Building LARO: Language Agnostic Sentence Embeddings from finetuned RoBERTa *

Andre Soblechero Salvado

[1] Hamburg University of Applied Science
[2] Faculty of Computer Science and Engineering
[3] Department of Computer Science
[4] Berliner Tor 7, 20099 Hamburg
[5] andre.soblecherosalvado@haw-hamburg.de

Abstract. In this paper an architecture will be introduced which consists of a new kind of autoencoder Transformer architecture which is capable of generating language agnostic sentence embeddings such as recurrent neuronal networks can. A proof of concept will show the practicality.

Keywords: Transformer · Auto-Encoder · LASER · Language Agnostic · Sentence embeddings · Context embeddings · Attention is all you need · OPUS-100 · Cross-Lingual · RoBERTa · Distilled · Cross-Lingual

1 Introduction

Natural language processing (NLP) is a branch of artificial intelligence that deals with the processing of natural language. There exist a lot of different subbranches like, eg. understanding the intent behind a statement or question by processing the question or statement by a machine or generating good translations from one language to another automatically. This paper is about building and training a machine translation based autoencoder, a machine learning model, which is capable of generating language agnostic sentence embeddings. LARO is a combination of the Transformer architecture, the in LASER use method and transfer learning.[3][7][16] These kind of embeddings, also known as context embeddings, are vectors which express the meaning of a sentence. In addition the model which will be build should be capable of extracting the meaning of a sentence completely independent of the language used. This means that for example the sentence "How are you?" in English and the sentence "Wie geht es dir?" in German should be encoded by the model to a very similar vector. Language agnostic models are important because they make it possible to train a model in one language and use it after training with a lot of other languages without having to train the model on them. This is more time-effective, helps with low-resource languages and costs less hence less data needs to be gathered/generated.

* Supported by Creative Space for Technical Innovations at the Hamburg University of Applied Sciences

The following architecture which will be introduced consists of a new kind of autoencoder Transformer architecture which is capable of generating sentence embeddings, like recurrent neuronal networks can.

2 Cross-language information retrieval

In this section a basic usage for multilingual models will be represented in order to underline their importance. Cross-language information retrieval is the name for a problem where a query of an user to for example a search engine can be in an arbitrary language and a collection of documents, which may answer the query independent from the query language, is returned.

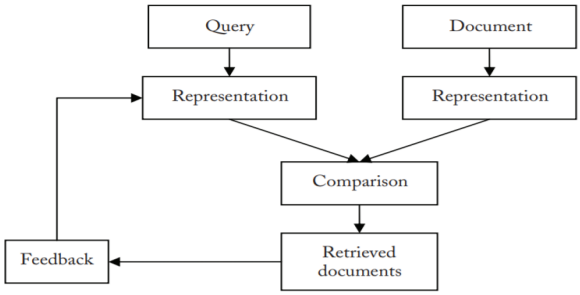

Fig. 1. Information Retrieval Procedure [1]

For this purpose documents and queries are often represented as a vector which eg. represents the meaning/content of the document. The query and document need to be mapped into the same vector space. A similarity algorithm then decides whether the representations are similar or not and thus if the respective document is returned. Later in this paper it will be explained on how to compute similarity of vectors and how to generate representations of sentences.[1]

3 Neural Machine Translation

A basic task today in machine learning is to translate from one language to another using neuronal networks. Because of the complexity of the problem and usefulness it is heavily used as benchmark and commercial products arised as well. "From a probabilistic perspective, translation is equivalent to finding a target sentence y that maximizes the conditional probability of y given a source sentence x, i.e., arg max_y $p(y|x)$."[2] The following sections will will be introducing a method which purpose is to obtain language agnostic sentence embeddings.

4 LASER Language-Agnostic Sentence Representations

LASER is an encoder which main characteristic is the ability to map sentences in 93 languages to one vector space. This can be done completely language agnostic, what describes the capability of the encoder to encode similar sentences in different languages to similar vectors. One obvious advantage is that task specific models which are trained on the output vectors of LASER are language agnostic as well. Thus in theory it is possible to train a classifier on the embeddings in one language and use this classifier in production with other languages. Considering the shortage of task specific data in deep learning, models like LASER can improve the efficiency of companies because data in less languages is required for training.[3]

4.1 Model

To train a decent language agnostic representation for sentences in form of a vector an autoencoder can be used. Furthermore it is important to take into consideration that the encoder may not have information about the input language to ensure that every input sentence is treated the same way, As a consequence only the meaning of a sentence gets encoded. The decoder on the other hand needs some information about the target language because it decodes to more than one pivot language. A pivot language is a target language in this kind of task. The architecture of LASER consists of two recurrent networks.[3]

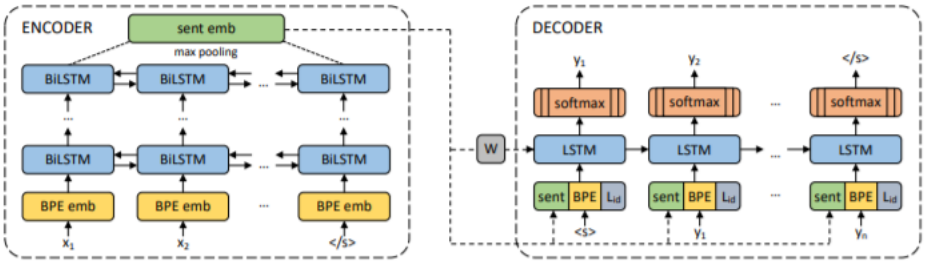

Fig. 2. LASER Architecture [3]

Encoder Each token gets a standard token embedding vector. Those token embeddings are then fed into a stacked Bidirectional LSTM with a max pooling on top of the last hidden layers to get a single vector output of dimension $d = 1024$. After training the output vector the meaning and language agnostic representation of a sentence and is often called context vector or sentence embedding.[3][28]

Decoder The decoder consists of a simple LSTM. The final token embeddings are the sum of three vectors. One vector is each time the same context vector from the encoder. The second vector is the token embedding for the current token. The first token is a start of sequence token thus the beginning of a sentence is predefined. The third embedding is a language embedding which gives the decoder information about the target language. This is of high importance because the target language should not depend on the encoded sentence and therefore on the training's-data distribution of languages. The final token embeddings are then fed into the LSTM to decode the translation. The first hidden vector of the LSTM is the context vector. Underlining that the decoder exactly uses one vector from the encoder to generate the translation. This is by design because the authors wanted that this context vector holds all information about the encoded sentence.

4.2 Data

A combination of several corpora where used to train LASER with 93 input languages. The corpora combination consists of Europarl, United Nations, OpenSubtitles2018, Global Voices, Tanzil and Tatoeba which are publicly available on the OPUS website.[5] Considering that the amount of data to train increases quadratic with the amount of target languages the authors decided to only use 2 target/pivot languages which are common, Spanish and English. Also some low-resource languages like Swahili where used as input-data.[4][5]

5 Attention is all you need

Attention is all you need marks the starting point of most today's state of the art language models. In this section some fundamental ideas of this paper will be covered intuitively. Attention is all you need introduced the Transformer model which has the ability of learning sequences without being recurrent thus it made it possible to make more efficient use of GPUs. Deep learning models in computer vision where already able to do so. Nowadays deep learning models in natural language processing can be trained on 10,000 GPUs with an batch-size of several million only thanks to the introduction of Transformers. [6][7]

5.1 Byte Pair Encoding

Today in practice there exist dictionaries which translate a from the training known token to a mathematical representation like a One-Hot-Vector or Embedding.[9] Some problems remain with all those translating systems. The model only knows the mathematical representation of tokens which are in the training dataset and there exist too many words as well as conjugations of words as that it is possible to cover all up in one dictionary.
Byte Pair Encoding was introduced as a simple data compression algorithm and is being used to create a dictionary capable of almost solving this problem. Instead of lemmatizing or stemming tokens during preprocessing, the Byte Pair encoding Algorithm tries to find common character sequences. Afterwards it adds them into the dictionary until a predefined maximum amount of tokens (sequences of bytes) in the dictionary is reached. To convert a sentence into a sequence of dictionary entries a greedy algorithm is used.[10]

Iteration	Sequence	Vocabulary
0	a b a b c a b c	{a, b, c}
1	ab ab c ab c	{a, b, c, ab}
2	ab abc abc	{a, b, c, ab, abc}
3	ababc abc	{a, b, c, ab, abc, ababc}
4	ababcabc	{a, b, c, ab, abc, ababc, ababcabc}

Fig. 3. Byte Pair Encoding example.[8]

5.2 Model

The basic form of a Transformer model consists of a encoder and a decoder. While the encoder takes a sequence and encodes it into a mathematically representation the decoder decodes to an other sequence. Recent papers proposed to only use the encoder (BERT) or decoder (GPT2) alone to train language models. [11][12]

Positional Encoding The position of a token in a sequence may be important for the meaning/sense of a sequence thus it should be included. In recurrent networks the position-information of a token is given implicit, Transformers in contrast need this information explicit because of the lack of recurrence. This objective is reached by generating a unique vector per position in sequence eg. the first token-embedding of a sequence gets always a unique vector added to it.

Encoder The encoder of a Transformer tries to encode the information of a sequence to provide a representation to the decoder. This goal is attained by feeding the sequences of vectors to several Layers where the input dimension equals output dimension. The encoder can be mathematically represented as follows:

$$E : \mathbb{R}^{nxd} \mapsto \mathbb{R}^{nxd}, \tag{1}$$

where n is the number of elements in the sequence and d a fixed dimension. Each encoder layer consists of three blocks.

1. **Multi-Head Attention** This block is the part where information from all input tokens can be combined or routed from one token to another.
2. **Add Norm** This block is against overtraining and completes the residual block [13]
3. **Feed Forward** This block consists of feed forward layers
4. **Add Norm** This block is against overtraining and completes the second residual block

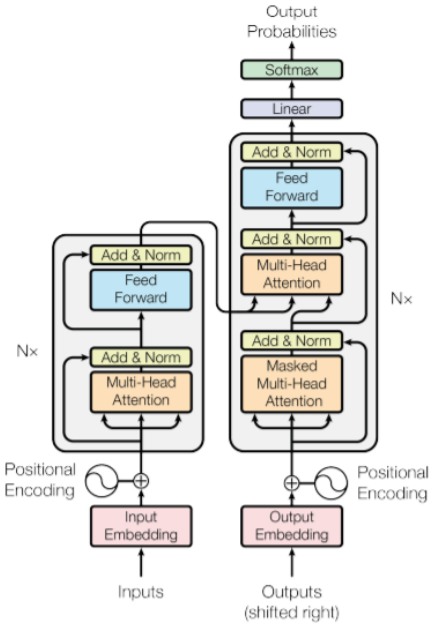

Fig. 4. Orginal Transformer visualised [7]

Decoder The Transformer decoder differs from the Transformer encoder mainly in three ways. The decoder can be mathematically represented as follows:

$$D : \mathbb{R}^{nxd} \mapsto \mathbb{R}^{nxd}, \tag{2}$$

1. **Shifted Input** The input sequence is shifted to the right. Example: the input sentence "I am dancing" would be changed to "[CLS] I am dancing" where [CLS] is the start of sequence token. A [EOS] token will be added to the target sequence which the decoder should learn to generate eg. "I am dancing [EOS]" in this case the target token of "[CLS]" would be "I".
2. **Masked Multi-Head Attention** This Attention-Block forces the decoder to learn the next token by only using the left attending tokens.
3. **Second Multi-Head Attention** This Attention-Block adds information from the encoder output.

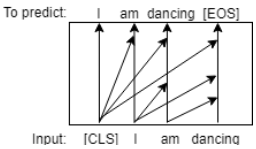

Fig. 5. This picture visualizes the mechanism of left attending Transformer decoders

5.3 Training

The original Transformer was trained on a translation task. The encoder encodes a sentence in one language and the decoder translates the encoded sentence to another language.

6 XLM-RoBERTa

XLM-RoBERTa is the successor of RoBERTa and a large multi-lingual language model. It is a Transformer-based model, technically a Transformer encoder, and was trained on 2.5TB of filtered CommonCrawl data in 100 different languages.[15]

6.1 Masked Language Model

The pretraining task was the masked language model. Its task is to predict words in a sentence which are masked or changed to a random other token. The Transformer encoder generates exactly one output-vector for each input token. The objective is to train those output-vectors to represent each input-token in context of the whole sentence. This objective is accomplished by predicting the masked/changed token with a special mapping in form of a Linear Layer from the corresponding output-vector of the encoder to a vector with the length n, where n is the size of the dictionary (250K Tokens). Finally a Softmax is applied to the vector.[14] The sentences are all treated the same way thus there are no language specific differences in the task. As an example the input "I am dancing" would be randomly masked to be "I am [MASK]", where the [MASK]-Token is a new special token. The model now should predict "dancing" by classifying the output-vector which corresponds to the [MASK]-Token.

This training results in the model approximating the meaning and syntax of all languages in the training data.

6.2 Pretraning models

There exist two pretrained models which can be used for finetuning. The first one is $XLM-RoBERTa_{Base}$ with 125M parameters, 12 layers, a hidden dimension d=768 and 8 heads for each multi-head attention layer. The second one is $XLM-RoBERTa_{Large}$ with 355M parameters, 24 layers, a hidden dimension d=1027 and 16 heads for each multi-head attention layer.

7 Introducing LARO

Most Transformer models and their variations are not providing any point in their architecture where the model learns a representation of the whole sentence but rather representations of tokens/words in context of a sentence. The original Transformer encoder outputs a matrix with variable number of columns which represent the encoded sentence. Therefore they are not feasible as a representation in eg. a vector space. What if it would be possible to use the Transformers as an autoencoder which encodes a sequence to a vector with fixed dimension

and then decodes from the vector to an other sequence? With such an architecture a model could be trained with the same objective as LASER, but with the state of the art power of Transformers. Further a pretrained encoder like XLM-RoBERTa could be used to profit from transfer learning too.[16]

7.1 Model

A basic transformer model capable of being trained the same way as LASER can be easily created. To do so first each encoder input gets a [CLS] Token added to it like in BERT.[11] The german sentence "Ich tanze" would result to "[CLS] ich tanze". This special token, like other special tokens eg. [EOS], [PAD], [MASK] or [UNK], is in the dictionary thus will not be splitted by the byte pair encoding.

Since each input-token has exactly one corresponding encoder output-vector the to the [CLS] token corresponding encoder output-vector will be trained to be the encoding of the meaning. This objective is reached by overwriting each column of the encoder output matrix with the first column of it which is the to the [CLS]-Token corresponding output-vector.

$$E_{new} = (A_1^T|...|A_1^T)^T \tag{3}$$

where A_1 denotes the first output column of the standard encoder E. This matrix is then fed the same way as before to the decoder. The training task would remain the same as for LASER, translate from many languages to some pivot languages.

The model can be extended by several new techniques, too, such as using lower dimensional token embeddings with transformations via Lineare Layers to higher ones or only use the parameters of one Layer and repeat this Layer several times to reduce the size of the model like in ALBERT.[17] Furthermore a pretrained encoder like XLM-RoBERTa as encoder and/or decoder could be used to reduce the number of epochs needed. This maybe improve the general capability of the model because of transfer learning. Otherwise a pretrained model could restrict the Transformer model to the same languages, or similar ones, as used in the pretraining process of them because the dictionary built with Byte Pair Encoding is language specific. Furthermore the decoder could receive information about the target language like in LASER.

8 Experimental Setup

In this section all practical aspects will be covered. Starting with data selection to the used platform, training and used frameworks.

8.1 Frameworks

As underlying script language Python 3.8 will be used. Above of Python the Pytorch framework will be utilized to take heavy advantage of the power of the four NVIDIA Quadro P6000 while performing math. Furthermore the Huggingface Transformers framework will be used to be flexible with precomputed tokenizers and pretrained models eg. RoBERTa-XLM. The LARO model will derive from the EncoderDecoder class in the Framework and the trainer will also be from Huggingface Transformers. In addition NVIDIA APEX will be utilized to speed up the training by using mixed precision operations. Tensorboard is utilized to monitor the training and GPU-VRAM usage. [18][19]

8.2 Data

As training-data the OPUS-100 Corpus will be used.[20] The OPUS-100 is a multilingual parallel corpus which is randomly generated from all available dataset in the OPUS collection. The corpus is English-centric in other words in every translation either the source or target sentence is in English. "OPUS-100 contains approximately 55 million sentence pairs. Of the 99 language pairs, 44 have 1M sentence pairs of training data, 73 have at least 100k, and 95 have at least 10k." [5]
Although only 79 languages of the dataset where used during the XLM-RoBERTa training 100 languages will be used during training. The hundredth language is going to be English considering that English should also be used as input language. Because LASER has shown that two pivot languages are sufficient we will only utilize English as pivot language.(See the two tables on page 18) Because only one pivot language is used the language embeddings for the decoder like in LASER are unnecessary. The dataset is already split into train/dev/eval sets. Additionally the Corpus needs to be extended by English to English sentence pairs because without them the encoder may not be capable of encoding English sentences effectively. The English sentences of the English to Spanish sentence pairs will be used for this purpose. [5]

8.3 Pipeline

A training is technically defined as a job. A job is a running docker container. The platform will have four NVIDIA Quadro P6000. Each of the GPUs have 24GB of GDDR5 VRAM, which is important to notice because batch size matters and thus VRAM size. The platform is providing 36 Threads and has 252 GB of RAM.

Pretraining This model will first be trained on translating from many languages to a pivot language like LASER. In contrast to LASER, LARO will be trained on a different dataset.

Training on specific tasks After training the model, the language agnostic encoder can be used for three different tasks:

1. **Finetuning** The resulting encoder could be finetuned for tasks like Sentiment-Analysis. This usage is similar to transfer-learning and requires most of the time a Linear Layer on top of the context-vector.
2. **Freezed training** The encoder parameters do not change while training a classifier on the context-vectors. This way of using the encoder makes it possible to maintain language agnostic context-vectors while training on eg. unilingual data. This implies that the classifier can be used independent from the input language.
3. **Mathematical comparability of sentences using cosine distance** The encoder makes it possible to project data into a vector space. Equal data may have similar vectors thus the encoder makes it possible to determine whether two encoded sentences are similar or not. This implies the effective usage of nearest neighbour search algorithms on texts which are eg. heavily used in recommendation systems.

9 Evaluation

Metrices are needed for two purposes. The first one is comparison to other multilanguage models. The second one is to find a sufficient convergence criterion to stop training the model before overtraining. For the reason that the model can be used in multiple ways there are multiple metrics.

As convergence criteria metrics for generative tasks like BLEU can be used to check whether the model is still converging or if progress stopped.
After training similarity metrics on a test-dataset can be used to evaluate if the encoder is generating sufficient sentence representations.
Furthermore standard metrics for benchmarks need to be used to compare the model with other models, but since the training is not about getting new state of the art scores and more about using data language agnostic as well as optimizing for low-resource languages, most benchmarks are negligible.

9.1 BLEU: a Method for Automatic Evaluation of Machine Translation

BLEU is a modified precision used to measure the quality of translations. The measurement was introduced because most scores until BLEU where not taking into account that translations could be syntactical variational. Furthermore BLEU respects not only the words which appear in the generated translation but also n-grams, which are fragments of the sentence where each fragment consists of n Tokens.[21]

Modified Precision Lets take the bad machine translation output "the the the the the the the" and the reference "The cat is on the mat". The normal precision would be $\frac{7}{7}$ because seven tokens of the machine translation output appear in the reference. This problem shows that a modified precision is required to evaluate machine translations. The modified precision score would be $\frac{2}{7}$ because "the" appears 2 times in the reference sentence and the length of the machine translations output sentence is 7.

N-Gram BLEU The machine translation output sentence "The cat the cat on the mat" would have a modified score of 7/7 because all tokens of the sentence appear in reference. This shows us another problem with the modified precision when only using tokens/unigrams. To solve this problem the N-Gram BLEU was introduced. The N-Gram BLEU takes N-Grams instead of unigrams/tokens.
"The cat The cat on the mat" in 2-Grams {The cat, cat The, The cat, cat on, on the, the mat}
"The cat is on the mat" in 2-Grams {The cat, cat is, is on, on the, the mat}
The modified precision would be $\frac{3}{6} = 0.5$ because 3 2-Grams of the reference sentence appear in the 6 2-Grams of the machine translation output. Formally this can be described as follows:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C'} Count(n-gram')} \tag{4}$$

Details If the machine translation output is much shorter that the reference than another problem appears. Taking last reference and the new machine translation output "The cat". The modified precision for 2-Grams would be $\frac{1}{1}$ because one 2-Gram out of one appears in the reference. This problem is solved by a brevity penalty:

$$BP = \begin{cases} 1, & \text{if c} > \text{r} \\ e^{1-r/c}, & \text{if c} \leq \text{r} \end{cases}$$

BLEU is then defined as follows:

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log p_n) \tag{5}$$

Usually N=4 is being used.

9.2 Cosine Similarity

The cosine similarity has the purpose to measure the similarity of two vectors. Unlike the measurement is the distance of two vectors the cosine similarity score measures the angle between two vectors. This is quite useful because distances in vector spaces of very high dimensions can vary quite heavy thus these scores can be useless. [22] When two vectors are more equal than other two than the resulting number is higher. Because the objective of this paper is to build a model which encodes sentences into vectors which somehow represent the meaning of a sentence this measurement can be quiet useful. For example it is possible to check if similar sentences in different languages are similar or unequal sentences less similar. This would give us the possibility to measure the effectiveness of a model. The cosine similarity is for two vectors A and B defined as follows:

$$\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

10 Benchmarks

NLP benchmarks are crucial for measuring general progress in the areas. There exists a variety of benchmarks which all try to measure different aspects of NLP. Because LAROs focus will be to generate language agnostic sentence embeddings only some benchmarks are qualified. Some of those benchmarks will be described in this section.

10.1 Cross-lingual Natural Language Inference (XNLI)

The XNLI corpus can be used to evaluate Cross-lingual encoders. The corpus builds around the question on how to "make predictions in any language at test time, when we only have English training data for that task?"[23][24] The XNLI Corpus is a crowd sourced collection of 5,000 test and 2,500 dev pair for the MultiNLI corpus. Those pairs are then translated into 14 languages thus it contains 112.5k annotated sentences pairs. Low resource languages like Swahili are included too. This benchmark can evaluate the effectiveness of LARO in cross lingual tasks as well as on how well it handles input sentences in languages which he was not trained on.

The evaluation procedure would be as follows:

1. Each sentence of each pair will be encoded by LARO.
2. The cosine similarity will be used to compute the similarity of each pair
3. When the similarity is below 0.5 then the pair count as false negative (FN) else true positive (TP).
4. The final score would be $S = \frac{TP}{TP+FN}$

By following this procedure it is possible to analyze up to what point the sentences are encoded language agnostic.

10.2 Tatoeba: similarity search

The Tatoeba benchmark evaluates in contrast to XNLI 112 languages. The corpus contains about 1,000 English-aligned sentence pairs for each language. For each language the 1,000 English-aligned sentence pairs do not necessary intersect. The task is to find the nearest neighbour for each sentence in the respective other language by using cosine similarity. Micro average is used as score. [3]

11 Proof of concept

As proof of concept the model has been implemented and trained on a machine translation task from Spanish to English for five epochs on four P6000 GPUs, this procedure took about 37 hours. Instead of working with the original XLM-RoBERTa model, two pretrained distilled RoBERTa models where used to decrease training time and VRAM usage because the basic XLM-RoBERTa is quite huge and the standard attention mechanism very expensive. The distilled RoBERTa model is a Transformer model with less Layers and Attention Heads and was trained to reproduce the outputs of the original RoBERTa model. [19][25] The encoder as well as the decoder where pretrained to reduce training duration. Furthermore every data-point which has more than 300 Tokens was ignored. Instead of the BLEU score after each epoch the dev loss was calculated to ensure that the model learns something. The training is followed by a evaluation of the evaluation data using the BLEU score. The BLEU score was not used between every epoch because decoding every translation autoregressivly is very time consuming. Additionally the cosine distance of several example sentences where calculated to ensure that the sentence embedding is capable of expressing the meaning of a sentence. The example sentences are several versions of "how are you" in Spanish and several sentences which are not equal to "how are you" in Spanish. The several versions of "how are you" in Spanish had all a very similar cosine similarity to each other. (> 0.8) The other

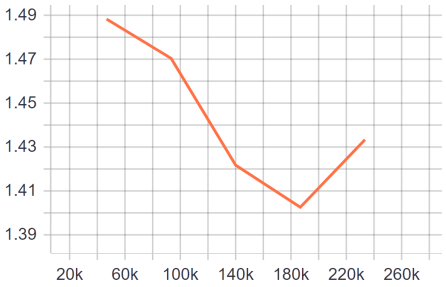

Fig. 6. DEV loss after each epoch

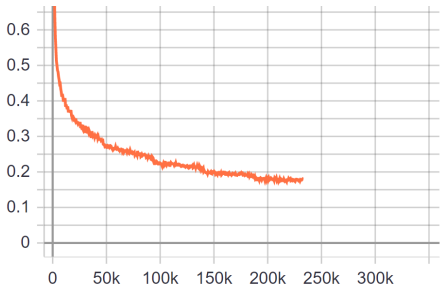

Fig. 7. Loss every 500 steps

sentences did not have a high similarity distance to any other sentence above (< 0.8). The BLEU Score was between 19 and 20, what indicates that the model produces sentences which can be described like following: "The gist is clear, but has significant grammatical errors".[27] This is not surprising considering that the model was only trained for 5 Epochs. This tests and evaluations lead me to the conclusion that it would be possible to train language agnostic sentence embeddings with LARO.[26]

12 Outlook and limitations

Finally the training of LARO is limited by time considering that 5 epochs of training on the Spanish-English dataset took about 37 hours and that the next objective is to train LARO on the whole dataset. A naive prognosis of the time which the final training would take considering that eight GPUs instead of four will be used, what halves the training duration, and that the Spanish-English dataset consists of 2 million data-points and the whole dataset of 50 million data-points would be about $37h/(2+2)*50 = 462.5h$ or about 19 days. Because of this number the distilled RoBERTa model or a more efficient model will be used in future experiments instead of the XLM-RoBERTa model. Furthermore Google AI has recently released LABSE, a BERT model which is capable of generating language agnostic sentence embeddings. It was trained on more data than the OPUS-100 dataset consists of and it is currently state of the art in some cross lingual benchmarks. For future experiments it may be helpful to train LARO at first on a small cross-lingual sub dataset of OPUS-100. Summarizing the final LARO model or LABSE will be a good basic instrument to perform Cross-language information retrieval tasks, benchmarks and recommendation systems in general.

References

1. Jian-Yun Nie: Cross-Language Information Retrieval. Morgan & Claypool, University of Montreal (2010) http://www.iro.umontreal.ca/ nie/IFT6255/Books/CLIR.pdf
2. Dzmitry Bahdanau and Kyunghyun Cho and Yoshua Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473. 2016
3. Mikel Artetxe and Holger Schwenk: Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. arXiv:1812.10464. 2019
4. Vishrav Chaudhary and Yuqing Tang and Francisco Guzmán and Holger Schwenk and Philipp Koehn: Low-Resource Corpus Filtering using Multilingual Sentence Embeddings. arXiv:1906.08885. 2019
5. Jörg Tiedemann: Parallel Data, Tools and Interfaces in OPUS. In: Nicoletta Calzolari (Conference Chair) and Khalid Choukri and Thierry Declerck and Mehmet Ugur Dogan and Bente Maegaard and Joseph Mariani and Jan Odijk and Stelios Piperidis, Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). European Language Resources Association (ELRA), Istanbul, Turkey (2012). https://doi.org/978-2-9517408-7-7 http://opus.nlpl.eu/
6. Tom B. Brown and Benjamin Mann and Nick Ryder and Melanie Subbiah and Jared Kaplan and Prafulla Dhariwal and Arvind Neelakantan and Pranav Shyam and Girish Sastry and Amanda Askell and Sandhini Agarwal and Ariel HerBERT-Voss and Gretchen Krueger and Tom Henighan and Rewon Child and Aditya Ramesh and Daniel M. Ziegler and Jeffrey Wu and Clemens Winter and Christopher Hesse and Mark Chen and Eric Sigler and Mateusz Litwin and Scott Gray and Benjamin Chess and Jack Clark and Christopher Berner and Sam McCandlish and Alec Radford and Ilya Sutskever and Dario Amodei: Language Models are Few-Shot Learners.arXiv:2005.14165. 2020
7. Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin: Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin. arXiv:1706.03762. 2017
8. Zhan, Junlang: "An effective feature representation of web log data by leveraging byte pair encoding and TF-IDF." Proceedings of the ACM Turing Celebration Conference - China (2019): n. pag.
9. Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff: Distributed Representations of Words and Phrases and their Compositionality. Distributed Representations of Words and Phrases and their Compositionality. C. J. C. Burges and L. Bottou and M. Welling and Z. Ghahramani and K. Q. Weinberger. Curran Associates, Inc. 2013: 3111–3119
10. Zhan, Junlang, X. Liao, Y. Bao, L. Gan, Zhiwen Tan, Mengxue Zhang, Ruan He and J. Lu: An effective feature representation of web log data by leveraging byte pair encoding and TF-IDF. Proceedings of the ACM Turing Celebration Conference - China (2019)
11. Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. 2019
12. Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya: Language Models are Unsupervised Multitask Learners. 2019
13. Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun: Deep Residual Learning for Image Recognition. arXiv:1512.03385. 2015

14. Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall: Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. arXiv:1811.03378v1. 2018
15. Unsupervised Cross-lingual Representation Learning at Scale: Alexis Conneau and Kartikay Khandelwal and Naman Goyal and Vishrav Chaudhary and Guillaume Wenzek and Francisco Guzmán and Edouard Grave and Myle Ott and Luke Zettlemoyer and Veselin Stoyanov. arXiv:1911.02116. 2020
16. Fuzhen Zhuang and Zhiyuan Qi and Keyu Duan and Dongbo Xi and Yongchun Zhu and Hengshu Zhu and Hui Xiong and Qing He: A Comprehensive Survey on Transfer Learning. arXiv:1911.02685. 2020
17. Zhenzhong Lan and Mingda Chen and Sebastian Goodman and Kevin Gimpel and Piyush Sharma and Radu Soricut: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942. 2020
18. Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.. 2019 :8024–8035 http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
19. Thomas Wolf and Lysandre Debut and Victor Sanh and Julien Chaumond and Clement Delangue and Anthony Moi and Pierric Cistac and Tim Rault and Rémi Louf and Morgan Funtowicz and Joe Davison and Sam Shleifer and Patrick von Platen and Clara Ma and Yacine Jernite and Julien Plu and Canwen Xu and Teven Le Scao and Sylvain Gugger and Mariama Drame and Quentin Lhoest and Alexander M. Rush: HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771. 2020
20. Biao Zhang and Philip Williams and Ivan Titov and Rico Sennrich: Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation. arXiv:2004.11867. 2020 http://opus.nlpl.eu/opus-100.php
21. Papineni, Kishore Roukos, Salim Ward, Todd Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.
22. Rahutomo, Faisal Kitasuka, Teruaki Aritsugi, Masayoshi. (2012). Semantic Cosine Similarity.
23. https://github.com/facebookresearch/XNLI
24. Alexis Conneau and Guillaume Lample and Ruty Rinott and Adina Williams and Samuel R. Bowman and Holger Schwenk and Veselin Stoyanov: XNLI: Evaluating Cross-lingual Sentence Representations. arXiv:1809.05053. 2018
25. Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108. 2020
26. LARO. 18.10.2020 https://github.com/AndreSoble/TransformerAutoencoder
27. Google Evaluation. 18.10.2020 https://cloud.google.com/translate/automl/docs/evaluate : :text=BLEU%20(BiLingual%20Evaluation%20Understudy)%20is,of%20high%20quality%20reference%20translations.
28. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 15, 1997), 1735–1780. DOI:https://doi.org/10.1162/neco.1997.9.8.1735

13 Attachement

ISO code	en
af	279512
am	93027
an	6961
ar	1004000
as	142479
az	266089
be	71312
bg	1004000
bn	1004000
br	157447
bs	1004000
ca	1004000
cs	1004000
cy	293521
da	1004000
de	1004000
dz	624
el	1004000
eo	682212
es	2008000
et	2008000
eu	2008000
fa	2008000
fi	2008000
fr	2008000
fy	116684
ga	587048
gd	39054
gl	1038688
gu	644612
ha	203966
he	2008000
hi	1076638
hr	2008000
hu	2008000
hy	14118
id	2008000
ig	44202
is	2008000
it	2008000
ja	2008000
ka	762612
kk	167854
km	230966
kn	32744
ko	2008000
ku	297688
ky	62430
li	59070
lt	2008000

ISO code	en
lv	2008000
mg	1189542
mk	2008000
ml	1653492
mn	8588
mr	62014
ms	2008000
mt	2008000
my	57188
nb	293812
ne	820762
nl	2008000
nn	980110
no	2008000
oc	79582
or	33816
pa	222592
pl	2008000
ps	166254
pt	2008000
ro	2008000
ru	2008000
rw	355646
se	79814
sh	542422
si	1966218
sk	2008000
sl	2008000
sq	2008000
sr	2008000
sv	2008000
ta	462028
te	136704
tg	395764
th	2008000
tk	33628
tr	2008000
tt	209686
ug	152340
uk	2008000
ur	1515826
uz	354314
vi	2008000
wa	216992
xh	887342
yi	38020
yo	20750
zh	2008000
zu	85232
en	2008000