Reproducing FastText Language Identification

Including Kazakh Language Support

Ruslan Khissamiyev

Macquarie University

October 11, 2024

Outline

Introduction

In this presentation, we describe the reproduction of the language identification task using fastText, including the addition of multiple languages such as Kazakh. This work involves:

- Downloading and processing data from various sources.
- Preparing data for training with fastText.
- Improving the model with hyperparameter tuning.
- Observing how adding more data improves model performance.
- Comparing results with the official fastText models.
- Highlighting the efficiency of using Tatoeba data over other sources.

Downloading Initial Data

We began by downloading a sample of the English Wikipedia dump using the Wikimedia API:

```
# Create a working directory
!mkdir fasttext_language_id
%cd fasttext_language_id

# Download a sample of the English Wikipedia dump
!wget https://dumps.wikimedia.org/enwiki/latest/
    enwiki-latest-pages-articles1.xml-p1p41242.bz2
    -O enwiki_sample.xml.bz2
```

Note: Downloading full Wikipedia dumps for all languages would take approximately 72 hours and over 300 GB of storage, which is not efficient for our purposes.

Processing Wikipedia Data

We used `WikiExtractor` to extract and clean text from the Wikipedia dump:

```
# Install WikiExtractor
!git clone https://github.com/attardi/
    wikiextractor.git

# Extract text from the Wikipedia dump
%cd wikiextractor
!python3 -m wikiextractor.WikiExtractor ../
    enwiki_sample.xml.bz2 -o ../extracted -b 100M
    --processes 4
```

After extraction, we combined all text files and added labels:

```
# Combine all text files into one
%cd ../extracted
!find . -name 'wiki_*' -exec cat {} + > ../en_text
    .txt

# Add labels to each line
```

Cleaning and Preparing Data

We cleaned the text data using a custom script $\texttt{clean}_t ext.py$:

```
# Clean the text data
!python3 clean_text.py en_text.txt
    final_cleaned_text.txt
```

Then we shuffled and split the data into training and testing sets:

```
# Shuffle the data
!shuf final_cleaned_text.txt >
    shuffled_cleaned_text.txt

# Split the data
total_lines=$(wc -l < shuffled_cleaned_text.txt)
train_lines=$(echo "$total_lines * 0.8 / 1" | bc)
test_lines=$(echo "$total_lines - $train_lines" |
    bc)

!head -n $train_lines shuffled_cleaned_text.txt >
    train.txt
```

Building and Testing fastText

We downloaded and built the fastText library:

```
# Downloading fastText
!wget https://github.com/facebookresearch/fastText
    /archive/v0.9.2.zip
!unzip v0.9.2.zip

# Building fastText
%cd fastText-0.9.2
!make

# Testing if fastText is working
!./fasttext
```

Training the Initial Model

We trained the initial language detection model on the English data:

```
# Training the model
!./fasttext supervised -input train.txt -output
    langdetect -dim 16
```

Output:

```
Read 7M words
Number of words:  377,291
Number of labels: 1
Progress: 100.0%  words/sec/thread: 3,271,998  lr: 0.000000
    avg.loss: 0.000000  ETA: 0h 0m 0s
```

Testing the model:

```
# Testing the model
!./fasttext test langdetect.bin test.txt
```

Results:

```
N       1480
P@1     1.0
R@1     1.0
```

Improving the Initial Model

We applied hyperparameter tuning to improve the model:

- Increased epochs to 15.
- Set learning rate to 1.0.
- Used hierarchical softmax loss function.
- Used character n-grams (minn=2, maxn=4).
- Included word n-grams (wordNgrams=2).

```
# Retraining with improvements
!./fasttext supervised \
-input train.txt \
-output langdetect \
-dim 16 \
-epoch 15 \
-lr 1.0 \
-loss hs \
-minn 2 -maxn 4 \
-wordNgrams 2
```

Testing the improved model:

Processing OSCAR Data

We added data from the OSCAR dataset to include multiple languages:

- Languages included: English, Chinese, Spanish, Arabic, French, Russian, Portuguese, German, Japanese, Hindi, and Kazakh.
- Due to the large size of OSCAR data (over 2 GB and 11 hours to process), we limited the data to 100 MB per language for efficiency.

```
from datasets import load_dataset
import re, os

languages = ["en", "zh", "es", "ar", "fr", "ru", "
    pt", "de", "ja", "hi", "kk"]

for lang in languages:
    # Load the dataset
    dataset = load_dataset("oscar", f"
        unshuffled_deduplicated_{lang}", split='
        train', streaming=True)
    # Process and clean the data
    # Limit the size to 100MB per language
    # Save cleaned data to files
```

Combining and Shuffling OSCAR Data

We combined data from all languages and shuffled:

```
# Combine all language data
!cat cleaned_*_data.txt > all_languages_text.txt

# Shuffle the combined data
!shuf all_languages_text.txt >
    shuffled_all_languages_text.txt
```

We then split the data:

```
# Split the data
total_lines=$(wc -l < shuffled_all_languages_text.
    txt)
train_lines=$(echo "$total_lines * 0.8 / 1" | bc)
test_lines=$(echo "$total_lines - $train_lines" |
    bc)

!head -n $train_lines shuffled_all_languages_text.
    txt > train.txt
!tail -n $test_lines shuffled_all_languages_text.
```

Training with OSCAR Data

We trained the model on the enhanced dataset:

```
# Training the model
!./fasttext supervised -input train.txt -output
    langdetect -dim 16
```

Results:

```
N       39736
P@1     0.96
R@1     0.96
```

Applying hyperparameter tuning:

```
# Retraining with improvements
!./fasttext supervised \
-input train.txt \
-output langdetect \
-dim 16 \
-epoch 15 \
-lr 1.0 \
-loss hs \
```

Incorporating Tatoeba Data

We added data from the Tatoeba dataset to further enhance the model:

- Tatoeba data contains millions of sentences in various languages.
- Downloading and processing Tatoeba data took only 5 minutes and required around 100 MB of storage.
- This is significantly more efficient compared to Wikipedia dumps (72 hours and 300 GB) and OSCAR data (11 hours and 2 GB).

```
# Downloading Tatoeba sentences
!wget http://downloads.tatoeba.org/exports/
    sentences.tar.bz2
!bunzip2 sentences.tar.bz2
!tar xvf sentences.tar

# Preparing the data
!awk -F "\t" '{print "__label__"$2" "$3}'
    sentences.csv | shuf > all_tatoeba.txt

# Adding Tatoeba data to previous data
!cat all_tatoeba.txt >> all_languages_text.txt
```

Shuffling and Splitting Data

We shuffled and split the combined data:

```
# Shuffle and split the data
!shuf all_languages_text.txt >
    shuffled_all_languages_text.txt

total_lines=$(wc -l < shuffled_all_languages_text.
    txt)
train_lines=$(echo "$total_lines * 0.8 / 1" | bc)
test_lines=$(echo "$total_lines - $train_lines" |
    bc)

!head -n $train_lines shuffled_all_languages_text.
    txt > train.txt
!tail -n $test_lines shuffled_all_languages_text.
    txt > test.txt

# Verify the split
!echo "Train set: $(wc -l < train.txt) lines"
!echo "Test set: $(wc -l < test.txt) lines"
```

Training with Tatoeba Data

We trained the model on the augmented dataset:

```
# Training the model
!./fasttext supervised -input train.txt -output
    langdetect -dim 16
```

Results:

```
N       2,500,834
P@1     0.946
R@1     0.946
```

Applying hyperparameter tuning:

```
# Retraining with improvements
!./fasttext supervised \
-input train.txt \
-output langdetect \
-dim 16 \
-epoch 15 \
-lr 1.0 \
-loss hs \
```

Comparing Results with Official Models

We downloaded the pre-trained official fastText language identification models:

```
# Downloading the pre-trained models
!wget https://dl.fbaipublicfiles.com/fasttext/
    supervised-models/lid.176.bin
!wget https://dl.fbaipublicfiles.com/fasttext/
    supervised-models/lid.176.ftz
```

Testing the official models:

```
# Testing the larger model
!./fasttext test lid.176.bin test.txt

# Testing the compressed model
!./fasttext test lid.176.ftz test.txt
```

Results:

- **lid.176.bin**:

```
N       2,500,834
P@1     0.946
```

Conclusion

Through this process, we have:

- Demonstrated that adding more data improves model performance.
- Shown that hyperparameter tuning further enhances accuracy.
- Highlighted the efficiency of using Tatoeba data over Wikipedia dumps and OSCAR data.
- Successfully included Kazakh language support in the model.
- Achieved higher accuracy than the official fastText models.

Efficiency Comparison:

- **Wikipedia Dumps**:
  - Download time: 72 hours
  - Storage required: ¿300 GB
- **OSCAR Data**:
  - Download and processing time: 11 hours
  - Storage required: 2 GB
- **Tatoeba Data**:
  - Download and processing time: 5 minutes
  - Storage required: 100 MB

Future Work: