Low-Resource Machine Translation

Andrea Cavallo, Lorenzo Scarciglia, Cristina Tortia

Deep Natural Language Processing
Politecnico di Torino

February 23rd, 2022

Table of Contents

Table of Contents

Problem Statement

Issue: Usually, Machine Translation relies on encoder-decoder architectures trained end-to-end. However, this approach performs poorly on *low-resource language pairs*.

Possible solutions:

- multilingual models
- large-scale pretraining on different tasks of multilingual models (e.g. mBART)
- train model on a rich language pair and finetune it on low-resource language pair (transfer learning)

Table of Contents

Original Paper

Dabre, Fujita, and Chu propose **Multilingual finetuning**:

- pretrain on high-resource language pair
- finetune using sentences from high-resource and different low-resource languages (**mixed**)
- finetune on one low-resource language pair (**pure**)

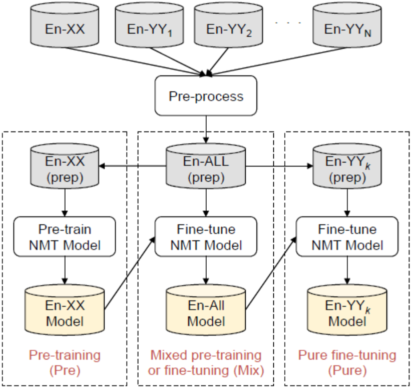


Figure: Finetuning strategy

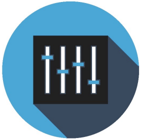
Table of Contents

Our implementation - Comparison

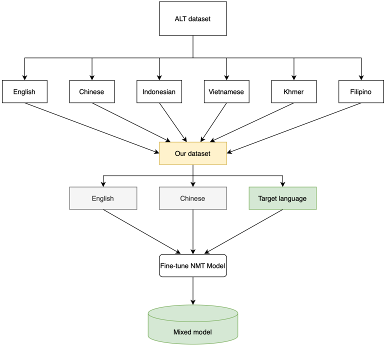
	Original method	Our method
Model	tensor2tensor: open-source implementation of the transformer model	MarianMT: transformer-based encoder-decoder architecture
Tokenizer	Moses for English, KyotoMorph and JUMAN for Chinese and Japanese, raw data for low-resource languages	Marian Tokenizer + mBART-50 encodings
Source language pair	English-Chinese English-Japanese	English-Chinese
Dataset	ALT 7 target languages	ALT 4 target languages: Vietnamese, Indonesian, Khmer, Filipino

Our implementation - Finetuning strategies

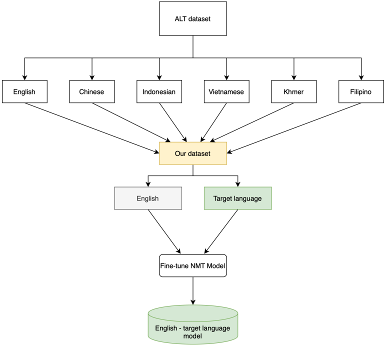
- **Pure**: finetune just on low-resource language pair (10 epochs)
- **Mixed**: finetune on a mix of one high-resource language pair (English-Chinese) and one low-resource language pair (5 epochs)
- **Mixed + Pure**: perform 5 epochs of pure finetuning after 5 epochs of mixed finetuning



Our implementation - Data selection



(a) Mixed step



(b) Pure step

Figure: Sketch of the dataset generation procedure for the mixed and pure finetuning process

Our implementation - Data preprocessing

- Add special token <*2zz*> in front of sentences to identify target language

Our implementation - Data preprocessing

- Add special token $<2zz>$ in front of sentences to identify target language
- Add to Marian tokenizer the tokens for target languages derived from mBART tokenizer

Our implementation - Data preprocessing

- Add special token <*2zz*> in front of sentences to identify target language
- Add to Marian tokenizer the tokens for target languages derived from mBART tokenizer
- Tokenize English input sentences with newly instantiated Marian tokenizer
- Tokenize target language input sentences with extended Marian tokenizer

Our implementation - Data preprocessing

- Add special token <*2zz*> in front of sentences to identify target language
- Add to Marian tokenizer the tokens for target languages derived from mBART tokenizer
- Tokenize English input sentences with newly instantiated Marian tokenizer
- Tokenize target language input sentences with extended Marian tokenizer
- Truncate sentences according to selected maximum length

Our implementation - Data preprocessing

- Add special token <*2zz*> in front of sentences to identify target language
- Add to Marian tokenizer the tokens for target languages derived from mBART tokenizer
- Tokenize English input sentences with newly instantiated Marian tokenizer
- Tokenize target language input sentences with extended Marian tokenizer
- Truncate sentences according to selected maximum length

Languages	**Input length**	**Cut-off sentences (%)**	
		English	Target language
Vietnamese	64	3.4	8.1
Indonesian	128	0.1	0.1
Filipino	128	0.1	0.3
Khmer	128	0.1	0.5

Table: Input length and cut-off sentences for the different languages

Our implementation - Framework

- Google Colab
- Hugging Face Trainer API
- Training
  - `10 epochs`
  - `learning rate = 2e-4`
  - `batch size = 16`
  - `weight decay = 0.01`

Our implementation - Results (1)

BLEU scores on test set

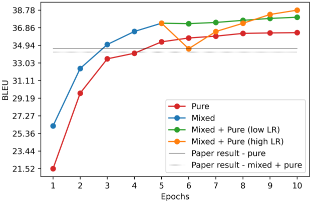
Languages	BLEU scores			
	Pure	Mixed	Mixed + Pure	Paper[1]
Vietnamese	36.56	38.07	38.9	34.22
Indonesian	37.27	37.35	37.74	25.62
Filipino	28.3	29.4	29.12	26.61
Khmer	25.06	32.65	33.32	27.49

Table: BLEU score for different finetuned models

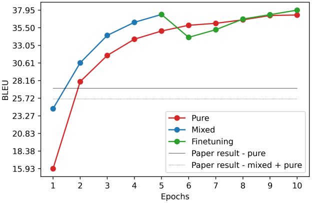
- Outperformed paper results for all languages
- Mixed always better than pure
- Sometimes mixed+pure achieves further improvements

[1] Configuration #4

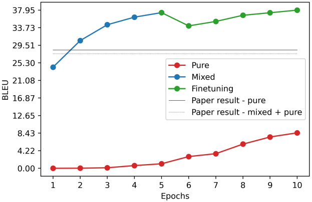
Our implementation - Results (2)



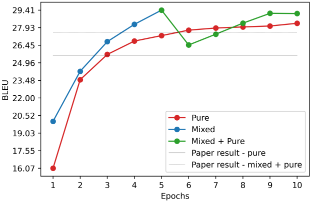
(a) English-Vietnamese



(b) English-Indonesian



(c) English-Khmer



(d) English-Filipino

Figure: BLEU score during training for different English-target language pairs.

Impact of learning rate

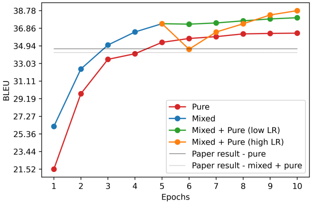


Figure: BLEU during training for Vietnamese

To avoid BLEU drop when starting pure, two learning rates for pure finetuning (2e−5 and 2e−4):

- Smaller LR avoid BLEU drop
- Higher LR achieves better performances in the end (so it is used in the other experiments)

Our implementation - Results (4)

Training for more epochs

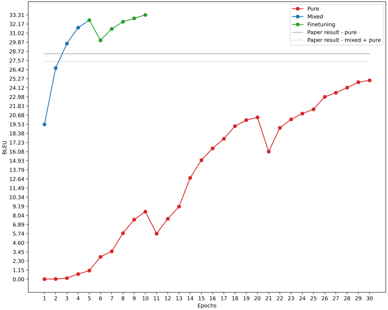


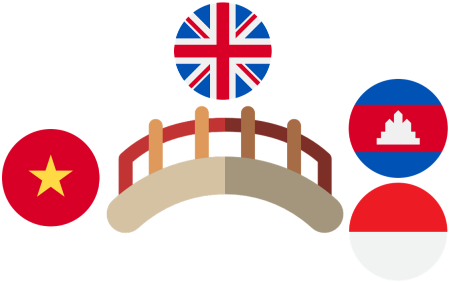
Figure: BLEU during training for Khmer

- Khmer needs more epochs to achieve satisfactory results on pure
- Mixed immediately performs much better

Table of Contents

Extension I - Idea

- Low-resource - Low-resource translator
  - English as *bridge language*
- **Steps:**
  - train a model to perform Vietnamese-English translation
  - use the previously finetuned models to translate from English to another low-resource language



Extension I - Results

Source language	Target language	BLEU score
Vietnamese	English	33.35
Vietnamese	Indonesian	23.00
Vietnamese	Khmer	24.13

Table: BLEU score for translations among low-resource language pairs

- Performances are satisfactory
- No baseline to compare results

Table of Contents

Extension II - Idea

- **Dataset:** WikiMatrix
  - automatically align sentences from Wikipedia articles
  - 1620 language pairs
- **Language pair:** English - Kazakh
- **Starting models:**
  - English - Turkish
  - English - Russian

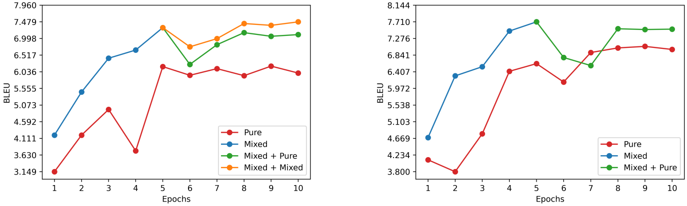
Extension II - Results (1)

Initial model	BLEU scores		
	Pure	Mixed	Mixed + Pure
En-Tk	6.93	7.33	7.26
En-Ru	6.18	7.31	7.11

Table: BLEU score for En-Kk with different initial models

- Lower performances with respect to ALT
- Mixed still performs better than pure

Extension II - Results (2)



(a) English-Russian (b) English-Turkish

Figure: BLEU score during training for English-Kazakh with different pre-trained models

- Mixed better than pure
- Mixed + mixed also better than mixed + pure

Table of Contents

Conclusions

- **Multilingual finetuning** greatly surpasses pure finetuning

[2] Translate from one low resource language to English

Conclusions

- **Multilingual finetuning** greatly surpasses pure finetuning
- **Adding tokens** from a pretrained tokenizer allows to deal with the low-resource languages
  - Although you need to add another tokenizer to tokenize English sentences

[2] Translate from one low resource language to English

Conclusions

- **Multilingual finetuning** greatly surpasses pure finetuning
- **Adding tokens** from a pretrained tokenizer allows to deal with the low-resource languages
    - Although you need to add another tokenizer to tokenize English sentences
- **This approach also works well on**:
    - The opposite direction[2]
    - Other low resource languages from another parallel corpus

[2] Translate from one low resource language to English

Conclusions - Example of usage

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
# Download the pretrained model for English-Vietnamese available on the hub
model = AutoModelForSeq2SeqLM.from_pretrained("CLAck/en-vi")

tokenizer = AutoTokenizer.from_pretrained("CLAck/en-vi")
# Download a tokenizer that can tokenize English since the model Tokenizer doesn't know anymore how to do it
# We used the one coming from the initial model
# This tokenizer is used to tokenize the input sentence
tokenizer_en = AutoTokenizer.from_pretrained('Helsinki-NLP/opus-mt-en-zh')
# These special tokens are needed to reproduce the original tokenizer
tokenizer_en.add_tokens(["<2zh>", "<2vi>"], special_tokens=True)

sentence = "The cat is on the table"
# This token is needed to identify the target language
input_sentence = "<2vi> " + sentence
translated = model.generate(**tokenizer_en(input_sentence, return_tensors="pt", padding=True))
output_sentence = [tokenizer.decode(t, skip_special_tokens=True) for t in translated]

print('en:', sentence) # -> "en: the cat is on the table"
print('vi:', output_sentence) # -> "vi: Con mè o đang trên bàn."
```

Figure: GitHub repository:
https://github.com/andrea-cavallo-98/Low-resource-Machine-Translation

Conclusions - Model Hub

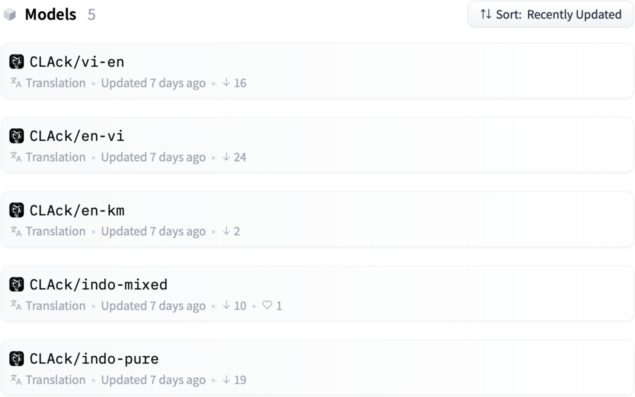


Figure: Some of the models were uploaded on the Hugging Face models hub. They are available on https://huggingface.co/CLAck

HUGGING FACE

Politecnico di Torino

Conclusions

Thank you!