Assignment 8
CSE 517: Natural Language Processing - University of Washington
Winter 2022

Solution 1.

Byte Pair encoding converts rare into a sequence of sub word units. Assume the model detects a word that is not in the model's vocabulary, let's say "walking". Rather than instantly defaulting to an unknown default token, we may see those words such as walked, walks exist in the corpus. If these words were divided into walk ed, walk s, and so forth, these words would all share a "walk", and the model would be able to include new information.

1.1 Byte pair encoding (BPE) ensures that the most frequently used words in the vocabulary are represented by a single token, while rare words are divided into two or more sub word tokens that may include useful information.
In our BPE algorithm, we are using "<s>" as a special token at the end of each word which is used to indicate a word boundary, allowing the program to determine the end of each word. This assists the algorithm in scanning each character and determining the character combination with the highest frequency. This enables the algorithm to distinguish between words such as "advance" and "workload." Both of these words include the word "ad," however one contains a "ad" token at the end and one at the beginning. As a result, tokens such as "ad" and "ad <s>" would be treated differently. If the algorithm encounters the token "ad <s>," it will recognize it as the token for the word "workload," not "Advance."

1.2 As with the BPE algorithm, we look for the most frequent pairings, combine them, and repeat the process until we reach our token or iteration limit. As a result, if the number of iterations is modified (let's say reduced), the number of tokens may change.

1.3 The scatter plot below illustrates points (x; y), one for each iteration of the algorithm, with x representing the current size of the type vocabulary and y representing the length of the training corpus (in tokens) for that vocabulary's types.

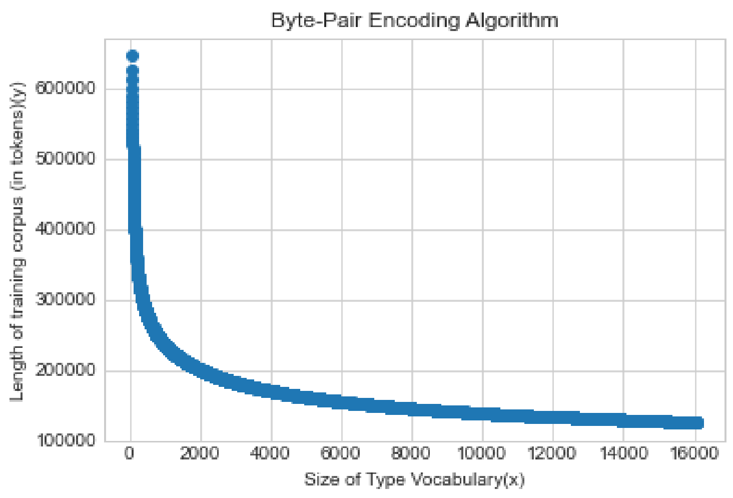
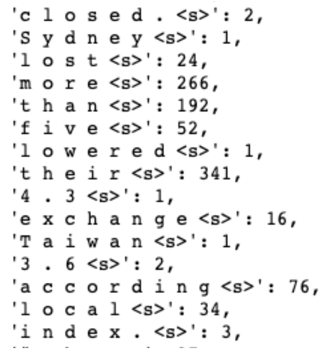


Fig 1. Scatter plot between length of training corpus vs size of type vocabulary

When the BPE algorithm is implemented, we got in total 15968 number of iterations with the length of types of vocab of 16051 and frequency of types of vocab is 17436.
Length of training vocab is 20660 (snippet attached below) and the frequency of training vocab came out to be 107262.
Length of training data under final type vocabulary is 124701



```
'c l o s e d . <s>': 2,
'S y d n e y <s>': 1,
'l o s t <s>': 24,
'm o r e <s>': 266,
't h a n <s>': 192,
'f i v e <s>': 52,
'l o w e r e d <s>': 1,
't h e i r <s>': 341,
'4 . 3 <s>': 1,
'e x c h a n g e <s>': 16,
'T a i w a n <s>': 1,
'3 . 6 <s>': 2,
'a c c o r d i n g <s>': 76,
'l o c a l <s>': 34,
'i n d e x . <s>': 3,
```

1.4 Encode text: “stun unrest “

First splitting the text : s t u n <s> u n r e s t <s>

Iteration 1

Bigram to merge: st

Encoded word: st u n <s> u n r e st <s>

Iteration 2

Bigram to merge: un

Encoded word: st un <s> un r e st <s>

In this example, we get encoding: st un <s> un r e st <s>

1.5 *Advantages of character level encoding:*

- Let’s say we have following sentences:

  “Equal 7/9” 234 kgs slips 50K LBS”

  When working at the word level, we must first pick a vocabulary; that is, the list of words that our model can handle is fixed, and all others are mapped to UNK. The tokens 7/9", and 50K are absent from word vectors trained on GoogleNews. Also, the closest terms to slips are slip, fall, and so forth, showing that the model word slip is a verb and slips is an inflection of the verb. Therefore, our model will perceive the sentence as mentioned below:

  “Equal UNK UNK UNK Fall UNK LBS”

  We would be able to understand the information using character level encoding, which has an effectively infinite vocabulary.

- NLP is frequently used to evaluate user-generated text, which is filled with spelling mistakes, emoticons, acronyms, jargon, and syntax. In addition, as changes over time, new words and symbols are introduced on a regular basis. Although character level models may not solve all of the problems associated with human text, they do allow our models to evaluate it more extensively.

Disadvantages of character level encoding:

- Working at the character level effectively multiplies your sequence's length by the average number of characters in a word. The large sequence length, when compared to word-level models, makes training more difficult due to the gradient descent vanishing/exploding problem. Hence, longer sequence increases computational cost.

- Individual characters are typically irrelevant when performing tasks such as named entity recognition that require sequence tagging. The NER model determines if a given word is an entity or not and then evaluates it based on that output. If a character-level encoding model is used in that situation, the output will be a character, which adds more work.

Solution 2.

1. If Abigail makes noise, no one can sleep.

$$MAKES-NOISE(ABIGAIL) \rightarrow (\forall x\ \neg CAN-SLEEP(x)$$

2. If Abigail makes noise, someone cannot slep.

$$MAKES-NOISE(ABHIGAIL) \rightarrow \exists y \neg\ CAN-SLEP(y)$$

3. None of Abigail's brothers can sleep.

$$\forall x \neg\ CAN-SLEEP(BROTHER-OF(ABIGAIL)$$

OR

$$\forall x \neg\ CAN-SLEEP(x) \land BROTHER(x, ABIGAIL)$$

4. If one of Abigail's brothers makes noise, Abigail cannot sleep.

$$\exists x\ MAKES-NOISE(x) \land BROTHER(x, ABIGAIL) \rightarrow\ \neg CAN-SLEEP(ABIGAIL)$$

Solution 3.
3.1

Figure 1: Version #1 *He saw her duck (VP).*

	Stack	**Buffer**	**Action**
0		*He*| *saw*| *her*| *duck*|.	NT(S)
1	(S	*He*| *saw*| *her*| *duck*|.	NT(NP)
2	(S|(NP	*He*| *saw*| *her*| *duck*|.	SHIFT
3	(S|(NP| *He*	*saw*| *her*| *duck*|.	REDUCED
4	(S|(NP| *He*)	*saw*| *her*| *duck*|.	NT(VP)
6	(S|(NP *He*)|(VP	*saw*| *her*| *duck*|.	SHIFT
7	(S|(NP *He*)|(VP *saw*	*saw*| *her*| *duck*|.	REDUCED
8	(S|(NP *He*)|(VP *saw*)	*her*| *duck*|.	NP
9	(S|(NP *He*)|(VP *saw*)|(NP	*her*| *duck*|.	SHIFT
10	(S|(NP *He*)|(VP *saw*)|(NP *her*	*duck*|.	REDUCED
11	(S|(NP *He*)|(VP *saw*)|(NP *her*)	*duck*|.	VP
12	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(VP	*duck*|.	SHIFT
13	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(VP *duck*	.	REDUCED

14	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(VP *duck*)	.	SHIFT
15	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(VP *duck*)|.		REDUCED
16	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(VP *duck*).)		

Figure 2: Version #2 He *saw her duck (NP).*

	Stack	**Buffer**	**Action**
0		*He*| *saw*| *her*| *duck*|.	NT(S)
1	(S	*He*| *saw*| *her*| *duck*|.	NT(NP)
2	(S|(NP	*He*| *saw*| *her*| *duck*|.	SHIFT
3	(S|(NP| *He*	*saw*| *her*| *duck*|.	REDUCED
4	(S|(NP| *He*)	*saw*| *her*| *duck*|.	NT(VP)
6	(S|(NP *He*)|(VP	*saw*| *her*| *duck*|.	SHIFT
7	(S|(NP *He*)|(VP *saw*	*saw*| *her*| *duck*|.	REDUCED
8	(S|(NP *He*)|(VP *saw*)	*her*| *duck*|.	NP
9	(S|(NP *He*)|(VP *saw*)|(NP	*her*| *duck*|.	SHIFT
10	(S|(NP *He*)|(VP *saw*)|(NP *her*	*duck*|.	REDUCED
11	(S|(NP *He*)|(VP *saw*)|(NP *her*)	*duck*|.	NP
12	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(NP	*duck*|.	SHIFT
13	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(NP *duck*	.	REDUCED
14	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(NP *duck*)	.	SHIFT
15	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(NP *duck*)|.		REDUCED
16	(S|(NP *He*)|(VP *saw*)|(NP *her*)|(NP *duck*).)		

3.2 All three features, stack, buffer, and dependency arc are significant in this classifier. The stack represents the elements that correspond to nodes (words + Root), the buffer represents the node list, and the dependency arcs represent the action set.

3.3 Somewhat surprisingly, the approach above was described without mention of the grammar! Suppose you have a PCFG, and you want to apply top-down transition-based parsing to find a tree with high 4 probabilities for the input sentence. How might you adapt the search version of this parser to try to find a high-probability tree?

3.4 In greedy top-down transition-based parsing, a bad decision early on can really mess things up. One danger comes about when the grammar has left-recursive rules like A ! A B or "cycles" of them, like A ! B C and B ! A D. Describe a way to constrain such a parser so that it won't go into an infinite loop.

References:

[1] https://leimao.github.io/blog/Byte-Pair-Encoding/
[2] https://github.com/rsennrich/subword-nmt
[3] https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0
[4] https://www.lighttag.io/blog/character-level-NLP/
[5] Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2021. All rights reserved. Draft of December 29, 2021.

[6] Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.