can be seen that the performance of LASER_6, which is our re-implementation of LASER_5$_{2019}$, lies between LASER_5$_{2019}$ and the older LASER$_{2018}$. It clearly shows that the newer, bigger LASER

[7]It is the combination of Europarl, United Nations, OpenSubtitles2018, Global Voices, Tanzil and Tatoeba corpora, consisting of 93 languages.

[8]It would take five days using 16 V100 GPUs.

model of 2019 performs better than its smaller predecessor of 2018. As LASER_6 was implemented according to LASER_5$_{2019}$, we believe the performance difference between our LASER_6 and LASER_5$_{2019}$ is due to the difference in the amount of training data: 5-language Europarl subset in LASER_6 and the 93-language corpus combination in LASER_5$_{2019}$ [4]. On the other hand, even though our new T-MDE model was also trained with the smaller 5-language Europarl data, it achieves comparable performance as that of LASER_5$_{2019}$, showing the efficacy of the transformer implementation used in our T-MDE.

The English-Y result is not a good measurement of overall embedding quality.[9] Choosing English as the default vector space for cross-lingual embedding is not always the best option [2]. Due to the enormous quality and quantity difference in the training data between T-MDE/LASER$_{2018}$, multiCCA, mBERT and LASER_5$_{2019}$, it is unfair to make a direct comparison between these models. Nevertheless, more English-Y results were reported in the literature than the general X-Y results for the MLDoc dataset. Despite the limitations mentioned above, the English-Y evaluation allows the readers to compare our new proposed model against more previous publications in the same task.

4.4.2 Evaluation II

Table 4.3: *MLDoc X-Y classification accuracy (%) of various LASER models trained with the 5-language Europarl subset.*

Model	Cross	Same	All	WPS
LASER$_{2018}$	68.0	88.3	72.6	
XLM-R	69.4	87.9	73.1	
LASER_1	70.5	88.5	74.1	14.3K
LASER_6	70.3	87.2	73.7	12.4K
cLASER_6	72.6	87.8	75.6	9.6K
T-MDE	73.5	88.6	76.5	23.5K
cT-MDE	75.2	89.2	78.0	15.2 K
cT-MDE_P	76.0	88.8	78.5	15.1 K

Table 4.3 summarizes the MLDoc X-Y document classification performance of our re-implemented LASER models and the newly proposed T-MDE models together with their ‘c-versions’ that were

[9]One may easily tune up English-Y performance by using an unbalanced training curriculum.

Table 4.4: *Significant tests of the cross-lingual accuracy ('Cross') between LASER_6 and T-MDE models in the paired t-test. A result is significant at* $p < 0.01$.

Group A	**Group B**	**p**
LASER_6	**cLASER_6**	<0.0001
T-MDE	**cT-MDE**	0.0008
LASER_6	**T-MDE**	0.0001
cT-MDE	**cT-MDE_P**	0.0048

trained with the distance constraint. In the tables, 'Cross' is the average accuracy of all X-Y cross-lingual tests given by a model (where $X \neq Y$; 20 in total for five languages). This result shows the ability of the model to embed texts of different languages into the same vector space or transfer knowledge from a training language to a test language. 'Same' is the average accuracy for the five monolingual X-X tests. 'All' is the average of all the 25 cross-lingual and monolingual tests. All LASER models in Table 4.3 were trained and tested with the same uniform setup as T-MDE's. Therefore, it is fair to compare the results between LASER_1 and LASER_6 with T-MDE and cT-MDE. We validate the significance of the results difference by conducting paired (two-tailed) t-tests; please refer to Table 4.12 and the Discussion section for details.

In Table 4.3, LASER_6 and LASER_1 were implemented according to the latest SOTA LASER. Hence LASER_1 and LASER_6 perform better than the older LASER_{2018} by 2.5% and 2.3% for the cross-lingual performance (results under 'Cross'), respectively overall. This proves that our LASER re-implementation and testing procedure are sound. The performance gap between LASER_1 and is LASER_6 insignificant (p=0.0373).

LASER_6 has the same number of encoder layers and a similar parameter number compared to T-MDE. However, its WPS is only half of the T-MDE's, and T-MDE outperforms it by 3.2% in cross-lingual tests ('Cross'). Even our cT-MDE (trained with distance constraint) has a faster WPS than LASER_6 and still beats LASER_6 by nearly 5%. The effectiveness of the distance constraint training is also confirmed by comparing the performance of LASER_6 and cLASER_6: the latter outperforms the former by 2.3%. Compared to the basic T-MDE, the T-MDE_P with a more complicated information concentration design and learnable intelligent filter has an 0.8% increment in performance with a negligible decrease in training speed (table 4.1).

Interestingly, although distance constraint training significantly improves cross-lingual classification performance, it contributes little to monolingual performance. This is not unexpected as the distance constraint is only designed to pull the embeddings of the same text in different languages closer. Although it could be argued that T-MDE has more parameters and encoder layers than LASER_1 and LASER$_{2018}$, T-MDE and cT-MDE still have a faster training speed (WPS) compared to LASER_1. This is because the transformer layer allows better GPU computational parallelism than LSTM.

Our T-MDE model performs 4% better than the XLM-R model[10] in Table 4.3, despite the fact that both models employ a transformer encoder as their principal method of information extraction. We believe the difference lies in the design philosophy between these two models. In T-MDE, we only allow the sentence embedding as 'key' and 'value' for the encoder-decoder attention. By purposefully creating such *information bottleneck*, we compile all context information into the sentence/document embedding (first position of the encoder output) in the pre-training stage.

4.4.3 Evaluation III

Table 4.5: *Bilingual experiment with English and Spanish only.*

	T-MDE		**cT-MDE**	
Train	**en**	**es**	**en**	**es**
en:	87.3	42.8	88.4	68.0
es:	60.6	91.5	67.7	91.6
	Same:	89.4	**Same**:	90.0
	Cross:	51.7	**Cross**:	67.9

Table 4.5 shows a pair of T-MDE and cT-MDE models trained under a bilingual setting. The bilingual T-MDE models would use the exact same training setting as the multilingual models but with only two translation language pairs ('en-es, es-en'). We took both models at the 10-th epoch and tested them on the MLDoc dataset. It can be seen that while both models have similar monolingual document classification performance (89.4% and 90%), cT-MDE has a much better cross-lingual performance of 67.9% vs. 51.7% by T-MDE.

This bilingual experiment would demonstrate two points. Firstly, the LASER/T-MDE framework needs pivot languages on both the target and source sides. The 'en-es, es-en' training curriculum lacks the pivot language on the source side (a common input language for the same target language). Thus T-MDE has excellent same-language performance and bad cross-lingual performance. Secondly,

[10]XLM-R model is chosen among the best in the MLDoc test from XLM-R$_{\mathtt{base}}$ XLM-R$_{\mathtt{large}}$ and XLM-R$_{\mathtt{europarl}}$.

we can overcome this problem by including the distance constraint term and get much better cross-lingual performance (over 16%). In the previous five-language setting, both the translation and distance constraint losses would drive the two document embeddings closer. In this bilingual case, due to the lack of the pivot language on the source side, the translation loss is mainly used to model the information adequacy (the same language performance). The distance constraint loss is the most important driving force between cross-lingual knowledge transfer. Through this bilingual experiment, we can examine the effectiveness of the distance constraint training more independently. Moreover, it also shows that the distance constraint training can offer a simple solution to the cases where pivot language is absent [11].

4.4.4 The effectiveness of the bottleneck attention

XLM-R uses a language modeling framework that does not have such information concentration. We further conduct an ablation study on the T-MDE model by replacing the bottleneck attention (on the first position) with the traditional full-length attention (over the entire encoder outputs) between the encoder and decoder. We name the model variant that has the full-length attention as T-MDE_F. In T-MDE_F, the document/sentence embedding is still taken from the encoder output's first position as in T-MDE. However, it has the complete decoder mechanism of the transformer NMT. The decoder's translation/generation task takes all the encoder information instead of just querying on the encoder output's first position. From Table 4.6, it is clear that T-MDE (with *information bottleneck*) is significantly better than T-MDE_F (with full-length attention) in terms of embedding quality and training speed.

Table 4.6: *MLDoc X-Y classification accuracy (%) of T-MDE with and without information bottleneck.*

Model	**Cross**	**Same**	**All**	**WPS**
T-MDE	73.5	88.6	76.5	23.5K
T-MDE_F	60.7	82.5	65.1	17.4K

[11] On the other hand, an 'en-en,en-es,es-es,es-en' training curriculum would be twice costly.

4.4.5 The impact of β and cross-task performance of distance constraint training

Table 4.7 shows that β has a consistent and significant positive impact on cross-lingual accuracy while having little impact on monolingual accuracy. We also tested the exact same T-MDE, LASER_6 models on the 'Cross-lingual Natural Language Inference corpus' (XNLI) [9] for the cross-tasks stability of distance constraint training. Note that we used a much smaller pre-training corpus compared to LASER_5_{2019}. XLM-R's performance for XNLI is abysmal without fine-tuning.[12]

Table 4.7: *The effectiveness and sensitivity of distance constraint training (β).*

		cT-MDE		**cLASER_6**	
	β	**Same**	**Cross**	**Same**	**Cross**
MLDoc	0.5	88.9	74.2	87.7	72.1
	0.25	89.2	75.2	87.8	72.6
	0.1	89.3	73.7	87.8	71.5
	0	88.6	73.5	87.2	70.3
XNLI	0.25		70.38		70.22
	0		69.79		69.09

4.5 Discussions

4.5.1 The different roles between β and λ

In (4.21), β and λ are hyperparameters to control the importance of different terms. In general, if a task wants the data points of the same class to cluster as close as possible, we could turn up β. On the other hand, if a task wants the distance among data points from two different classes to get as far as possible, we would turn up λ. In theory, β is good for tighter clusters, while λ is good for differentiation/classification.

Unfortunately, we did not have enough computation resources to fine-tune a pair of hyper-parameters, as the training time would increase in squares. So we set $\beta = \lambda$ as a heuristic to reduce the training cost. β was the only hyperparameter we had searched. We might have gotten even better results if we could search for different λ using the development set.

[12]We use the 'en','fr','es' 'de' subset with 'zero-shot' setup without fine-tuning the embedding models on XNLI data.

In some earlier research, we would first train a T-MDE model for several epochs and then adapted the cT-MDE model on top of it. The '0.5' multiplier on the translation cost ($\mathtt{l_{mt}}$) prevents a sudden increase in the net loss that triggers the stop-training criteria when we shift from regular training to distance constraint training—none of the results in this thesis used this adaptation method.

4.5.2 Maximum input length in the test

Whereas the full-length texts were used in LASER_1 and LASER_6 to generate document embeddings, only the first 750 tokens were used to produce T-MDE's document embeddings to balance its GPU memory consumption and test accuracy. This is because the transformer takes in a fix-sized input with zero paddings.

4.5.3 pre-training data comparison

Europarl is large enough to support most NMT models. The five languages that were chosen in the test are the overlapping languages between Europarl and MLDoc. This subset of five languages is sufficient to infer the quality of knowledge transfer between language pairs; many previous publications also work on 2–5 languages [20, 7, 36, 5, 17, 19, 3, 44, 46, 44].

LASER_5$_{2019}$ vs. T-MDE:

The latest published LASER results [4] was trained on a much larger corpus. It is the combination of Europarl, United Nations, OpenSubtitles2018, Global Voices, Tanzil and Tatoeba corpora, consisting of 93 languages. It is difficult for us to obtain the exact same corpus and train our models with the 93-language corpus in LASER; training a model with the 93-language corpus would take five days using 16 V100 GPUs. Thus, we re-implemented the current LASER version according to the latest description [4] and trained it with the Europarl corpus as the baseline (LASER_1 and LASER_6).

It suffices to say that LASER_5$_{2019}$ in Table 2 enjoys an enormous training data advantage. Therefore, the comparison between it and T-MDE is not fair.

XLM-R vs. T-MDE:

Two different training corpora were tried in XLM-R. Firstly, we trained an XLM-R$_{\mathtt{europarl}}$ model with the default 'MLM+TLM' setting [27] using the Europarl corpus. Moreover, we downloaded the pre-trained XLM-R$_{\mathtt{base}}$ and XLM-R$_{\mathtt{large}}$ models and tested them with the same zero-shot setting as in LASER and T-MDE. The pre-trained XLM-R$_{\mathtt{base}}$ and XLM-R$_{\mathtt{large}}$ model was trained with the 100 language Common Crawl Corpus [27]. It is an enormously huge corpus compared to the Europarl, but it is not parallel.

We found that XLM-R$_{\mathtt{base}}$ had the best performance compared to XLM-R$_{\mathtt{europarl}}$ and XLM-R$_{\mathtt{large}}$. XLM-R$_{\mathtt{europarl}}$ is significantly worse than XLM-R$_{\mathtt{base}}$, XLM-R$_{\mathtt{large}}$ and even T-MDE_F. Unlike LASER, XLM-R has always been trained on a huge non-parallel corpus in its previous publications [26, 8]. Thus, we chose the best performing XLM-R$_{\mathtt{base}}$ as the XLM-R baseline to guarantee that the XLM-R result would not have a disadvantage due to the differences in the training data .

4.5.4 Cross-lingual accuracy on test and development set by epochs

Table 4.11 to Table 4.11 present the cross-lingual accuracy on the standard test and development set epoch by epoch. These corresponding test/validation results were for easier experiment replication. We kept model savings at 10-20th epochs and used 'Cross' as the main metric to select model and hyper-parameters.

Table 4.8: *Cross-lingual accuracy for cT-MDE by epochs*

	Dev set		Test set	
epoch	Same	Cross	Same	Cross
10	89.8	73.0	88.8	72.9
11	89.7	72.0	89.1	71.9
12	90.0	74.0	88.9	73.8
13	90.1	73.1	89.1	73.0
14	90.0	73.6	89.2	73.2
15	90.2	74.2	89.4	74.2
16	90.1	73.9	89.2	73.9
17	89.5	72.8	89.5	72.8
18	90.1	75.5	89.2	75.2
19	90.3	74.4	89.4	74.3
20	90.4	74.6	89.4	74.9

Table 4.9: *Cross-lingual accuracy for T-MDE by epochs*

	Dev set		Test set	
epoch	Same	Cross	Same	Cross
10	89.3	72.3	88.4	72.6
11	89.5	72.2	88.6	72.4
12	89.8	71.9	88.5	72.3
13	89.8	73.5	88.6	73.5
14	89.7	72.8	88.6	72.9
15	89.9	73.4	88.6	73.6
16	89.7	72.4	88.4	72.5
17	89.8	73.3	88.7	73.9
18	89.7	72.8	88.9	72.9
19	89.7	73.1	88.6	73.4
20	89.9	72.5	89.0	72.8

Table 4.10: *Cross-lingual accuracy for cLASER_6 by epochs*

	Dev set		Test set	
epoch	Same	Cross	Same	Cross
10	88.6	71.3	87.8	71.5
11	88.9	72.7	87.8	72.6
12	88.9	70.3	88.0	70.3
13	88.9	69.2	88.1	69.6
14	88.6	67.2	87.8	67.4
15	88.8	68.7	88.0	68.5
16	88.7	68.4	88.0	68.4
17	88.6	68.6	87.8	68.4
18	88.4	69.1	87.8	68.9
19	88.4	69.5	88.0	69.3
20	88.7	69.6	88.0	69.6

Table 4.11: *Cross-lingual accuracy for LASER_6 by epochs*

	Dev set		Test set	
epoch	Same	Cross	Same	Cross
10	88.9	70.3	87.8	70.1
11	89.1	69.8	87.6	69.8
12	88.5	70.3	87.2	70.3
13	88.7	68.8	87.7	68.9
14	89.0	70.2	87.7	70.1
15	88.8	69.3	87.9	69.3
16	88.8	68.7	87.7	68.7
17	88.9	69.4	87.7	69.3
18	88.7	69.9	87.8	69.9
19	88.8	69.3	87.6	69.3
20	88.8	67.7	88.0	67.6

4.6 Significant tests

Table 4.12: *Significant tests of the cross-lingual accuracy ('Cross') between different models in the 10-fold paired t-test. A result is significant if* $p < 0.01$.

Group A	Group B	p
LASER_1	**LASER_6**	0.0373
LASER_6	**cLASER_6**	<0.0001
T-MDE	**cT-MDE**	0.0008
LASER_6	**T-MDE**	0.0001
T-MDE_F	**T-MDE**	<0.0001
XLM-R	**T-MDE**	<0.0001

Apart from the standard partition used in this thesis and prior publications, MLDoc also contains a 10K large train set [44]. We combined this 10K train set with the standard test set and development set. From this combination, We drew 2K non-overlapping samples five times. For each of these 2K subsets, we rotated the train/test set once. In the end, we had 10 non-overlapping subsets, each with 1K/1K train/test partitions. We performed paired (2-tailed) t-tests on the statistical significance over these 10 new subsets. A result is significant if $p < 0.01$.

Table 4.12 further support our conclusion in the result section. All the document vectors in Group B perform significantly better than those in Group A as $p < 0.01$, except for LASER_1 and LASER_6. It proves that T-MDE's performance is significantly better than LASER and XLM-R. Distance constraint training in cT-MDE and cLASER_6 would let them have superior performance than the same model without distance constraint training (T-MDE and LASER_6). The bottleneck attention in T-MDE significantly improves performance compared to the T-MDE_F model without the bottleneck attention.

4.7 Summary

In this chapter, a transformer-based document embedding model cT-MDE is presented. cT-MDE enjoys two advantages over the LASER model. Firstly, it adopts a transformer architecture, which is more capable and faster than the traditional LSTM encoder in LASER. Moreover, the distance constraint training in the cT-MDE model also improves the cross-lingual performance significantly.

On the other hand, the main difference between the T-MDE and other transformer-based models like mBERT and XLM-R is the bottleneck attention and the multilingual machine translation objective. In a pre-training-and-fine-tuning model like XLM-R, the model fine-tuning on the downstream task can incentivize feature selection and information concentration towards the first position encoder output. However, for the document embedding task (without downstream fine-tuning), XLM-R and mBERT lack the incentive to concentrate information. After pre-training, most document-level information from the encoder would still spread across the entire sequence of encoder outputs. Both the bottleneck in T-MDE and the distance constraint training provide such incentives. Therefore cT-MDE has much better performance in the document embedding tasks. The effectiveness of the bottleneck attention design and the distance constraint training in cT-LASER are proved via experiments.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

This thesis mainly explores the deep neural network method for multilingual document embedding. In this branch of research, the main difficulty lies twofold. First, the embedding model should establish relationships between different languages so that text of similar content but in different languages can be projected in nearby space. Secondly, the embedding model should be able to select and summarize information from a sequence of variable lengths into a vector of fixed dimensions. This thesis describes a transformer-based multilingual document embedding model (T-MDE). A neural machine translation-based framework is adopted in the T-MDE model to solve the first problem. A bottleneck self-attention structure is used in the T-MDE model to solve the second problem.

Both NTDV and T-MDE are embedding models based on the neural machine-translation framework. NTDV has an independent encoder/decoder for each language, while T-MDE adopts a shared encoder-decoder structure. This research found that T-MDE architecture is superior as it has better performance and can be expanded into many languages. The performance of T-MDE and the state-of-the-art baselines like LASER and XLM-R are also compared in detail. This thesis also presents a novel distance constraint function which is applied on both T-MDE and LASER. The experiments prove that both cT-MDE and cLASER are superior to their counterpart under standard translation training objectives. All in all, cT-MDE with bottleneck attention and distance constraint training significantly improves the document embedding quality and beats the other baseline model.

The current T-MDE is examined in the two most popular cross-lingual classification tasks (MLDoc and XNLI). The current T-MDE could be expanded into more tasks in the future. Moreover, T-MDE can also be expanded into a dual-purpose pre-training-and-fine-tuning model, which is good at both document embedding tasks and other sequence-to-sequence tasks.

5.2 Future works

For a multilingual encoding model, two particular kinds of downstream tasks would take place most frequently. One is the document/text embedding task, in which the encoder would embed a text document into a single fix-sized vector. We can encode all the train/test data into document vectors in a downstream task (e.g., MLDoc) into a common vector space using the pre-trained multilingual encoding model. A simpler model can then be trained or tested on these vectors to further compare, index, and cluster them.

Another particular kind of task is sequential tagging or translation. Instead of expecting a single vector to represent the entire sentence/document, this category of tasks requires the encoding model to produce a sequence of encoder outputs to represent detailed token-by-token temporal information. The decoder part of the model can then attend to every encoder output and produce a correct tag/translation for each inputting token.

The conflict of interest between the document embedding tasks and sequential tagging tasks places a dilemma upon the current SOTA models. A natural question comes to our mind: could there be a document embedding model that can compile overall text information into a single vector, and at the same time, its encoder can also provide detailed token-by-token information to support other down-stream tasks; such as sequential tagging or translation?

Future studies could improve the TME model by devising a 'dual-band' connection between the transformer encoder and decoder. This model could be called dual-band T-MDE (dbT-MDE). dbT-MDE has both bottleneck attention and full-length attention between the encoder and decoder. The bottleneck attention would encapsulate general text information into one vector, thus suitable for document embedding and text classification. In contrast, full-length attention, which can provide token-by-token temporal information, would be suitable for sequential tagging/translation-related tasks. dbT-MDE can outperform other state-of-the-art models in cross-lingual text classification tasks and POS tagging tasks with the right balance of training objectives.

REFERENCES

[1] Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 133–142. ACM, 2016.

[2] Antonios Anastasopoulos and Graham Neubig. Should all cross-lingual embeddings speak English? *arXiv preprint arXiv:1911.03058*, 2019.

[3] Mikel Artetxe and Holger Schwenk. Margin-based parallel corpus mining with multilingual sentence embeddings. *arXiv preprint arXiv:1811.01136*, 2018.

[4] 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.

[5] Alexandre Bérard, Christophe Servan, Olivier Pietquin, and Laurent Besacier. Multivec: a multilingual and multilevel representation learning toolkit for NLP. In *The 10th edition of the Language Resources and Evaluation Conference (LREC)*, 2016.

[6] Ana Margarida de Jesus Cardoso Cachopo. *Improving methods for single-label text categorization*. PhD thesis, Universidade Técnica de Lisboa, 2007.

[7] Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861, 2014.

[8] 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.

[9] 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*, 2018.

[10] Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. Trans-gram, fast cross-lingual word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113, 2015.

[11] Andrew M. Dai, Christopher Olah, and Quoc V. Le. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*, 2015.

[12] 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.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Multilingual BERT, 2019.

[14] Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. The form is the substance: Classification of genres in text. In *Proceedings of the Workshop on Human Language Technology and Knowledge Management*, pages 1–8. Association for Computational Linguistics, 2001.

[15] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[16] Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1150–1159, 2017.

[17] Daniel C Ferreira, André FT Martins, and Mariana SC Almeida. Jointly learning to embed and predict with multiple languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2019–2028, 2016.

[18] Luanne Freund, Charles L. A. Clarke, and Elaine G Toms. Towards genre classification for IR in the workplace. In *Proceedings of the 1st International Conference on Information Interaction in Context*, pages 30–36. ACM, 2006.

[19] Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756, 2015.

[20] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributional semantics. In *Proceedings of ACL*, 2014.

[21] Jussi Karlgren and Douglass Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th Conference on Computational Linguistics*, volume 2, pages 1071–1075. Association for Computational Linguistics, 1994.

[22] Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 32–38. Association for Computational Linguistics, 1997.

[23] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015.

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

[25] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

[26] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[27] Guillaume Lample and Alexis Conneau. Pytorch original implementation of cross-lingual language model pretraining, 2020.

[28] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[29] Yong-Bae Lee and Sung Hyon Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 145–150. ACM, 2002.

[30] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[31] Wei Li and Brian Mak. Fast derivation of cross-lingual document vectors from self-attentive neural machine translation model. In *Proceedings of Interspeech*, pages 107–111, Hyderabad, India, September 2018.

[32] Wei Li and Brian Mak. Nmt-based cross-lingual document embeddings. In *Manuscript submitted for publication*, September 2020.

[33] Wei Li and Brian Mak. Transformer based multilingual document embedding model. *arXiv preprint arXiv:2008.08567*, 2020.

[34] Thang Luong, Hieu Pham, and Christopher D Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015.

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

[36] Aditya Mogadala and Achim Rettinger. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 692–702, 2016.

[37] Jaimin Mungalpara. What does it mean by bidirectional LSTM?, 2021.

[38] Christopher Olah. Understanding lstm networks, 2015.

[39] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[40] Philipp Petrenz. *Assessing approaches to genre classification*. PhD thesis, M. Sc. Thesis, School of Informatics, University of Edinburgh, 2009.

[41] Hieu Pham, Thang Luong, and Christopher Manning. Learning distributed representations for multilingual text sequences. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 88–94, 2015.

[42] Stephen E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[43] Holger Schwenk and Matthijs Douze. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*, 2017.

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

[45] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain, April 2017. Association for Computational Linguistics.

[46] Roberta A Sinoara, Jose Camacho-Collados, Rafael G Rossi, Roberto Navigli, and Solange O Rezende. Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, 163:955–971, 2019.

[47] Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. Text genre detection using common word frequencies. In *Proceedings of the 18th Conference on Computational Linguistics*, volume 2, pages 808–814. Association for Computational Linguistics, 2000.

[48] Raymond Hendy Susanto. fairseq-laser: Implementing LASER architecture using fairseq library, 2020.

[49] Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. Context gates for neural machine translation. *Transactions of the Association of Computational Linguistics*, 5(1):87–99, 2017.

[50] 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 6000–6010, 2017.

[51] Bonnie Webber. Genre distinctions for discourse in the Penn TreeBank. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2, pages 674–682. Association for Computational Linguistics, 2009.

[52] Maria Wolters and Mathias Kirsten. Exploring the use of linguistic features in domain and genre classification. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, pages 142–149. Association for Computational Linguistics, 1999.

[53] Shijie Wu and Mark Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. *arXiv preprint arXiv:1904.09077*, 2019.

[54] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNET: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.