Idea/Goal: Unsupervised way to create dictionaries for low-resource language varieties.

Data
Data
fast_align
Alignments
MUSE
Synthetically created Dictionary
OpenNMT

? Improving Word Translation via Two-Stage Contrastive Learning

? Handling morphological features of dialects.

Example Output (word vector for the German "auf"):

```
[3.7160760e-03  9.8595268e-04  1.2535693e-03 -4.6180072e-03 -1.6646691e-03  2.6355174e-03
-1.9086715e-03  1.8761938e-03 -2.5620037e-03  7.4709172e-04 -6.8516075e-04  1.7379265e-03
 8.2080549e-04 -8.2831604e-05  1.3117041e-04 -8.1630358e-05 -3.5634693e-03  1.1626842e-03
 5.5191561e-04 -6.1425767e-03 -5.3140690e-04 -3.9802465e-04 -1.2250873e-03  6.8854127e-04
-2.0702968e-03  2.4453271e-03  3.5143844e-03 -1.2637241e-03 -1.3017138e-03  1.0678902e-03
-2.1267894e-03 -1.3999820e-03 -1.0113821e-03  5.6023046e-06 -1.7908153e-03  1.9548502e-04
-2.9054503e-03  1.5841710e-03 -4.3602297e-03 -2.0499572e-03  1.7644017e-04  1.9481324e-03
-3.6898530e-03  1.9832905e-03  2.5702843e-03  1.1929306e-03  1.8659927e-03 -1.2305015e-03
-3.6735437e-03  1.2999651e-03 -3.6822129e-03  1.0840441e-03  2.1708327e-05 -7.2017021e-04
 2.6750427e-03  2.1874509e-03  2.0030404e-03 -2.3773254e-03  8.1840513e-04  2.7687040e-03
-1.7367934e-03 -1.1503287e-03  1.6837085e-03  3.6429542e-03 -2.1345709e-03 -2.7334681e-03
-2.9286132e-03  1.7842485e-04 -9.5358602e-04  1.5936659e-03 -1.3632205e-03  9.0761448e-04
 1.4652534e-03 -4.0345103e-03  2.4099380e-03 -1.3646146e-03 -3.8518133e-03 -4.2414963e-03
-4.6926154e-05 -5.5025001e-05  1.8696323e-03  1.0118287e-03  1.9117359e-04  3.1511887e-04
-9.2808454e-04 -1.5991109e-03 -5.4876314e-04  1.3769942e-03 -2.2713500e-03  2.2788078e-03
 3.5310598e-04  2.1700491e-03 -3.4564147e-03  1.0314359e-03 -1.3736184e-04 -2.8736801e-03
                -1.9837085e-03 -3.7012931e-03 -2.9395248e-03 -1.1259456e-03]
```

Example Output:

Example Output:
0-0 1-1 2-2 3-3 4-4 5-5 6-6

Example Output:

Prepare directories for experiment → Get data for experiment → Prepare data for experiment → Setup fast_align → Create Alignments → Setup fasttext → Create Embeddings → Setup MUSE → Create Cross-Lingual Word Embeddings → Create Dictionaries

wheel
NumPy
PyTorch
Faiss

! MUSE requires word embeddings as input

Data Raw → Data Clean → fast_align → Alignments → fasttext → Word Embeddings / Vectors → MUSE → Cross-Lingual Word Embeddings → Synthetically created Dictionary

Clean Text (Sentence per Line)

Central Kurdish

HRL Standard Variety	SL.txt Suleimani (Babanî)
LRL	HW.txt Hewleri (Hewlêrî | Erbil)
	MH.txt Mehabad (Mukriyanî)
	SN.txt Sine (Sanandaî)

Input Format for fast_align

Must be tokenized.

Aligned parallel sentences.

Each line:
"Source sent" ||| "Target sent"

Example Input:

doch jetzt ist der Held gefallen . ||| but now the hero has fallen .

fasttest takes "text" as input...
How to incorporate the fastalign alignments?

Currently: Skipping fast_align

fastText provides two models for computing word representations: skipgram and cbow

model.bin

is a text file containing the word vectors, one per line.

model.vec

is a binary file containing the parameters of the model along with the dictionary and all hyper parameters.

When loading embeddings, the model can load:

- PyTorch binary files previously generated by MUSE (.pth files)
- fastText binary files previously generated by fastText (.bin files)
- text files (text file with one word embedding per line)

The two first options are very fast and can load 1 million embeddings in a few seconds, while loading text files can take a while.

German, English, French, Ukrainian

test_mono.deu
test_mono.eng
test_mono.fra
test_mono.ukr

→ test_deu.eng, test_deu.fra, test_deu.ukr, test_eng.fra, test_eng.ukr, test_fra.ukr

(Elliott et al., 2016)
Title: Multi30K: Multilingual English-German Image Descriptions
URL: https://arxiv.org/abs/1605.00459

(Saichyshyna et al., 2023)
Title: Extension Multi30K: Multimodal Dataset for Integrated Vision and Language Research in Ukrainian
URL: https://aclanthology.org/2023.unlp-1.7

Word Representation Learning

UTF-8 encoded text.

By default the word vectors will take into account character n-grams from 3 to 6 characters.

Obtain Word Vectors for Out-Of-Vocabulary Words

using the above model and queries containing words for which you want to compute vector

The previously trained model can be used to compute word vectors for out-of-vocabulary words.

Obtain Cross-Lingual Word Embeddings

Supervised MUSE

iterative Procrustes (CPU|GPU)

learning a mapping between the source and the target space

Uses Bilingual Dictionary or Identical Character Strings.

using a train bilingual dictionary (or identical character strings as anchor points), learn a mapping from the source to the target space using (iterative) Procrustes alignment.

Unsupervised MUSE

Uses no parallel data.

learning a mapping using adversarial training and iterative Procrustes refinement

adversarial training and refinement (CPU|GPU)

without any parallel data or anchor point, learn a mapping from the source to the target space using adversarial training and (iterative) Procrustes refinement.

CBOW ("Continuous-Bag-Of-Words")

The cbow model predicts the target word according to its context. The context is represented as a bag of the words contained in a fixed size window around the target word.

SKIPGRAM

The skipgram model learns to predict a target word thanks to a nearby word.

EXAMPLE

given the sentence
'Poets have been mysteriously silent on the subject of cheese'
and the target word '*silent*'.

A skipgram model tries to predict the target using a random close-by word, like '*subject' or '*mysteriously*'.*

The cbow model takes all the words in a surrounding window, like {been, mysteriously, on, the},
and uses the sum of their vectors to predict the target.