THE UNIVERSITY OF
MELBOURNE

Comp90042 Natural Language Processing

Workshop (Week2 – 2025s1)

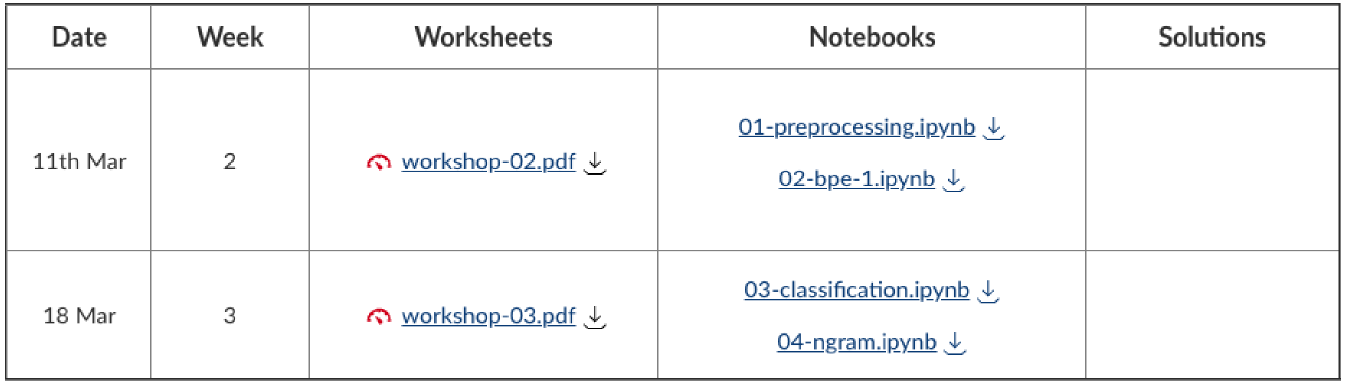
Dr Jean Lee

About Workshop

<u>COMP90042_2025_SM1 > Modules > Workshops > Worksheets</u>

- **Discussion**: regarding key concepts in the lectures **(~ 30mins)**
  - Worksheet (.pdf file)
- **Programming**: linking theory to practice **(~ 30mins)**
  - Jupyter notebook on Google Colab (.ipynb file)

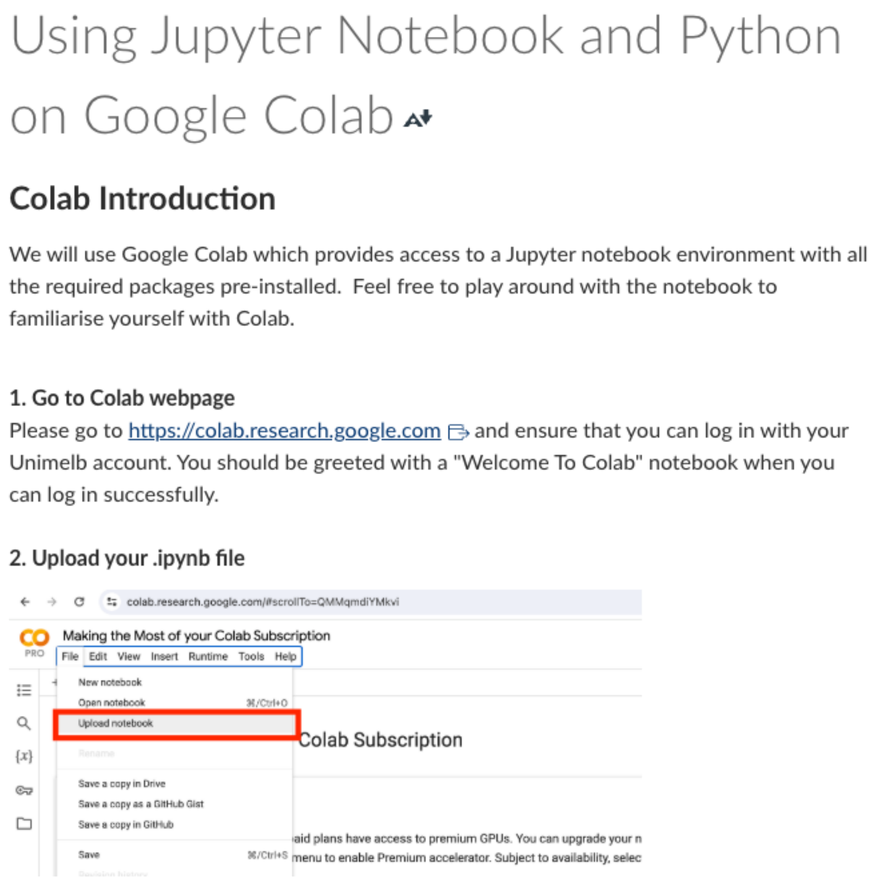
- *We are not able to cover all questions in one-hour workshop. Please make the most of **Ed Discussion**.*
- *The Solution to the tutorial will be released on **Next Week**.*



Date	Week	Worksheets	Notebooks	Solutions
11th Mar	2	workshop-02.pdf	01-preprocessing.ipynb 02-bpe-1.ipynb	
18 Mar	3	workshop-03.pdf	03-classification.ipynb 04-ngram.ipynb	

Jupyter Notebook on Google Colab

COMP90042_2025_SM1 > Modules > Resources > Using Jupyter Notebook...



Using Jupyter Notebook and Python on Google Colab

Colab Introduction

We will use Google Colab which provides access to a Jupyter notebook environment with all the required packages pre-installed. Feel free to play around with the notebook to familiarise yourself with Colab.

1. Go to Colab webpage

Please go to https://colab.research.google.com and ensure that you can log in with your Unimelb account. You should be greeted with a "Welcome To Colab" notebook when you can log in successfully.

2. Upload your .ipynb file

colab.research.google.com/#scrollTo=QMMqmdiYMkvi

Making the Most of your Colab Subscription

Colab Subscription

aid plans have access to premium GPUs. You can upgrade your n
menu to enable Premium accelerator. Subject to availability, selec

https://colab.research.google.com/

Agenda

1. Icebreaker with your peers
2. Discussion on worksheet questions
3. Notebook on programming

At the end of this workshop you should:

- *be familiar with the* ***Tokenisation*** *and* ***Normalisation*** *(stemming and lemmatisation)*
- *be familiar with* ***Byte-pair Encoding(BPE)*** *algorithm, and be able to implement it*

Icebreaker

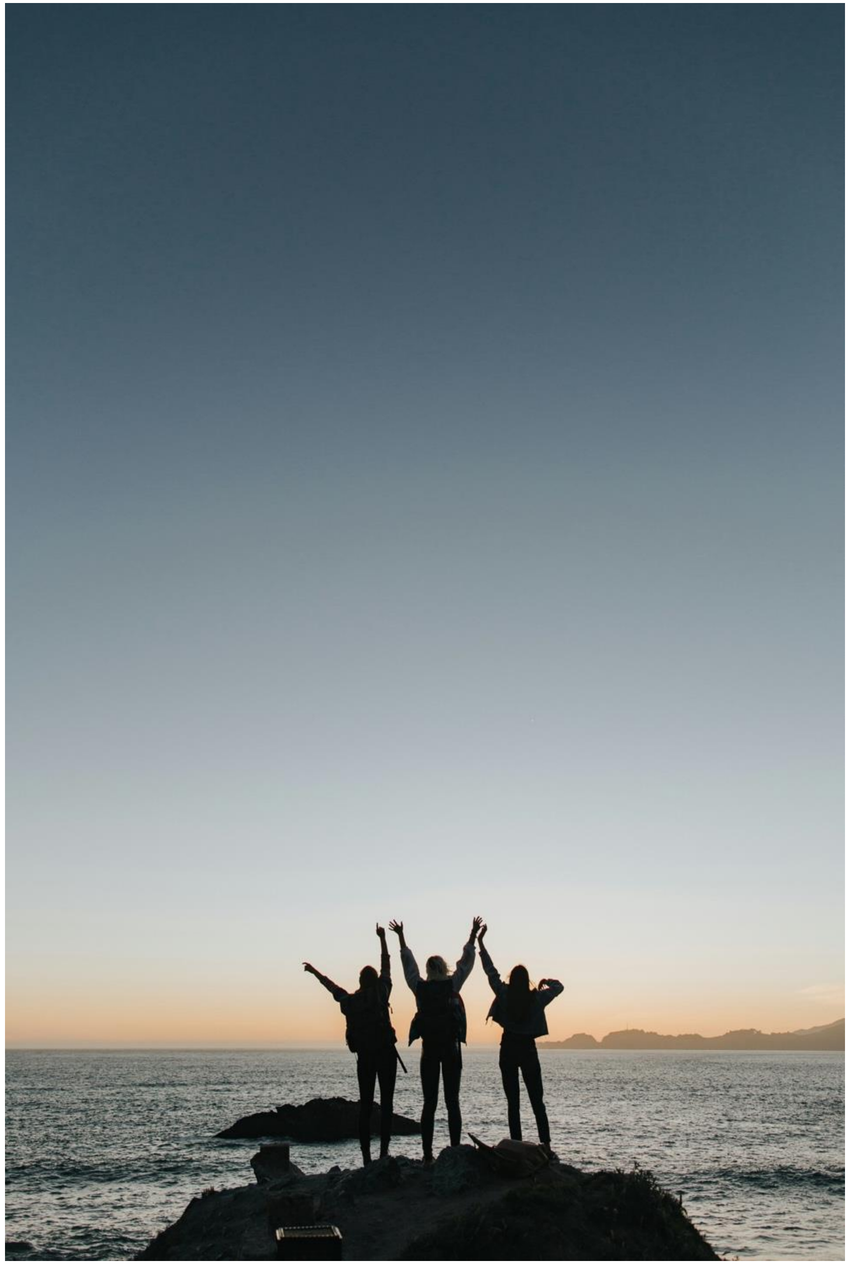
Get into groups of 2-4 and Introduce yourself

- What is your name?
- What are you studying?
- Why choose COMP90042?

Then, Discuss the first question of the worksheet

- Give some examples of text processing applications that you use on a daily basis.

You should start to know each other as future teammates for the ***project*** *and* ***peer review*** *task.*



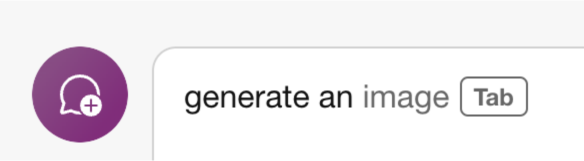
Discussion
- Examples of text processing applications

Possible answers

- Web search engines (Google, Copilot, Bing, Perplexity,…)
- Speech-to-text systems (Siri, Alexa, Google, …)
- Spelling correction (Grammarly, …)
- Machine translation (Google, DeepL, …)

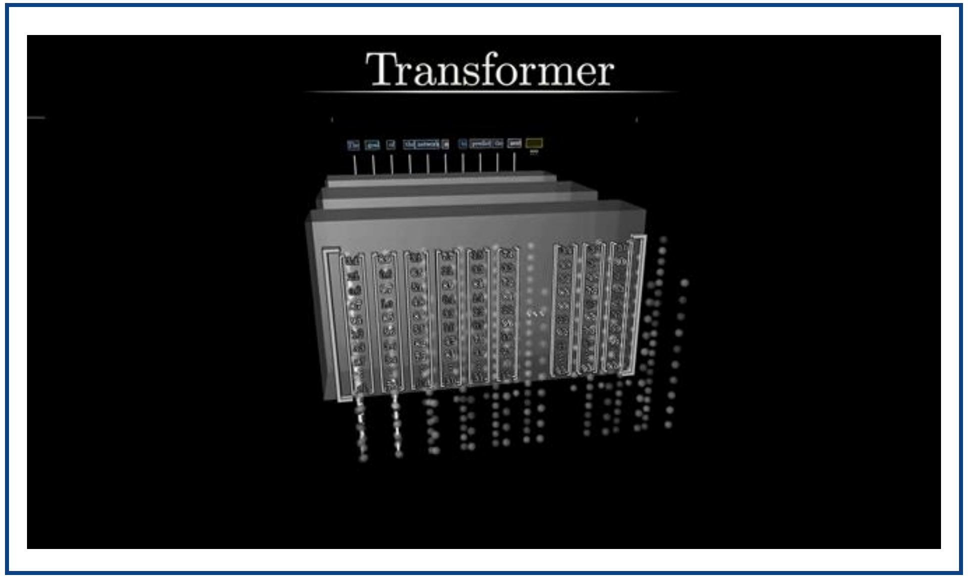
Discussion - Examples of text processing applications (GenAI)

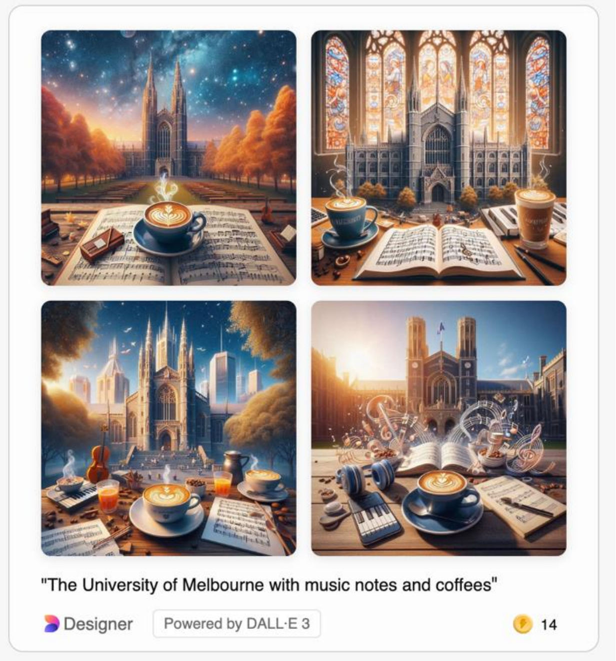
text-to-image
text-to-code
text-to-speech
speech-to-text
⋮



generate an image Tab

“The University of Melbourne with music notes and coffee”



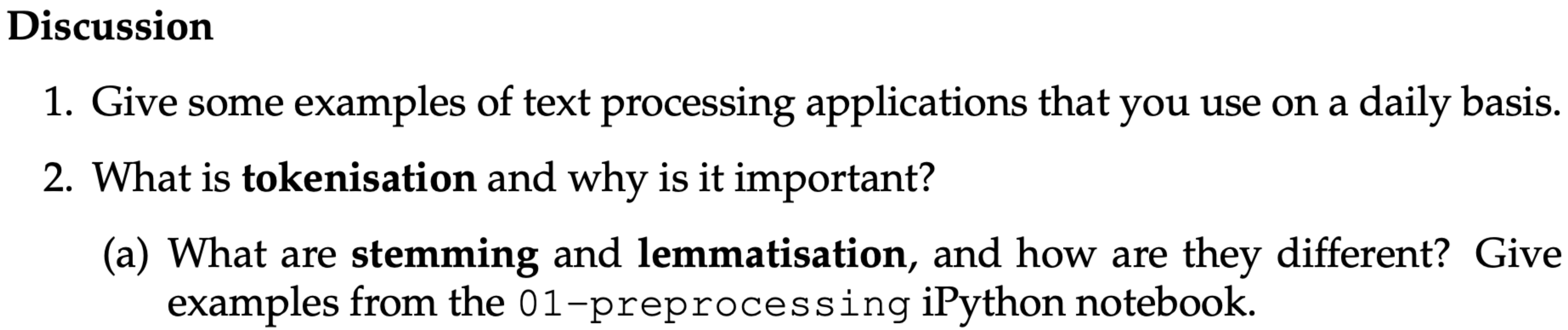


"The University of Melbourne with music notes and coffees"

Designer Powered by DALL·E 3 14

- Image: But what is a GPT? Visual intro to transformers (3B1B) [Youtube], Generated by DALL-E3

Discussion



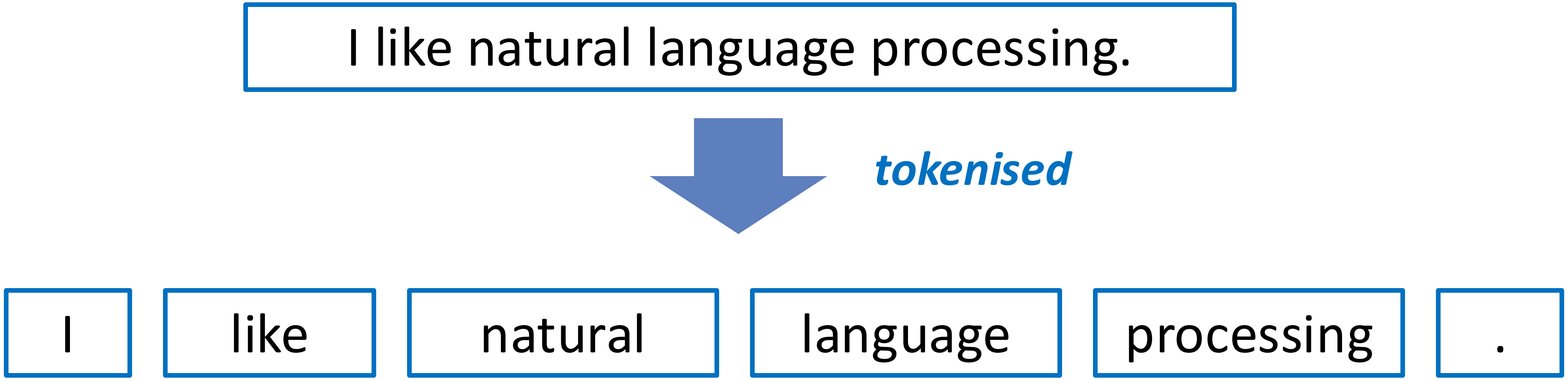
Discussion

1. Give some examples of text processing applications that you use on a daily basis.
2. What is **tokenisation** and why is it important?
   (a) What are **stemming** and **lemmatisation**, and how are they different? Give examples from the `01-preprocessing` iPython notebook.

Discussion - Word Tokenisation

What is tokenisation?

- Segmenting text into tokens (words/subwords)



I like natural language processing.

tokenised

I | like | natural | language | processing | .

Discussion - Word Tokenisation

What is tokenisation?

- Segmenting text into tokens (words/subwords)

Why is it important?

- A document may too long to manipulate directly.
- Easier for machine to understand.
- Human can break it into individual components, machine should do the same.

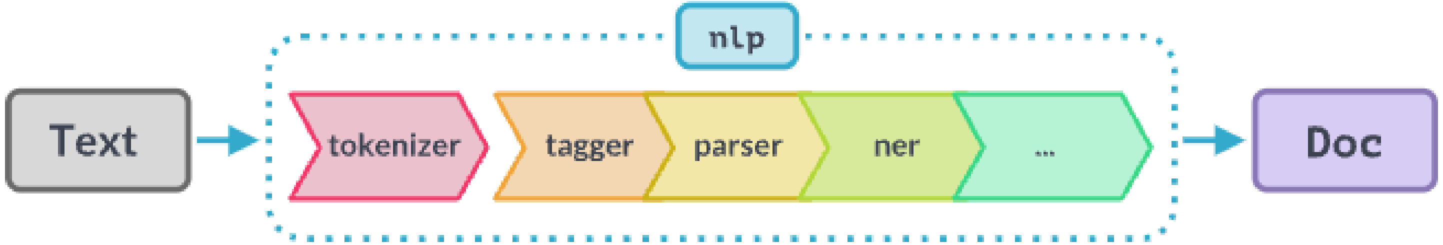


Figure : https://spacy.io/usage/linguistic-features

Discussion - Word Tokenisation

What are stemming and lemmatisation?

(Quiz)

- Computers -> Computer : Lemmatisation *removing any inflection to reach the uninflected form, the **lemma***
- Computers -> Comput : Stemming *strips off all suffixes, leaving a **stem***

What are Pros & Cons?

- Pros (+): Keep the **semantic information** at some level *(meaning of words and phrases)*
  Keep **efficiency** for the downstream task
- Cons (-): Loss of **contextual information** *(surrounding environment that influence meaning)*

Discussion - Word Tokenisation

Inflectional and Derivational Morphology

"form, shape"

***->** the study of the internal structure of words*

(e.g.) Comput -> Comput**ers**

- -s: **inflectional** morphology

  *is the systematic process by which tokens are **altered** to conform to certain **grammatical constraints***

  (e.g.) teacher (singular) -> teachers (plural)

- -er: **derivational** morphology

  *is the (semi-)systematic process by which we **transform** terms of one class **into a different class**.*

  (e.g.) teach (verb) -> teacher (noun)

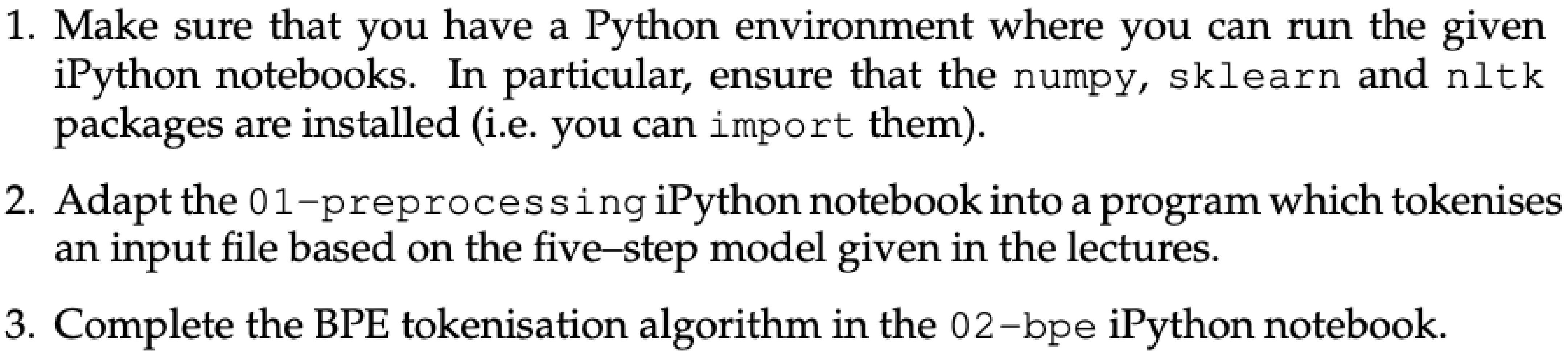
Discussion - Word Tokenisation

How are they different?

- **Lemmatisation :** removing any inflection to reach the uninflected form, the *lemma*
- **Stemming** :strips off all suffixes, leaving a *stem*

	Stemming	*Lemmatisation*
*May output **garbage tokens***	✓	
*Remove **inflectional** morphology*	✓	✓
*Usually remove **derivational** morphology*	✓	
*Works with a **lexicon** (a list of valid words)*		✓
*Remove or replace **affixes** (primarily **suffixes**)*	✓	✓
*Transform a token into a **normalised** form*	✓	✓

Programming!

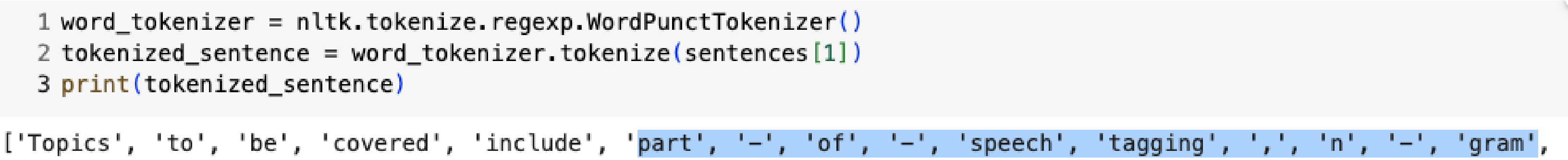


1. Make sure that you have a Python environment where you can run the given iPython notebooks. In particular, ensure that the `numpy`, `sklearn` and `nltk` packages are installed (i.e. you can `import` them).
2. Adapt the `01-preprocessing` iPython notebook into a program which tokenises an input file based on the five–step model given in the lectures.
3. Complete the BPE tokenisation algorithm in the `02-bpe` iPython notebook.

Stemming and Lemmatisation

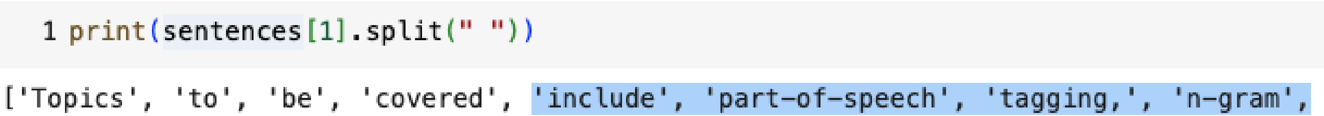
Examples from the `01-preprocessing` notebook

```
'Topics to be covered include part-of-speech tagging, n-gram language modelling,
syntactic parsing and deep learning.'
```



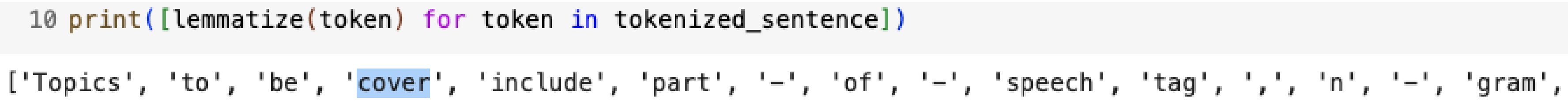
```
word_tokenizer = nltk.tokenize.regexp.WordPunctTokenizer()
tokenized_sentence = word_tokenizer.tokenize(sentences[1])
print(tokenized_sentence)
```

```
['Topics', 'to', 'be', 'covered', 'include', 'part', '-', 'of', '-', 'speech', 'tagging', ',', 'n', '-', 'gram',
```



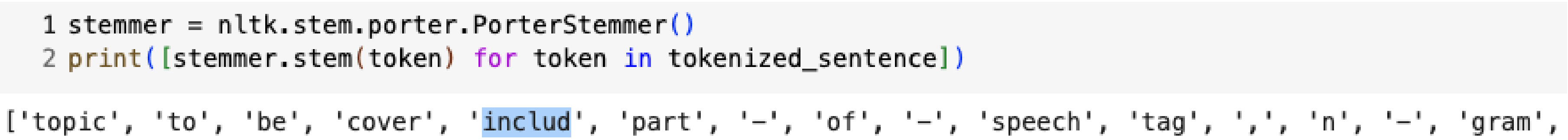
```
print(sentences[1].split(" "))
```

```
['Topics', 'to', 'be', 'covered', 'include', 'part-of-speech', 'tagging,', 'n-gram',
```



```
print([lemmatize(token) for token in tokenized_sentence])
```

```
['Topics', 'to', 'be', 'cover', 'include', 'part', '-', 'of', '-', 'speech', 'tag', ',', 'n', '-', 'gram',
```



```
stemmer = nltk.stem.porter.PorterStemmer()
print([stemmer.stem(token) for token in tokenized_sentence])
```

```
['topic', 'to', 'be', 'cover', 'includ', 'part', '-', 'of', '-', 'speech', 'tag', ',', 'n', '-', 'gram',
```

https://www.nltk.org/api/nltk.tokenize.html

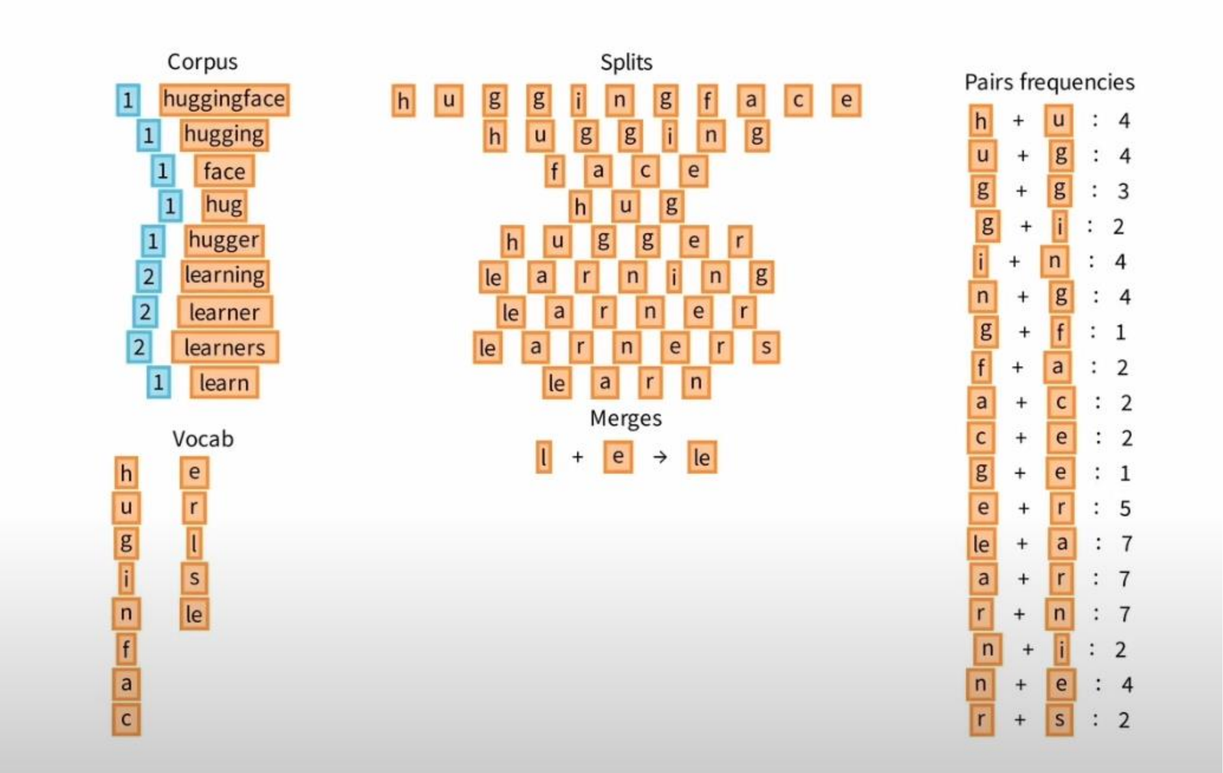
Byte-Pair Encoding (BPE)

What is Byte-Pair Encoding (BPE)?

- a subword tokenization algorithm
- iteratively merge frequent pairs of characters

Advantage of BPE?

- Data-informed tokenisation
- Works for different languages
- Deals better with unknown words



Corpus

1	huggingface
1	hugging
1	face
1	hug
1	hugger
2	learning
2	learner
2	learners
1	learn

Splits

h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
le a r n i n g
le a r n e r
le a r n e r s
le a r n

Merges

l + e → le

Vocab

h u g i n f a c e r l s le

Pairs frequencies

h + u : 4
u + g : 4
g + g : 3
g + i : 2
i + n : 4
n + g : 4
g + f : 1
f + a : 2
a + c : 2
c + e : 2
g + e : 1
e + r : 5
le + a : 7
a + r : 7
r + n : 7
n + i : 2
n + e : 4
r + s : 2

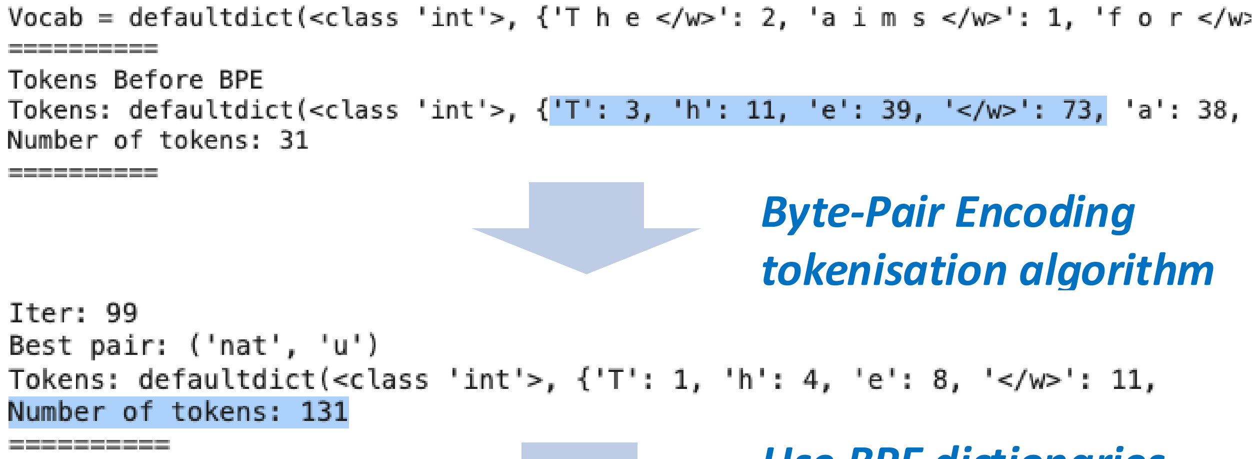
https://huggingface.co/learn/nlp-course/en/chapter6/5

https://youtu.be/HEikzVL-lZU?si=1Y02D59TokID48DQ

Byte-Pair Encoding (BPE)

Examples from the `02-bpe` notebook

```
text = "The aims for this subject is for students to develop an understanding of the
main algorithms used in natural language processing,…”
```



```
Vocab = defaultdict(<class 'int'>, {'T h e </w>': 2, 'a i m s </w>': 1, 'f o r </w
==========
Tokens Before BPE
Tokens: defaultdict(<class 'int'>, {'T': 3, 'h': 11, 'e': 39, '</w>': 73, 'a': 38,
Number of tokens: 31
==========
```

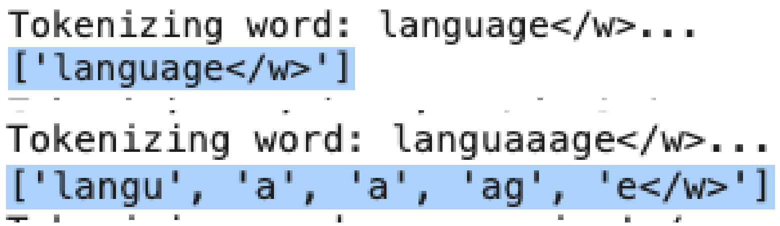
Byte-Pair Encoding tokenisation algorithm

```
Iter: 99
Best pair: ('nat', 'u')
Tokens: defaultdict(<class 'int'>, {'T': 1, 'h': 4, 'e': 8, '</w>': 11,
Number of tokens: 131
==========
```

```
def get_vocab(text):
def get_stats(vocab):
def merge_vocab(pair, v_in):
def get_tokens(vocab):
num_merges = 100
```

Use BPE dictionaries to tokenise sentences

```
sentence_1 = 'I like natural language processing!’
sentence_2 = 'I like natural languaaage processing!'
```



```
Tokenizing word: language</w>...
['language</w>']
Tokenizing word: languaaage</w>...
['langu', 'a', 'a', 'ag', 'e</w>']
```