Natural Language Processing

Sub-word Tokenization

Outline

Recap

Word Segmentation

Sub-word Tokenization

The Prompt Report: A Systematic Survey of Prompting Techniques, https://arxiv.org/pdf/2406.06608

News

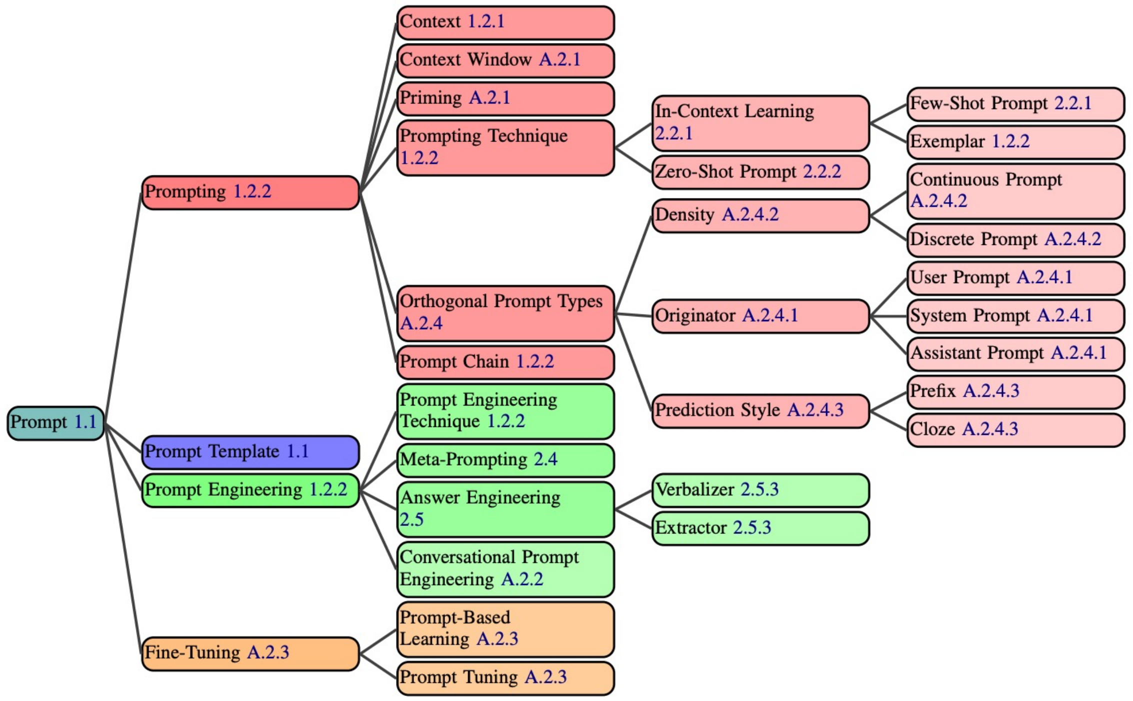
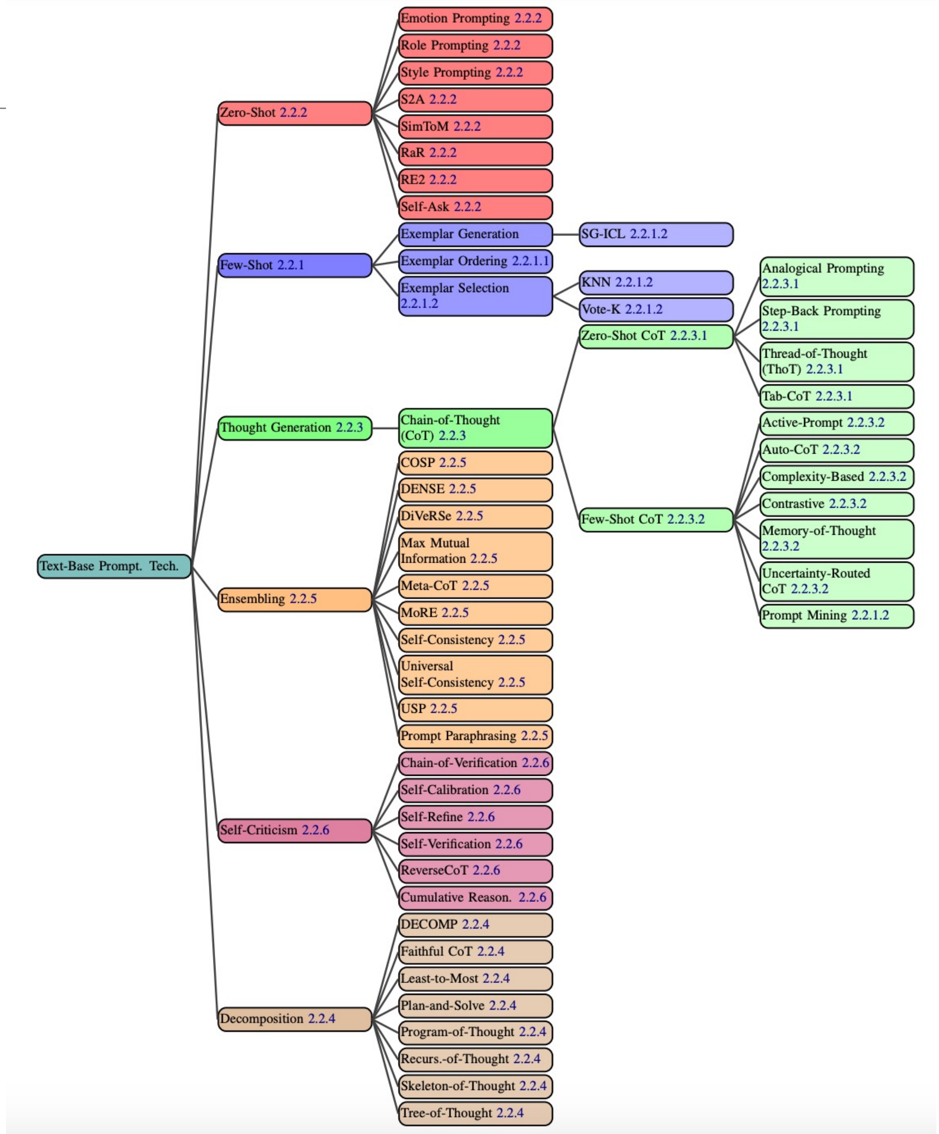
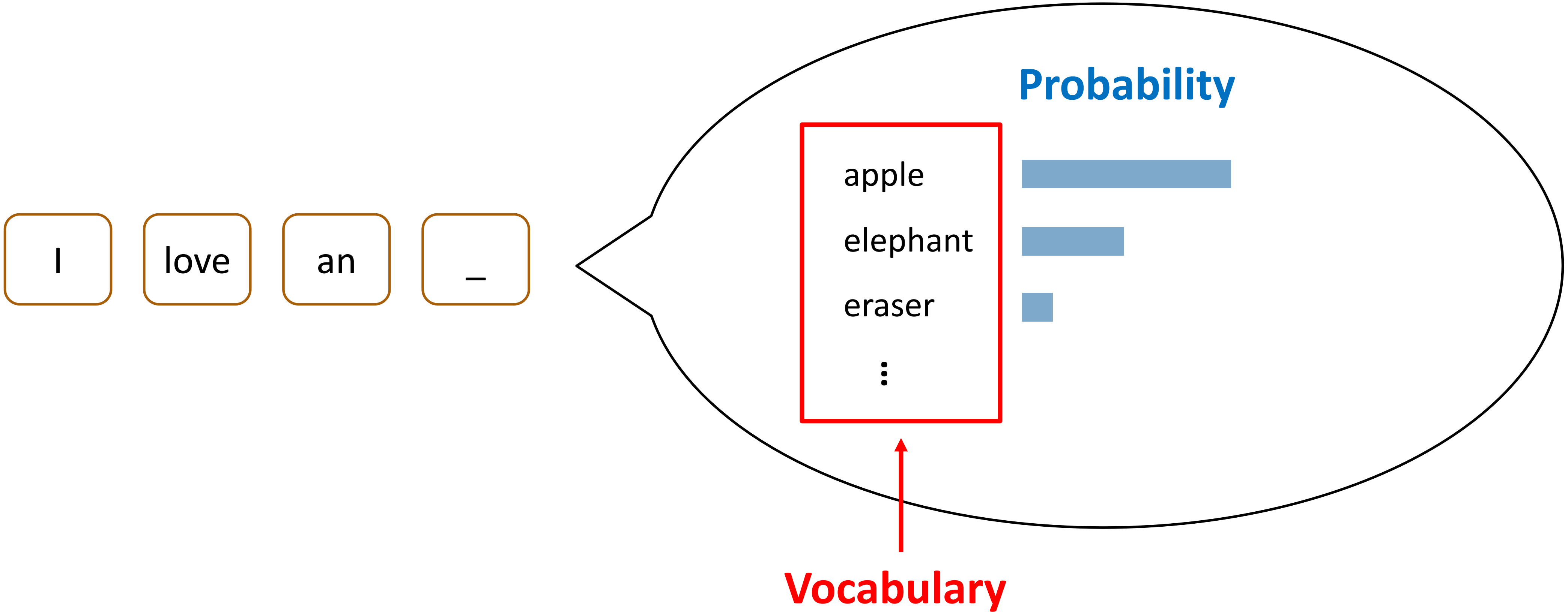


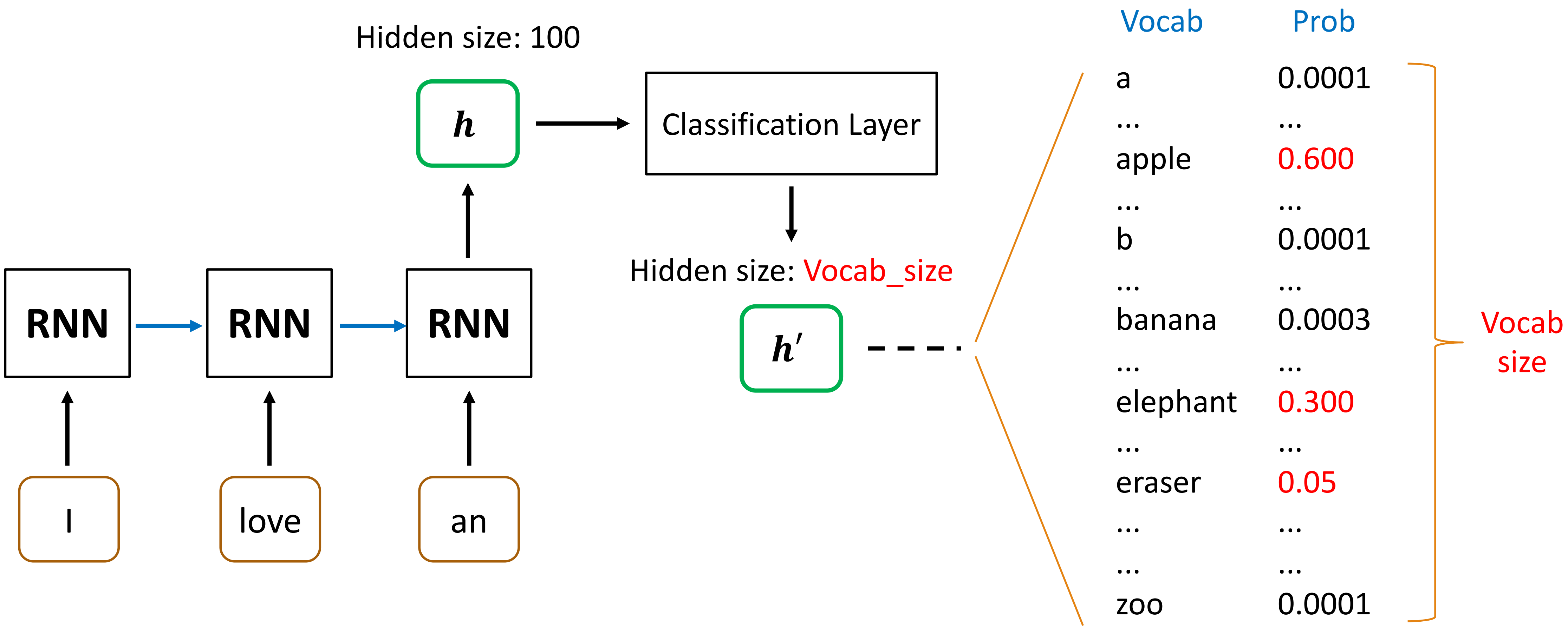
Figure 1.3: A Terminology of prompting. Terms with links to the appendix are not sufficiently critical to describe in the main paper, but are important to the field of prompting. Prompting techniques are shown in Figure 2.2



Recap: Word Representations



Recap: Word Representations (Details)



Basic Pipeline of Natural Language Processing

Build the vocabulary

↓

Learn the representations (Training)

↓

Perform predictions (Testing)

Size(vocabulary)
BERT: 30522
GPT-2 / GPT-3: 50257
T5: 32,128

How to build the vocabulary?

- Segment words based on delimiters (e.g., white spaces).
- Then we can collect the words from training corpora.

I love apples. I like apples and pineapples.

(You can also perform stemming after word segmentation!)

Vocabulary

Word	**ID**
and	0
apples	1
I	2
like	3
love	4
pineapples	5
.	6
<UNK>	7

Unknown words (not shown up in the training corpora)

Issues of Delimiter-based Segmentation

- Only work for Western languages.
  - Cannot work for Chinese, Japanese, ...
- Cannot handle unseen words (not shown up in the training corpora)
  - A misspelled word contains morphological information but become an unknown word.

Issues of Delimiter-based Segmentation (Continued.)

- For machine translation, there is not always a 1-to-1 correspondence between source and target words since compound words may exist in target language.

- For example, sewage water treatment plant (English) -> Abwasserbehandlungsanlage (German)

Abwasser | behandlungs | anlage
sewage water | treatment | plant/facility

Sub-word units are favored.

Common Sub-word Tokenization Algorithms

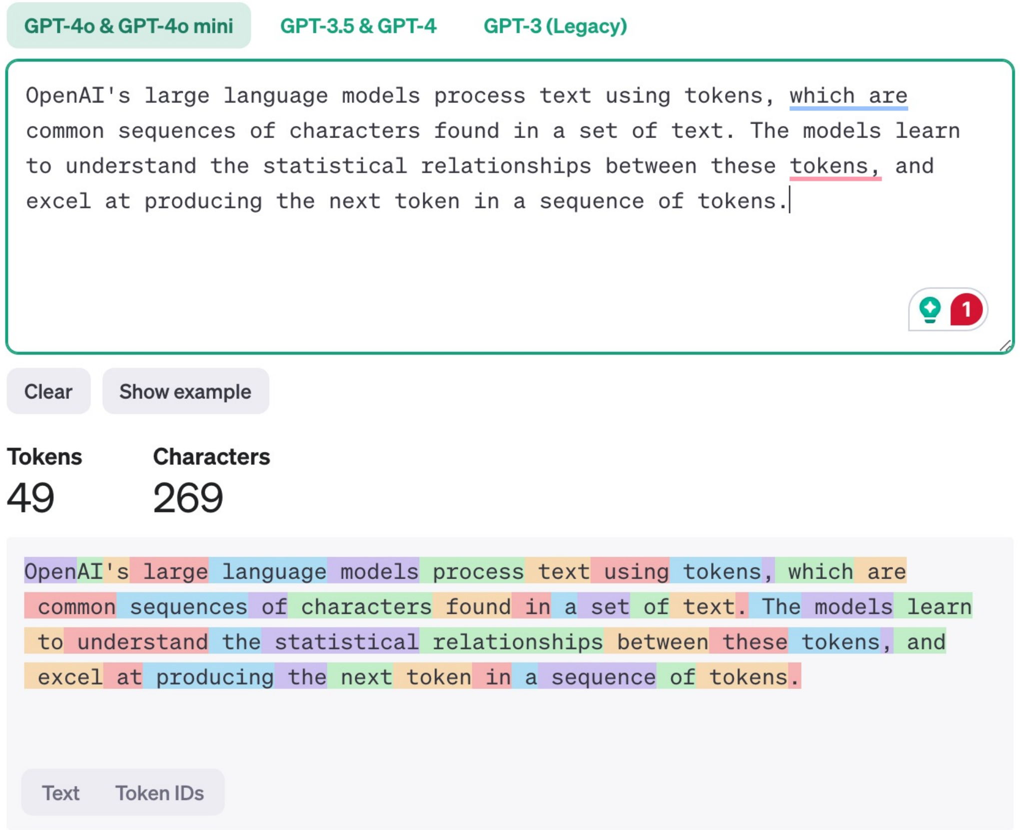
- Byte Pair Encoding (BPE) (Sennrich et al., 2016)[1]
  - GPT series
- WordPiece (Schuster and Nakajima, 2012)[2]
  - BERT, T5
- Unigram Language Model (Kudo, 2018)[3]

[1] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), 2016.
[2] Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2012.
[3] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL). 2018.

Segmentation vs. Tokenization

- All tokenization is segmentation, but not all segmentation is tokenization.
- Segmentation can be:
  - Word segmentation from a sentence
  - Sentence segmentation from a document
- Tokenization can be:
  - Word segmentation from a sentence (then `words` become `tokens`)
  - Sub-word tokenization from a word or sentence

OpenAI Tokenizer https://platform.openai.com/tokenizer



Sub-word Tokenization (1)

- Given a training corpus, we first turn each word into a sequence of characters (separated by spaces)

Training corpus

low low low low low lower lower newest newest newest newest newest newest widest widest widest

Word	Frequency
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

- </w> is a special end-of-word symbol, allowing us to restore the original tokenization.

Sub-word Tokenization (1)

- Initialize the vocabulary with the existing characters:

low low low low low lower lower newest newest newest newest newest newest widest widest widest

Initial vocab: </w>, d, e, i, l, n, o, r, s, t, w

Sub-word Tokenization (2)

- Find the character **pair** with the highest frequency

Word	Frequency
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

Found “e s” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w

Sub-word Tokenization (3)

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w $\boxed{\text{e s}}$ t </w>	6
w i d $\boxed{\text{e s}}$ t </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (4)

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **es** t </w>	6
w i d **es** t </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (2)

Repeated (2)-(4) according to `num_merges`

- Find the character **pair** with the highest frequency

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w es t </w>	6
w i d es t </w>	3

Found “es t” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (3)

Repeated (2)-(4) according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w $\boxed{\text{es t}}$ </w>	6
w i d $\boxed{\text{es t}}$ </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es est

Sub-word Tokenization (4)

Repeated (2)-(4) according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **est** </w>	6
w i d **est** </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est

Sub-word Tokenization (2)

Repeated (2)-(4)
according to `num_merges`

- Find the character **pair** with the highest frequency

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w $\boxed{\text{est </w>}}$	6
w i d $\boxed{\text{est </w>}}$	3

Found “est </w>” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est

Sub-word Tokenization (3)

Repeated (2)-(4) according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est </w>	6
w i d est </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (4)

Repeated (2)-(4) according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **est</w>**	6
w i d **est</w>**	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (2)

Repeated (2)-(4)
according to `num_merges`

- Find the character **pair** with the highest frequency

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Found “l o” with the highest frequency 5+2=7

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (3)

Repeated (2)-(4) according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

Sub-word Tokenization (4)

Repeated (2)-(4) according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	Frequency
lo w </w>	5
lo w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

Finish Sub-word Leaning

- Assume `num_merges`=4 (we just repeated four times.)
- `num_merges` is a hyperparameter that you need to set for BPE.
- The learned vocabulary is: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

Tokenization with Learned BPE

- **The learned vocabulary** is: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo
- We first turn each word into a sequence of characters with </w> placed at the end of a word, the same as the first step during the learning phase.
- Then we can merge the characters according to **the learned vocabulary**.
- Examples:

l o w </w> → lo | w | </w>

w i d e s t </w> → w | i | d | est</w>

Properties of BPE

- The final learned vocabulary size = **initial** size + `num_merges`

  Initial vocab: </w>, d, e, i, l, n, o, r, s, t, w

  Learned vocab: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

- This algorithm is based on statistics, so frequent sub-word units in provided corpora will be put to the learned vocabulary.

The Problem of BPE

- BPE splits a sentence with larger sub-words in the vocabulary in default.

greedy, deterministic, and left-to-right

Original: Hello world

BPE: Hell / o /world ← **May be sub-optimal.**

Other choices: H / ello / world

He / llo / world

He / l / l / o / world

H / el / l / o / world

Tokenization with Probabilities?

H	0.03
He	0.001
Hell	0.007
el	0.002
ello	0.024
llo	0.062
l	0.003
o	0.055
world	0.011

Sub-word Tokens	Product of occurrence probabilities
Hell / o /world	0.007 * 0.055 * 0.011 = 0.000004235
H / ello / world	0.03 * 0.024 * 0.011 = 0.00000792
He / llo / world	0.001 * 0.062 * 0.011 = 0.000000682
He / l / l / o / world	0.001 * 0.003 * 0.003 * 0.055 * 0.011 = 5.445E-12
H / el / l / o / world	0.03 * 0.002 * 0.003 * 0.055 * 0.011 = 1.089E-10

(BPE example)

Unigram Language Model Tokenization

- Similar to Byte Pair Encoding (BPE), Unigram Language Model Tokenization (ULM) provides an algorithm to tokenize a sentence into subwords.
- Unlike BPE, ULM performs tokenization based on joint probabilities for each sentence.
- In other words, every token in the vocabulary has each own probability trained from the corpus.

Steps of Unigram Tokenization

Step1 Define a vocab size and build the vocabulary.

↓

Step2 Train a unigram gram language model

↓

Step3 Prune subwords from the vocabulary.

↓

Get the final vocabulary with probabilities.

Step1: Define a vocab size and build the vocabulary (1/2)

- Define a **vocab size** that you desire.
- Get all the subwords (including characters) from the corpus.
- Keep the most frequent subwords that meet the **vocab size**.
- Enhanced Suffix Array (not included in this course) is used to speed up this process in the original paper (Kudo, 2018).

[Kudo, 2018] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." ACL 2018.

Step1: Define a vocab size and build the vocabulary (2/2)

- Once you get the most frequent subwords with counts, you can calculate frequencies for each subword:

$$frequency(x_i) = \frac{\text{Count } of\ x_i}{Sum(\text{all\_counts})}$$

, where x_i is a subword from the vocab.

[Kudo, 2018] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." ACL 2018.

Step2: Train a unigram gram language model

- This language model is not based on neural networks
- Instead, it is a probabilistic model.

Probabilities of all segments of the best segmentation in X

$$\mathcal{L} = \sum_{s=1}^{|D|} \log(P(X^{(s)})) = \underbrace{\sum_{s=1}^{|D|} \log\Big(\underbrace{\sum_{\mathbf{x} \in \mathcal{S}(X^{(s)})} P(\mathbf{x})}\Big)}$$

Summation of all probabilities among the corpus

Symbol	Meaning
$\mathcal{S}$	Set of segments
$X^{(s)}$	s-th sentence
D	corpus
P	probability
$\mathcal{L}$	likelihood for the EM (Expectation-Maximization) algorithm

- P(x) at the right adopts the best segmentation for a sentence. The Viterbi algorithm (not included in this course) is used to speed up this finding process.

Step3: Prune subwords from the vocabulary.

- Iteratively calculate the loss for each subword in the vocab.
- Keep the top η % of the vocab size for the minimization of loss.
  - E.g., η can be set as 80.
- Finally, a vocabulary with ULM is created.
- Check these two links for more implementation details!
  - https://github.com/google/sentencepiece
  - https://huggingface.co/learn/nlp-course/chapter6/7

Inference and Subword Sampling

- After a ULM is trained, a word can be split into subwords according its joint probability.
- However, not always the segmentation with the greatest probability is selected.
- Instead, subword sampling was adopted in the original paper of ULM.

Symbol	Meaning
X	a sentence
x_i	i-th segmentation from the l best segmentations
l	hyperparameter for the number of best segmentations
α	hyperparameter for controlling the smoothness of the distribution
$P(\mathrm{x}_i \| X)$	probability of a segmentation given a sentence X
n_i	number of tokens (t_k) in x_i

$$P(\mathrm{x}_i|X) \cong \underbrace{P(\mathrm{x}_i)^{\alpha} / \sum_{i=1}^{l} P(\mathrm{x}_i)^{\alpha}}_{\text{multinomial distribution}} \;,\; P(\mathrm{x}_i) = \prod_{k=1}^{n_i} P(t_k)$$

Inference and Subword Sampling

- Enable subword sampling, a word can be produced into different segmentations.
- The following outputs are results using T5-base tokenizer (5 times), given a word “internationalization”:

```
▁in / tern / at / i / o / n / ali / z / ation
▁ / inter / national / ization
▁ / international / ization
▁ / in / t / e / r / nati / on / al / ization
▁inter / nati / on / a / liz / a / tion
```

- Note that T5 tokenizer add “_” before a word for boundary.

Sub-word Tokenization Choices of PLMs

Sub-word Tokenization	Models
WordPiece (Schuster and Nakajima, 2012)	BERT, ALBERT, MT-DNN
BPE (Sennrich et al., 2016)	RoBERTa, XLM, GPT-1, GPT-2, GPT-3
Unigram (Kudo, 2018)	XLNet, T5, mT5

[Schuster and Nakajima, 2012] Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." ICASSP 2012.
[Sennrich et al., 2016] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units. ACL 2016.
[Kudo, 2018] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." ACL 2018.

Comparison between BPE and ULM

Aspect	ULM	BPE
Algorithm Type	Expectation-Maximization (EM) algorithm	Greedy algorithm without probabilistic modeling
Training Complexity	Slower	Faster
Segmentation Consistency	Can produce different segmentations for the same word	Produces consistent segmentations for the same word every time.
Speed at Inference Time	Slower	Faster
Downstream tasks	Better at machine translation (maybe)	Suitable for most tasks

Why do we need Sub-word Tokenization?

- With sub-word tokenization algorithms, we can handle representations for unknown words (or mis-spelled words / compound words).
- In machine translation, the compound word issues between source and target languages can be alleviated.
- State-of-the-art pre-trained language models (e.g., GPT-3, BERT) adopt sub-word tokenization algorithms before pre-training.

Limitations of Sub-word Tokenization

- (Not many disadvantages for sub-word tokenization)
- The hyperparameter `num_merges` needs to be tuned.
- Once the learned vocabulary is created, it becomes fixed. The algorithm needs to be re-run after adding new data.
- How about Chinese?
  - Character-level encoding