SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing

Taku Kudo, John Richardson

presented by Daria Pirozhkova

NSU

April 13, 2021

Overview

1. Introduction
2. Tokenization's types
3. Tokenization algorithms
4. SentencePiece

Introduction

What is subword tokenization?

Tokenization is the task of splitting a sequence of text into units with semantic meaning. These units are called tokens, and the difficulty in tokenization lies on how to get the ideal split so that all the tokens in the text have the correct meaning, and there are no left out tokens.

What problem does it solve?

Example: "I love chocolateandcheese"
Word tokenization: "I", "love", "chocolateandcheese"
Subword tokenization: "I", "lo","ve", "choco","la","te","and", "chee","se"
Language specification: Japanese and Chinese

Tokenization's types

- Word level tokenization: "faster"
- Character level tokenization: f-a-s-t-e-r
- Subword level tokenization: "fast", "er"

Tokenization algorithms

BPE

BPE creates a base vocabulary consisting of all symbols that occur in the set of unique words and learns merge rules to form a new symbol from two symbols of the base vocabulary. It does so until the vocabulary has attained the desired vocabulary size.

AABABCABBAABAC	ADDCDBADAC	EDCDBEAC
AA – 2	AD – 2 AD = E	
AB – 4 AB = D	DD – 1	
BA – 3	DC – 1	
BC – 1	CD – 1	
CA – 1	DB – 1	
BB – 1	DA – 1	
AC – 1	AC – 1	

Table: Table caption

Tokenization algorithms

Unigram

Unigram algorithm defines a loss over the training data given the current vocabulary.
Then, for each symbol in the vocabulary, the algorithm computes how much the overall loss would increase if the symbol was to be removed from the vocabulary.
Unigram then removes p percent of the symbols whose loss increase is the lowest,
i.e. those symbols that least affect the overall loss over the training data.
This process is repeated until the vocabulary has reached the desired size

$$\mathcal{L} = -\sum_{i=1}^{N} \log \left(\sum_{x \in S(x_i)} p(x) \right)$$

Table: Table caption

SentencePiece

.

- SentencePiece is a language-independent subword tokenizer and detokenizer designed for Neural-based text processing. (including Neural Machine Translation)
- SentencePiece implements two subword segmentation algorithms, byte-pairencoding (BPE) and unigram language model.
- It enables building a purely end-to-end system that does not depend on any language specific processing.

SentencePiece

SentencePiece comprises four main components:

- Normalizer: a module to normalize semantically equivalent (unicode characters into canonical forms).
- Trainer
- Encoder
- Decoder

Lossless tokenization:
$$\text{Decode}(\text{Encode}(\text{Normalize}(\text{text}))) = \text{Normalize}(\text{text}).$$

SentencePiece

- vocabulary size:It reserves vocabulary ids for special meta symbols, e.g., unknown symbol (unk), BOS (s), EOS (/s) and padding (pad).
- normalization rule name: customizable character normalization

SentencePiece

```
import sentencepiece as spm

params = ('--input=input.txt␣'
          '--model_prefix=spm␣'
          '--vocab_size=1000')
spm.SentencePieceTrainer.Train(params)

sp = spm.SentencePieceProcessor()
sp.Load('spm.model')

print(sp.EncodeAsPieces('Hello␣world.'))
print(sp.EncodeAsIds('Hello␣world.'))
print(sp.DecodeIds([151, 88, 21, 887, 6]))
```

Figure 4: Python API usage (The same as Figure 1.)

```
import tensorflow as tf
import tf_sentencepiece as tfs

model = tf.gfile.GFile('spm.model', 'rb').read()

input_text = tf.placeholder(tf.string, [None])
ids, lens = tfs.encode(input_text, model_proto=model,
        out_type=tf.int32)
output_text = tfs.decode(ids, lens, model_proto=model)

with tf.Session() as sess:
   text = ['Hello␣world.', 'New␣York']
   ids_, lens_, output_text_ = sess.run([ids, lens, output_text
         ], feed_dict={input_text:text})
```

Figure 5: TensorFlow API usage

SentencePiece

Lang pair	setting (source/target)	# vocab.	BLEU
ja→en	Word model (baseline)	80k/80k	28.24
	SentencePiece	8k (shared)	29.55
	SentencePiece w/ pre-tok.	8k (shared)	29.85
	Word/SentencePiece	80k/8k	27.24
	SentencePiece/Word	8k/80k	29.14
en→ja	Word model (baseline)	80k/80k	20.06
	SentencePiece	8k (shared)	21.62
	SentencePiece w/ pre-tok.	8k (shared)	20.86
	Word/SentencePiece	80k/8k	21.41
	SentencePiece/Word	8k/80k	19.94

Table 1: Translation Results (BLEU(%))

Task	Tool	Pre-tok.	time (sec.) Japanese	time (sec.) English
Train	subword-nmt	yes	56.9	54.1
	SentencePiece	yes	10.1	16.8
	subword-nmt	no	528.0	94.7
	SentencePiece	no	217.3	21.8
Seg.	subword-nmt	yes	23.7	28.6
	SentencePiece	yes	8.2	20.3
	subword-nmt	no	216.2	36.1
	SentencePiece	no	5.9	20.3
Pre-tokenizaion KyTea(ja)/Moses(en)			24.6	15.8

Table 2: Segmentation performance. KFTT corpus (440k sentences) is used for evaluation. Experiments are executed on Linux with Xeon 3.5Ghz processors. The size of vocabulary is 16k. Moses and KyTea tokenizers are used for English and Japanese respectively. Note that we have to take the time of pre-tokenization into account to make a fair comparison with and without pre-tokenization. Because subword-nmt is based on BPE, we used the BPE model in SentencePiece. We found that BPE and unigram language models show almost comparable performance.

References

Taku Kudo and John Richardson (2018)

SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing

CoRR

Thank you for your attention