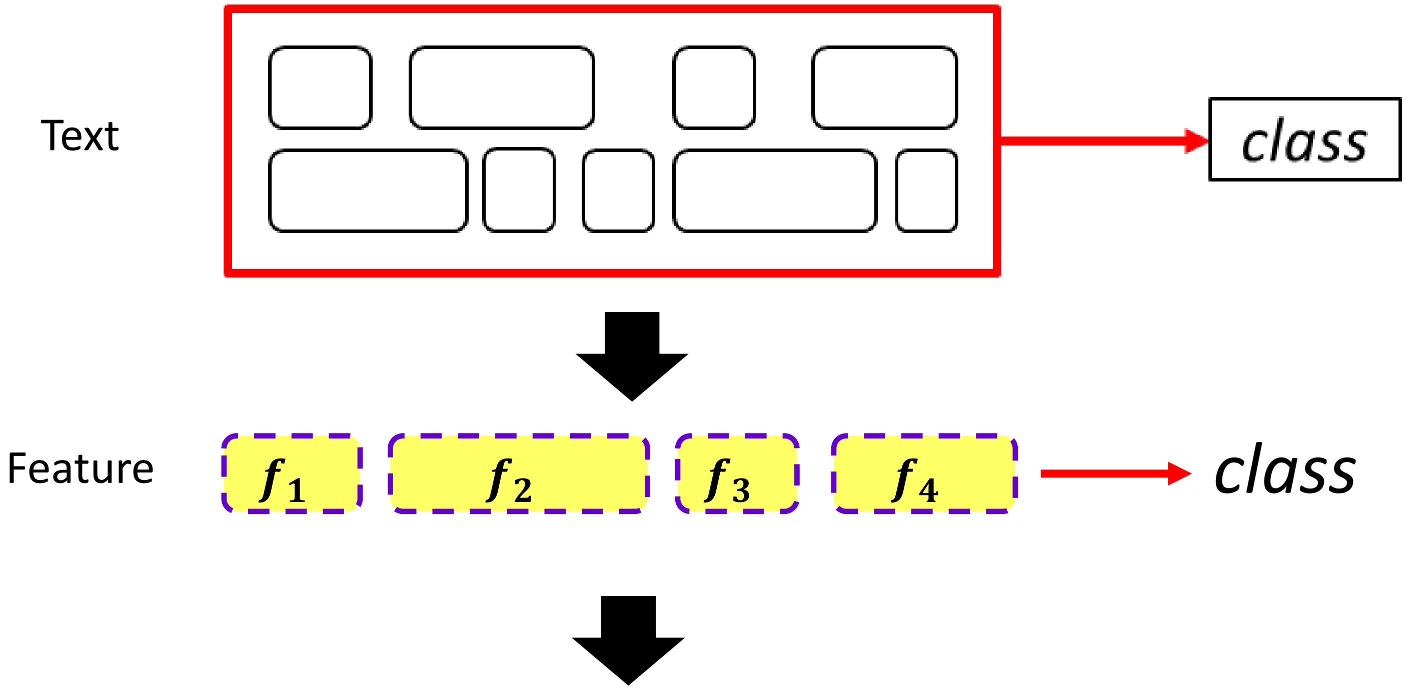
Natural Language Token Embeddings

정상근

(Traditional) Statistical-NLP

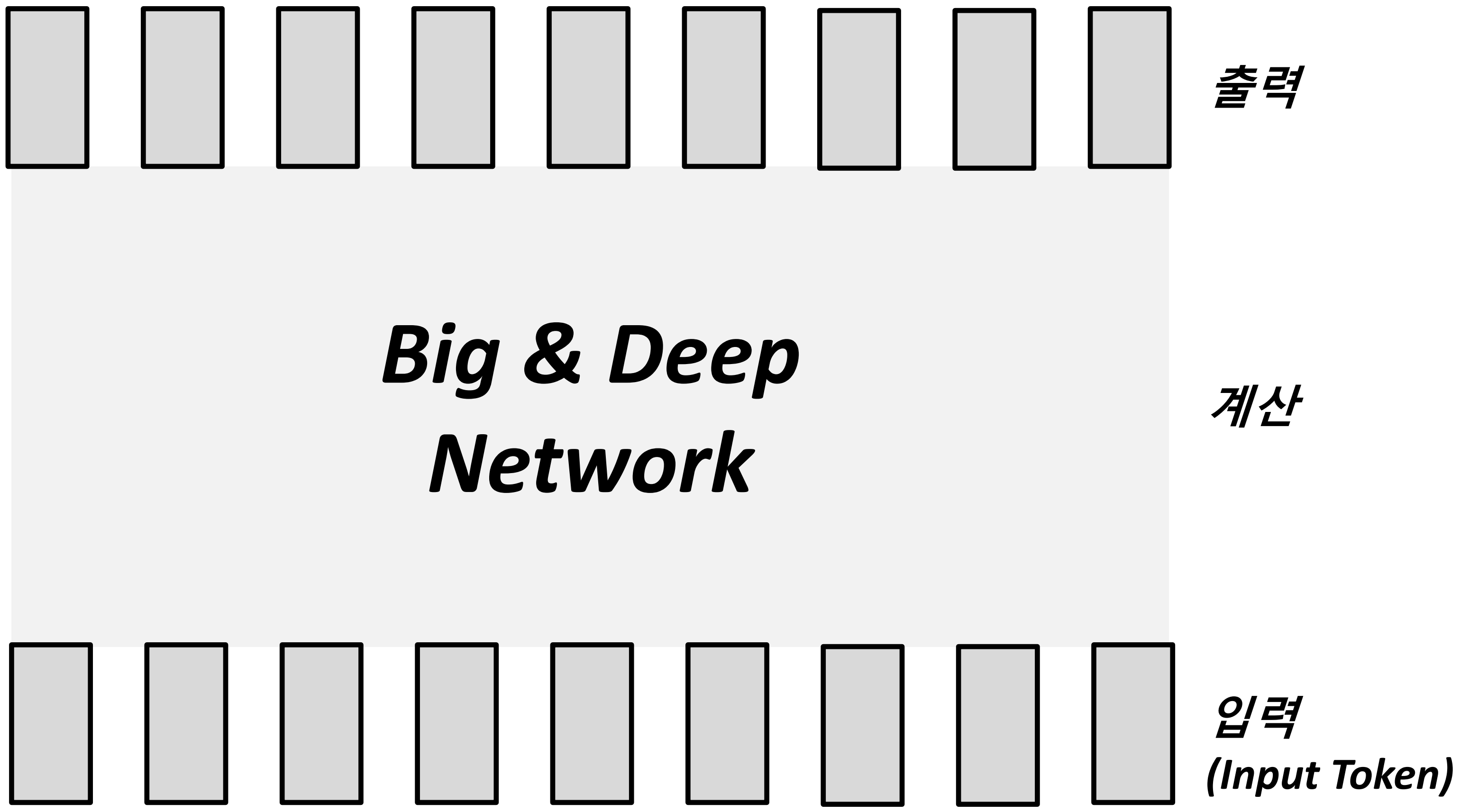


$$P(class \mid text) \approx P(class \mid f_1, f_2, f_3, f_4)$$

In above settings, token units do not matter since we use features to describe the sentence. The features can cross over token units.

Copyright, Sangkeun Jung, All Rights Reserved.

Big Picture – Deep Learning NLP



Sentence should be fed to DNN with token units

Copyright, Sangkeun Jung, All Rights Reserved.

[Question]

What are the appropriate token units?

Copyright, Sangkeun Jung, All Rights Reserved.

Candidate token units

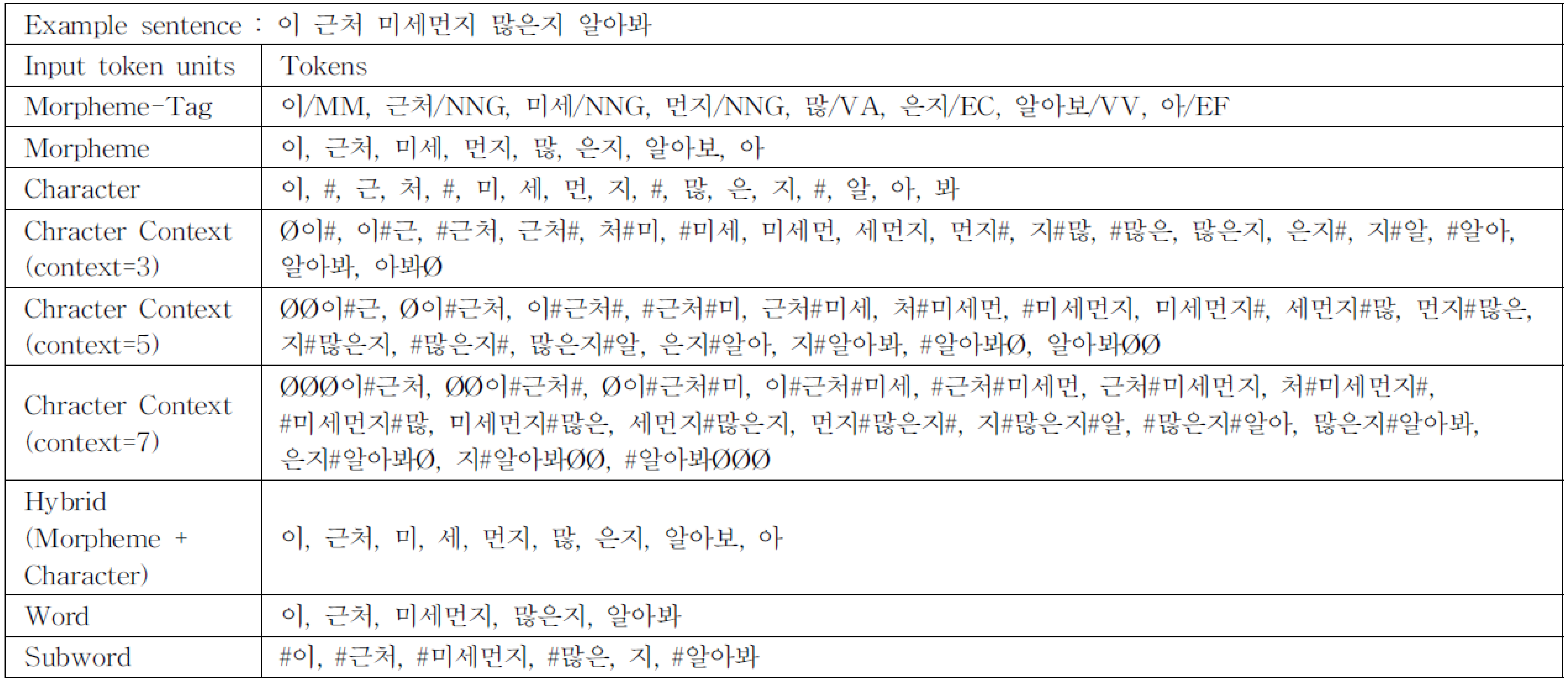
- **Word** : space segmentation units
- **Character** : letter units
- In Korean or Japanese,
  - **Morpheme**
  - **Hybrid** (Morpheme + Character)
    : Use frequently used top-K morpheme first.
    If not matched? Use character units.
- **Word-piece** : statistically induced tokens

Copyright, Sangkeun Jung, All Rights Reserved.

Example | Korean Token Units

표 1 단위 결정 방법론 별 입력 단위 예 (# 은 공백문자, Ø는 패딩(Padding) 문자를 의미함

Table 1 Examples of tokens corresponding to input token unit (# for space and Ø for padding)

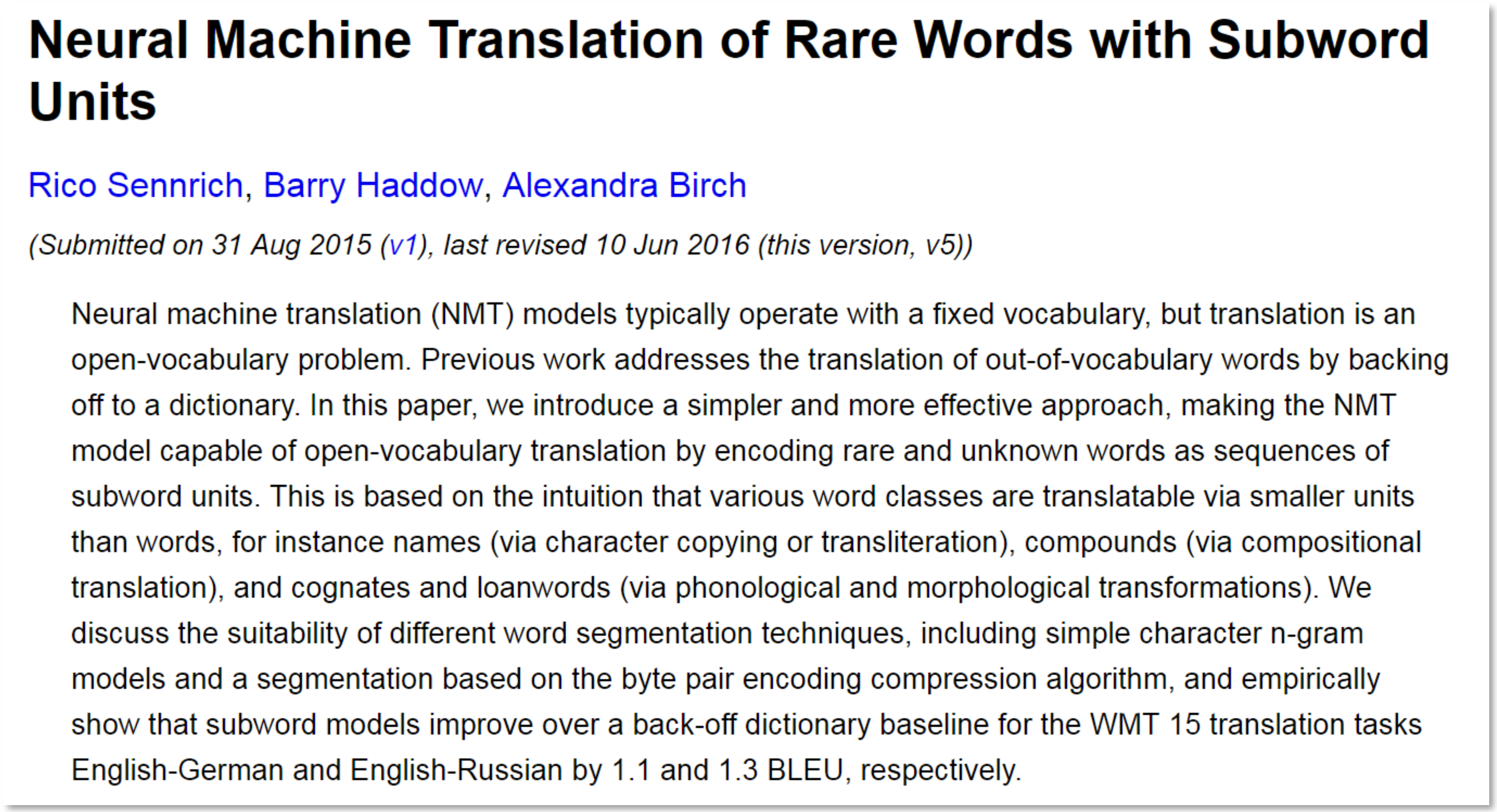


Example sentence : 이 근처 미세먼지 많은지 알아봐	
Input token units	Tokens
Morpheme-Tag	이/MM, 근처/NNG, 미세/NNG, 먼지/NNG, 많/VA, 은지/EC, 알아보/VV, 아/EF
Morpheme	이, 근처, 미세, 먼지, 많, 은지, 알아보, 아
Character	이, #, 근, 처, #, 미, 세, 먼, 지, #, 많, 은, 지, #, 알, 아, 봐
Chracter Context (context=3)	Ø이#, 이#근, #근처, 근처#, 처#미, #미세, 미세먼, 세먼지, 먼지#, 지#많, #많은, 많은지, 은지#, 지#알, #알아, 알아봐, 아봐Ø
Chracter Context (context=5)	ØØ이#근, Ø이#근처, 이#근처#, #근처#미, 근처#미세, 처#미세먼, #미세먼지, 미세먼지#, 세먼지#많, 먼지#많은, 지#많은지, #많은지#, 많은지#알, 은지#알아, 지#알아봐, #알아봐Ø, 알아봐ØØ
Chracter Context (context=7)	ØØØ이#근처, ØØ이#근처#, Ø이#근처#미, 이#근처#미세, #근처#미세먼, 근처#미세먼지, 처#미세먼지#, #미세먼지#많, 미세먼지#많은, 세먼지#많은지, 먼지#많은지#, 지#많은지#알, #많은지#알아, 많은지#알아봐, 은지#알아봐Ø, 지#알아봐ØØ, #알아봐ØØØ
Hybrid (Morpheme + Character)	이, 근처, 미, 세, 먼지, 많, 은지, 알아보, 아
Word	이, 근처, 미세먼지, 많은지, 알아봐
Subword	#이, #근처, #미세먼지, #많은, 지, #알아봐

Jung, “Effective Korean Token Units for Sequence Encoding in Deep Learning”, 2018, Journal of KISSE

Copyright, Sangkeun Jung, All Rights Reserved.

Word-Piece Model (Sentence-piece model)



Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich, Barry Haddow, Alexandra Birch

(Submitted on 31 Aug 2015 (v1), last revised 10 Jun 2016 (this version, v5))

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). We discuss the suitability of different word segmentation techniques, including simple character n-gram models and a segmentation based on the byte pair encoding compression algorithm, and empirically show that subword models improve over a back-off dictionary baseline for the WMT 15 translation tasks English-German and English-Russian by 1.1 and 1.3 BLEU, respectively.

Senrich et al., 2015

Copyright, Sangkeun Jung, All Rights Reserved.

Ideas of Sentencepiece (Senrich et al., 2015)

- Find suitable token units from the corpus
- Criteria
  - Frequently appeared longest units ➔ Token (K)
  - Rare word ➔ sperate characters ➔ Token (L)
  - Total # of vocab. = K+L
- How to find those units? Byte-Pair Encodings
  - There are many open s/ws
    - https://github.com/google/sentencepiece

Copyright, Sangkeun Jung, All Rights Reserved.

BPE Algorithm for find suitable vocab from documents

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

After 10 merges,

The vocab becomes

```
{
  'low</w>': 5,
  'low e r </w>': 2,
  'newest</w>': 6,
  'wi d est</w>’: 3
}
```

[Practice code available]

Copyright, Sangkeun Jung, All Rights Reserved.

BERT - Multilingual

Bert trained multilingual corpus into a single model

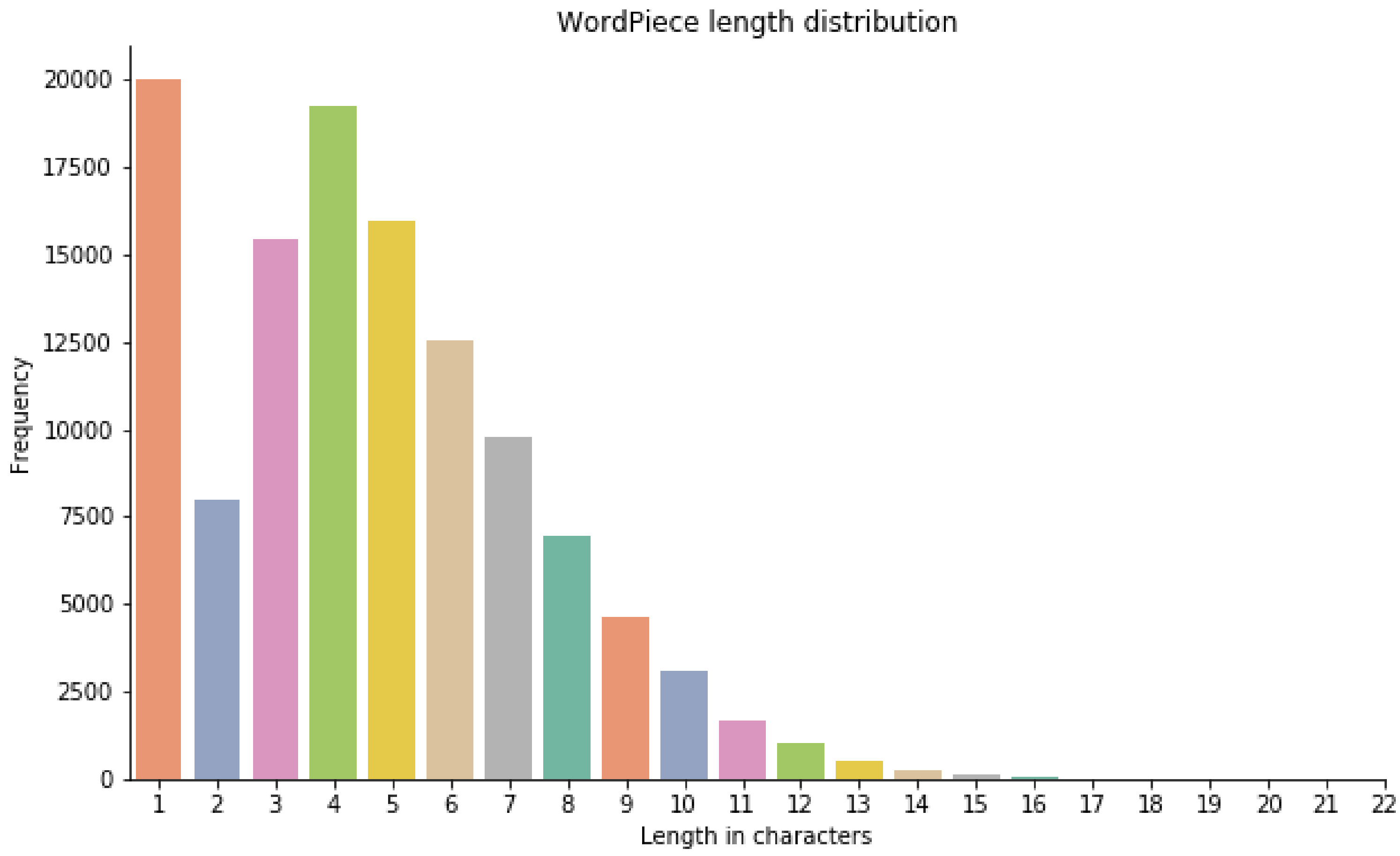
- 104 languages
- 119,547 WordPiece model,
- Non-word-initial units are prefixed with ##
- The first 106 symbols are reserved for constants like PAD and UNK and unused placeholders for application specific symbols.
- 36.5% of the vocabulary are non-initial word pieces.

- The vocabulary was created using the top 100 Wikipedia dumps.
- The authors oversampled small Wikipedias so we should see many non-English looking word pieces in the vocabulary.

http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

Copyright, Sangkeun Jung, All Rights Reserved.

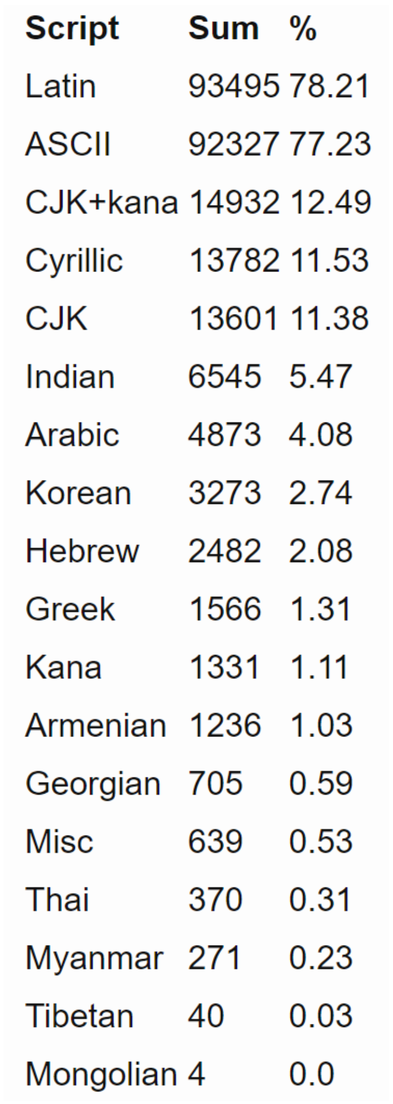
BERT – WordPiece Length Distribution



http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

Copyright, Sangkeun Jung, All Rights Reserved.

BERT – Multilingual (Language Source Analysis)



Script	Sum	%
Latin	93495	78.21
ASCII	92327	77.23
CJK+kana	14932	12.49
Cyrillic	13782	11.53
CJK	13601	11.38
Indian	6545	5.47
Arabic	4873	4.08
Korean	3273	2.74
Hebrew	2482	2.08
Greek	1566	1.31
Kana	1331	1.11
Armenian	1236	1.03
Georgian	705	0.59
Misc	639	0.53
Thai	370	0.31
Myanmar	271	0.23
Tibetan	40	0.03
Mongolian	4	0.0

I explored BERT's multilingual vocabulary by itself and through its tokenization on 54 languages that have UD treebanks. I found that the majority of elements in BERT's vocabulary are that of the **European languages**, most of them pure ASCII. Examining the output of BERT tokenizer confirmed that the tokenizer keeps English mostly intact while it **may generate different token distributions in morphologically rich languages.** The degree of how much this tokenization resembles a morphological segmentation remains to be explored.

http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

Copyright, Sangkeun Jung, All Rights Reserved.

도움자료

- 이론 : t.ly/WrVAi
- 실습
  - t.ly/rC-z6
  - t.ly/spiUc
  - t.ly/SrWDd

Copyright, Sangkeun Jung, All Rights Reserved.