LARO: Language Agnostic Sentence Representations 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. This paper introduces LARO, an as a dual encoder finetuned XLM-RoBERTa model that is capable of generating language agnostic sentence embeddings. The training of this model is based on LaBSE, a BERT model finetuned using contrastive learning. This paper shows, that RoBERTa can be effectively trained to produce language agnostic sentence embeddings with contrastive learning while using fewer data and training time than LaBSE. [6][21]

Keywords: RoBERTa · XLM-RoBERTa · BERT · language agnostic · sentence representations · dual encoder · contrastive loss · tatoeba · OPUS-100.

1 Introduction

Natural language processing is a branch of artificial intelligence. A variety of problems exist that are solvable by using natural language processing e.g. converting spoken language to texts, translating texts from one language to another. One of those problems is how to encode a sequence of words to a representative vector. In recent years, several attempts have been made to find a way to effectively and efficiently encode sentences to vector representations. One way is to train a model which encodes a sentence into a vector and decodes the input sentences from this vector again. This kind of model is also called auto-encoder. To train a model which is capable of producing language agnostic sentence representations, researchers started to use translations. They trained an encoder to encode a sentence in one language to a vector and a decoder to decode from this vector to the translation in another language.[14] Several languages need to be used for this objective, because just encoding sentences in one language would not result in a language agnostic model. This way of training has one drawback. After training the encoder and decoder, the decoder was not used any more. As a result the resources for training the decoder were somehow wasted. With the

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

introduction of contrastive learning a process where only encoders were trained this problem disappeared.[8] This paper introduces LARO, an as a dual encoder finetuned XLM-RoBERTa model that is capable of generating language agnostic sentence embeddings. The training of this model is based on LaBSE, a BERT model finetuned using contrastive learning. This paper shows, that RoBERTa can be effectively trained to produce language agnostic sentence embeddings with contrastive learning while using fewer data and training time than LaBSE. [6][21]

2 Language Agnostic Sentence Representations

Sentence Representations in form of vectors are heavily used today because they make it possible to easily and quickly rank similarities between texts as well as finetune a classifier on top of the vector sentence representation.[14][21] Sentence Representations are vectors which try to represent the sentence, thus similar sentences have similar vectors. Similar vectors are vectors which have a high cosine similarity. Language agnostic sentence representations are an extension of the sentence representations. A sentence representation is language agnostic, when a sentence in one language has a similar representation like the sentence translated into another language.

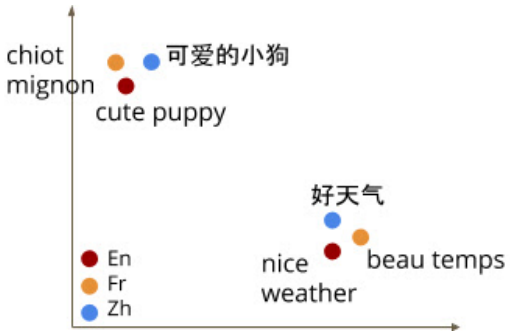

Fig. 1. Example for language agnostic embeddings[21]

3 Usage of Language Agnostic Sentence Representations

Language Agnostic Sentence Representations can be used in a variety of ways. To underline the importance of this kind of sentence embeddings, this section provides several examples of how language agnostic sentence representations/embeddings can be applied.[14]

3.1 Searching similar sentences in multiple languages

Using language agnostic sentence representations for cosine similarity search allows finding similar sentences independent of the language. In order to find a

similar sentence for the input sentence, the only thing to do is to compute the cosine similarity between the input sentence embedding and all other embeddings. The output would be a cosine similarity between the input sentence and all other sentences, which can be sorted in descending order by similarity and thus ranked. There exists several algorithms that make it possible to efficiently use this procedure.

3.2 Training a sentences classifiers

There are several ways of fine-tuning a language agnostic model. The first one is to train a classifier on the language agnostic sentence embeddings. As a consequence a possibility of qualifying the model on one language as well as using in production with multiple languages exists. The advantage is, that training data is not required for every language, because the embeddings are language agnostic. The second way would be to add a classifier head to the encoder and fine-tune the whole encoder including the classifier head. This procedure does not necessary have the same advantages as in the before explained way of training a classifier head with a freezed encoder.

3.3 Language agnostic clustering

Since similar sentences have a close vector the sentence representations can be easily used for clustering sentences and e.g. topic-modelling.

4 XLM-RoBERTa

XLM-Roberta is a successor of RoBERTa and BERT. [6][5][10] It has the same architecture as BERT thus it is a slightly modified transformer encoder.[18]

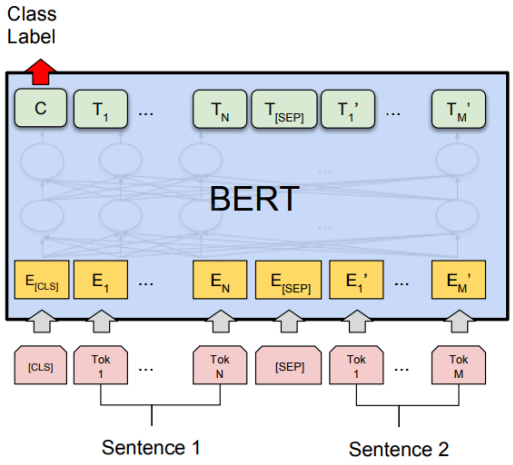

Fig. 2. BERT, the basic architecture of XLM-RoBERTa.[15]

XLM-Roberta is a Masked Language Model, which was trained to predict masked words from the input sentence. This model is special owing to the fact that it was trained on one hundred different languages rather than simply one. The training resulted in the model approximating the meaning and syntax of all languages in the training data. The dataset which XLM-Roberta was trained on consists of 2.5TB CommonCrawl Data. The architecture of XLM-Roberta is a

Transformer encoder, enabling to work with sequences without being recurrent thus allowing us to train the model efficient on multiple GPUs. This pretrained model will be finetuned to obtain language agnostic sentence representations.

The specific model which will be used is the $XLMRoBERT_{Base}$ with 125 million parameters, 12 layers, a hidden dimension d=768 and 8 heads for each multi-head attention layer.

5 Training

Initially the XLM-RoBERTa model should be finetuned on a machine translation task facilitate generating language agnostic sentence embeddings. To speed up the training the task changed to contrastive learning using a dual encoder with shared parameters. The just mentioned concepts will be explained in the following sections.

5.1 OPUS-100

The dataset used for training is the OPUS-100 dataset.[17][23][2] The OPUS-100 dataset is an English centric dataset that consists of sentence pairs with the source of target language in English and the corresponding language is one out of 99 other languages. It consists of about 50 million sentence pairs. A sentence pair in which one sentence is longer than 128 tokens is ignored. All sentences are lists of tokens which are truncated to a length of 64 tokens. The data is split into two datasets. 80% of the data will be used as train data a 20% of the data will be used as test data to determine the progress during the training.

5.2 Dual Encoder with shared parameters

While the usual approach of training a model that is capable of generating sentence embeddings was training an encoder-decoder model with a bottleneck layer between the encoder and decoder, which would represent the encoded sentence, LARO is trained as a dual encoder.[21] Instead of using a bottleneck layer while training on a machine translation task we will encode sentences in the source and in the target language to obtain embeddings for each sentence of a sentence pair. The task is to train the model to generate embeddings which are as similar as possible for sentences in the source language and in the target language. Shared parameters mean that we do not train two encoders to encode each sentence, but instead one encoder is used to encode the sentence in the source language and the same one to encode the sentence in the target language before computing the contrastive loss between the outputs of both encoders. The advantage of this method is, that the encoder understands both, the target and the source language, and additionally VRAM is being saved by only loading parameters of one model into the GPU Memory. In the next sections when speaking about the output of the first or second encoder it is referred to the encoded sentence in the source language or the encoded sentence of the target language. Both, the first and second encoder represent the same model.

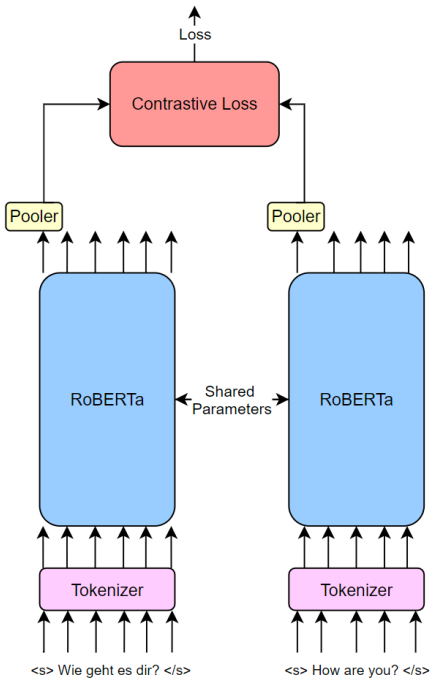

Fig. 3. LARO Training: One RoBERTa model encodes a sentence in the source language and in the target language. Each input token produces an output token. The embedding corresponding to the $< s >$ token for each sentence will be fed into a pooler layer to obtain a sentence embedding. During training these two embeddings are used to compute the contrastive loss.

5.3 Contrastive Learning

The goal is to train a dual encoder that is capable of generating language agnostic sentence embeddings. The simplest way of computing a loss for such a dual encoder would be (1 - cosineSim(s, t)) for each embedding s of the source language and each embedding t of the target language. If the embeddings are similar then the loss will be small and if the embeddings are not similar, then the loss will be higher but always below one. The problem of this approach is that the model could collapse during the training. A collapse in this case would mean that the model produces for each sentence the same embedding thus not meaningful representations of sentences but something that minimizes the loss. The model collapse can be prevented by using a contrastive loss, which not only takes into account which embeddings should be similar but also which embeddings should not be similar.[8][22][21][20] To use the contrastive loss the assumption is that in a batch the embedding produced by the first encoder has only one corresponding embedding produced by the second encoder. In conclusion the similarity of those two should be maximized.

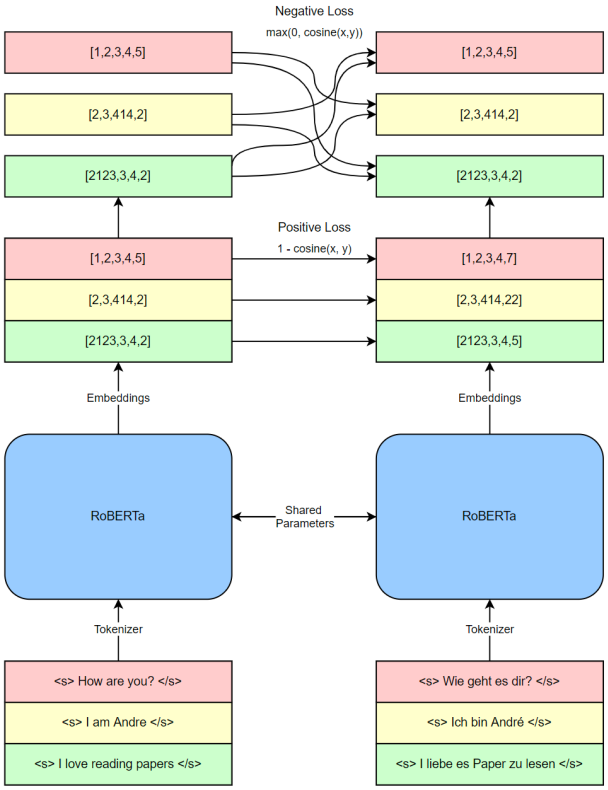

Fig. 4. LARO - contrastive loss: Visulalization of the contrastive loss ignoring details about the XLM-RoBERTa model. Originally a 768 dimensional sentence embedding is used.

The similarity of a specific embedding produced by the first encoder is not supposed to be similar to any other embedding produced by the second encoder except the corresponding one thus the cosine similarity of those embedding pairs should be minimized.

The cosine similarity between two vectors is defined as follows:

$$cosine(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

The positive loss which shows how similar a sentence embedding and the corresponding embedding of its translation of the sentence is defined as follows:

$$loss_{positive} = \frac{1}{N}\sum_{i=1}^{N}(1 - cosine(x_i, y_i)) \tag{2}$$

where N is the batch size, x_i the i_{th} embedding in the batch produced by the first encoder and y_i the i_{th} embedding produced by the second encoder.

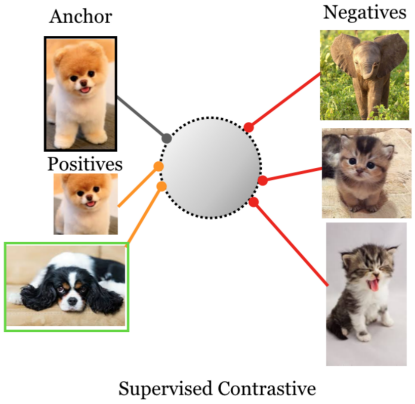

Fig. 5. Computer vision contrastive loss. Dogs should have similar embeddings and other animals different ones.[8] Two dogs would represent a sentence pair which should have similar embeddings while other animals represent the not corresponding translation should have different embeddings.

The negative loss which shows how well a model differentiates embeddings from other embeddings which should not be equal:

$$loss_{negative} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N}\sum_{j=1, i\neq j}^{N} max(0, cosine(x_i, y_j)) \tag{3}$$

where N is the batch size, x_i the i_{th} embedding in the batch produced by the first encoder and y_j the j_{th} embedding produced by the second encoder. For each embedding produced by the first encoder a loss was calculated to not corresponding embeddings produced by the second encoder. Finally a mean is used to obtain a general negative loss.

The overall loss which will be used for backpropagation is:

$$loss = loss_{positive} + loss_{negative} \tag{4}$$

The first loss indicates the model´s performance encoding similar sentences into similar embeddings and the second loss indicates how the embeddings for not similar sentences differ from each other.

5.4 Frameworks

A variety of frameworks were used to finetune XLM-RoBERTa. In this section the most important frameworks will be explained briefly.

PyTorch PyTorch is the underlying deep learning framework that makes it possible to train highly customizable machine learning models using GPU acceleration frameworks like Nvidia CUDA without the need to perfectly understand low level code.[12]

Huggingface Transformers Huggingface Transformers is a framework that supports a variety of Transformers models and simplifies the use of them.
Furthermore the trainer of this framework was used while a XLM-RoBERTa class was modified to be able to use it as a dual encoder model.[19]

FairScale FairScale is a framework that adds "high performance and large scale training on one or multiple machines/nodes" to PyTorch.[11] It uses a novel training procedure "Optimizer state sharding (ZeRO)" introduced by Microsoft researchers.[13] Usually when training on multiple GPUs, each GPUs needs to hold all parameters of the model in VRAM. With this novel protocol it is possible to shard all parameters over multiple GPUs and only load parameters from other shards/GPUs, when they are needed during the forward or backward pass. Applying this technique larger models, which until recently needed a lot of hardware resources, now can be trained on less GPUs. It is also possible to increase the batch size drastically. The batch size in this case is quite important because of the way the contrastive loss is being computed. A larger batch size means more negative examples and thus a better training.

5.5 Setup

For the training one machine with 164 GB of RAM and five P6000 GPUs, each with 24 GB of VRAM, was used. The code is publicly available on GitHub.[16]

5.6 Finetuning

The finetuning took about one hundred hours for two epochs with a batch size of 200 sentence pairs per GPU thus a general batch size of 1000 sentence pairs. A sentence pair is a source sentence and a target sentence in another language.

6 Tatoeba Benchmark

The Tatoeba benchmark published by Facebook Research was used to determine the overall performance of the model in Cross-lingual Text Retrieval.[3] The objective of the model is to find the nearest neighbor translation for a given sentence

using the cosine distance. The Benchmark consists of sentences in 112 languages and their corresponding English translation thus it is an English centric benchmark. Each language has up to 1000 translations to English. The sentences of different languages may intersect. LASER and LaBSE were used for comparison due to the fact that they are also able to generate language agnostic sentence representations.[21][14] Additionally, the benchmark was split into three different benchmarks. Two of those are a subset of the general benchmark and one is the complete benchmark.

- The first benchmark consists of 15 languages. It covers the most basic languages and writing systems like English, German, France, Wuu-Chinese, Japanese, Arabic, Russian.
- The second benchmark consists of 37 languages. It is an extension of the first benchmark and adds languages like Indonesian and Hebrew.
- The third benchmark consists of all 112 languages. The languages can be found in the cited GitHub Repository. [1]

6.1 LaBSE: Language-Agnostic BERT Sentence Embedding by Google AI

LaBSE is one of the two models used for comparison. LaBSE is a finetuned pretrained multilingual BERT model. It has almost the same architecture as XLM-RoBERTa but was pretrained with less multilingual data than XLM-RoBERTa.[6][21] The fine-tuning was a combination of the masked language model training, where a classifier is trained to predict masked tokens in the input sentence and afterwards on a contrastive learning task. They used 17 billion monolingual sentences and 6 billion bilingual sentence pairs in 109 languages to train the model on those two tasks. LaBSE is currently one of the best models in several benchmarks including the Tatoeba benchmark.

6.2 LASER Language-Agnostic SEntence Representations

LASER was in contrast to LARO and LaBSE trained as a machine translation model to enable obtaining language agnostic sentence representations.[14] LASER was trained on 223 million English or Spanish centric sentence pairs. The Toetaba benchmark was introduced together with LASER. The encoder, used for generating the sentence embedding, is a bidirectional LSTM.[7] The decoder is a simple LSTM which receives the sentence embedding and tries to decode the translation from the input sentence of this embedding. The decoder has not been published.

6.3 Evaluation procedure

The evaluation algorithm used to obtain a score for the benchmark is the following[3]:

1. Set true positives and false positives to null.
2. For each language l
   (a) For each sentence s in the language l
      i. Compute the cosine similarity of the sentence s to all English translations of all sentences from l.
      ii. If the translation with the highest cosine similarity is the gold standard translation of the sentence s then increase true positives by one and else increase false positives by one.
3. Compute the score $\frac{tp}{tp+fp}$ of the model.

6.4 Results

	15 languages	37 languages	All languages
LaBSE	0.95	0.95	0.84
LASER	0.83	0.87	0.69
LARO	0.75	0.75	0.59

LARO performs poorly in contrast to LASER and LaBSE. This may be the result of using fewer data and only training the model for two epochs. In fact considering that LARO was only trained on 50 million sentence pairs in contrast to LASERs 200 million and LaBSEs billions of sentence pairs, the model performs quite good. Additionally, LaBSE used a batch size of one million during contrastive learning and LARO only a batch size of one thousand what decreases the overall performance of a model trained using contrastive learning drastically.[21]
More details on how the model performed for each language can be seen in the section "Attachment - Results"

7 Outlook and limitations

In general the objective of training a model capable of generating language agnostic embeddings has been fulfilled. Unfortunately the model is not state-of-the-art, but it has the ability of becoming state-of-the-art, when trained longer, with more data and with a much bigger batch size. The batch size depends directly on the amount of GPUs/VRAM available thus much more GPUs are needed to actually compete against LaBSE. The performance of LASER could easily be beaten using more data and with more epochs.

Considering that a lot of researchers are moving away from vanilla transformers because of their memory inefficiency in the self attention layer, future experiments should be done using newer architectures. [4][9] Novel architectures like "FNet: Mixing Tokens with Fourier Transforms"[9], which replaces the self attention layer of a transformer based model with two Fourier transformations could be used instead.

References

1. Github laser repository. https://github.com/facebookresearch/LASER/tree/master/data/tatoeba/v1, accessed: 2021-06-06
2. Aharoni, R., Johnson, M., Firat, O.: Massively multilingual neural machine translation. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 3874–3884. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1388, https://www.aclweb.org/anthology/N19-1388
3. Artetxe, M., Schwenk, H.: Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond (2019)
4. Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., Weller, A.: Rethinking attention with performers (2021)
5. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale (2020)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
8. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning (2021)
9. Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontanon, S.: Fnet: Mixing tokens with fourier transforms (2021)
10. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (2019)

11. Mandeep Baines, Shruti Bhosale, V.C.N.G.S.G.M.O.B.L.V.L.M.R.S.S.A.S.M.X.: Fairscale: A general purpose modular pytorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale (2021)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
13. Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: Zero: Memory optimizations toward training trillion parameter models (2020)
14. Schwenk, H., Douze, M.: Learning joint multilingual sentence representations with neural machine translation (2017)
15. Seth, Y.: Bert explained – a list of frequently asked questions. https://yashuseth.blog/2019/06/12/bert-explained-faqs-understand-bert-working/, accessed: 2021-06-06
16. Soblechero, A.: Github laro repository. https://github.com/AndreSoble/laro, accessed: 2021-06-06
17. Tiedemann, J.: Parallel data, tools and interfaces in OPUS. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). pp. 2214–2218. European Language Resources Association (ELRA), Istanbul, Turkey (May 2012), http://www.lrec-conf.org/proceedings/lrec2012/pdf/463*Paper.pdf*
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)
19. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Huggingface's transformers: State-of-the-art natural language processing (2020)
20. Wu, Z., Wang, S., Gu, J., Khabsa, M., Sun, F., Ma, H.: Clear: Contrastive learning for sentence representation (2020)
21. Yang, Y., Feng, F.: Language-agnostic bert sentence embedding. https://ai.googleblog.com/2020/08/language-agnostic-bert-sentence.html, accessed: 2021-06-06
22. Yang, Y., Hernandez Abrego, G., Yuan, S., Guo, M., Shen, Q., Cer, D., Sung, Y.h., Strope, B., Kurzweil, R.: Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 5370–5378. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/746, https://doi.org/10.24963/ijcai.2019/746
23. Zhang, B., Williams, P., Titov, I., Sennrich, R.: Improving massively multilingual neural machine translation and zero-shot translation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1628–1639. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.148, https://www.aclweb.org/anthology/2020.acl-main.148

8 Attachment - Training-data

ISO 639-1	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 639-1	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

9 Attachment - Results

Performance for each language x to English for each evaluated model. Language codes are ISO 639-2 standard.

	LASER	LaBSE	LARO
afr	0.902000	0.971000	0.818000
amh	0.446429	0.934524	0.577381
ang	0.343284	0.671642	0.246269
ara	0.922000	0.911000	0.613000
arq	0.375412	0.497256	0.110867
arz	0.689727	0.800839	0.375262
ast	0.842520	0.929134	0.559055
awa	0.354978	0.770563	0.199134
aze	0.761000	0.961000	0.730000
bel	0.635000	0.961000	0.782000
ben	0.899000	0.911000	0.712000
ber	0.663000	0.107000	0.010000
bos	0.968927	0.960452	0.909605
bre	0.132000	0.186000	0.202000
bul	0.946000	0.959000	0.822000
cat	0.955000	0.964000	0.851000
cbk	0.782000	0.834000	0.337000
ceb	0.128333	0.691667	0.076667
ces	0.962000	0.975000	0.868000
cha	0.197080	0.364964	0.131387
cmn	0.903000	0.963000	0.766000
cor	0.065000	0.134000	0.027000
csb	0.411067	0.588933	0.177866
cym	0.060870	0.939130	0.401739
dan	0.960000	0.965000	0.895000
deu	0.990000	0.994000	0.922000
dsb	0.446764	0.705637	0.277662
dtp	0.057000	0.128000	0.035000

	LASER	LaBSE	LARO
ell	0.016000	0.965000	0.814000
epo	0.973000	0.981000	0.844000
est	0.968000	0.980000	0.853000
eus	0.943000	0.954000	0.750000
fao	0.717557	0.908397	0.496183
fin	0.963000	0.970000	0.830000
fra	0.955000	0.961000	0.836000
fry	0.543353	0.890173	0.485549
gla	0.044632	0.891435	0.173703
gle	0.062000	0.949000	0.378000
glg	0.954000	0.969000	0.837000
gsw	0.470085	0.504274	0.162393
heb	0.917000	0.924000	0.668000
hin	0.942000	0.979000	0.807000
hrv	0.972000	0.979000	0.863000
hsb	0.575569	0.699793	0.329193
hun	0.961000	0.970000	0.813000
hye	0.400270	0.946092	0.535040
ido	0.826000	0.898000	0.437000
ile	0.853000	0.855000	0.506000
ina	0.946000	0.953000	0.652000
ind	0.948000	0.956000	0.815000
isl	0.956000	0.964000	0.838000
ita	0.953000	0.953000	0.820000
jav	0.258537	0.853659	0.302439
jpn	0.694000	0.964000	0.654000
kab	0.609000	0.059000	0.016000
kat	0.396783	0.957105	0.719839

	LASER	LaBSE	LARO		LASER	LaBSE	LARO
kaz	0.198261	0.911304	0.638261	rus	0.951000	0.953000	0.837000
khm	0.181440	0.849030	0.509695	slk	0.969000	0.974000	0.882000
kor	0.316000	0.941000	0.667000	slv	0.955043	0.964763	0.846902
kur	0.197561	0.873171	0.209756	spa	0.981000	0.981000	0.881000
kzj	0.084000	0.146000	0.018000	sqi	0.982000	0.979000	0.891000
lat	0.584000	0.797000	0.257000	srp	0.957000	0.966000	0.834000
lfn	0.641000	0.697000	0.344000	swe	0.964000	0.965000	0.857000
lit	0.959000	0.973000	0.837000	swg	0.491071	0.633929	0.250000
lvs	0.955000	0.967000	0.799000	swh	0.538462	0.894872	0.248718
mal	0.959243	0.989811	0.880640	tam	0.680782	0.902280	0.618893
mar	0.909000	0.950000	0.783000	tat	0.280000	0.873000	0.072000
max	0.517606	0.732394	0.257042	tel	0.816239	0.982906	0.773504
mhr	0.123000	0.192000	0.033000	tgl	0.522000	0.980000	0.351000
mkd	0.948000	0.949000	0.743000	tha	0.950730	0.970803	0.815693
mon	0.104545	0.961364	0.640909	tuk	0.246305	0.798030	0.108374
nds	0.814000	0.792000	0.292000	tur	0.977000	0.982000	0.832000
nld	0.969000	0.975000	0.855000	tzl	0.423077	0.615385	0.240385
nno	0.866000	0.960000	0.767000	uig	0.401000	0.934000	0.566000
nob	0.987000	0.990000	0.905000	ukr	0.942000	0.953000	0.835000
nov	0.669261	0.774319	0.365759	urd	0.800000	0.960000	0.679000
oci	0.582000	0.684000	0.301000	uzb	0.163551	0.864486	0.282710
orv	0.317365	0.487425	0.200000	vie	0.966000	0.978000	0.804000
pam	0.068000	0.137000	0.014000	war	0.156000	0.655000	0.042000
pes	0.930000	0.963000	0.799000	wuu	0.743000	0.882000	0.372000
pms	0.459048	0.636190	0.215238	xho	0.091549	0.908451	0.359155
pol	0.980000	0.979000	0.862000	yid	0.066038	0.908019	0.437500
por	0.954000	0.957000	0.881000	yue	0.847000	0.925000	0.520000
ron	0.975000	0.981000	0.874000	zsm	0.966000	0.972000	0.853000