Huggingface Transformers: A short introduction

CSC401/2511 – Natural Language Computing – Winter 2023

University of Toronto

Logistics

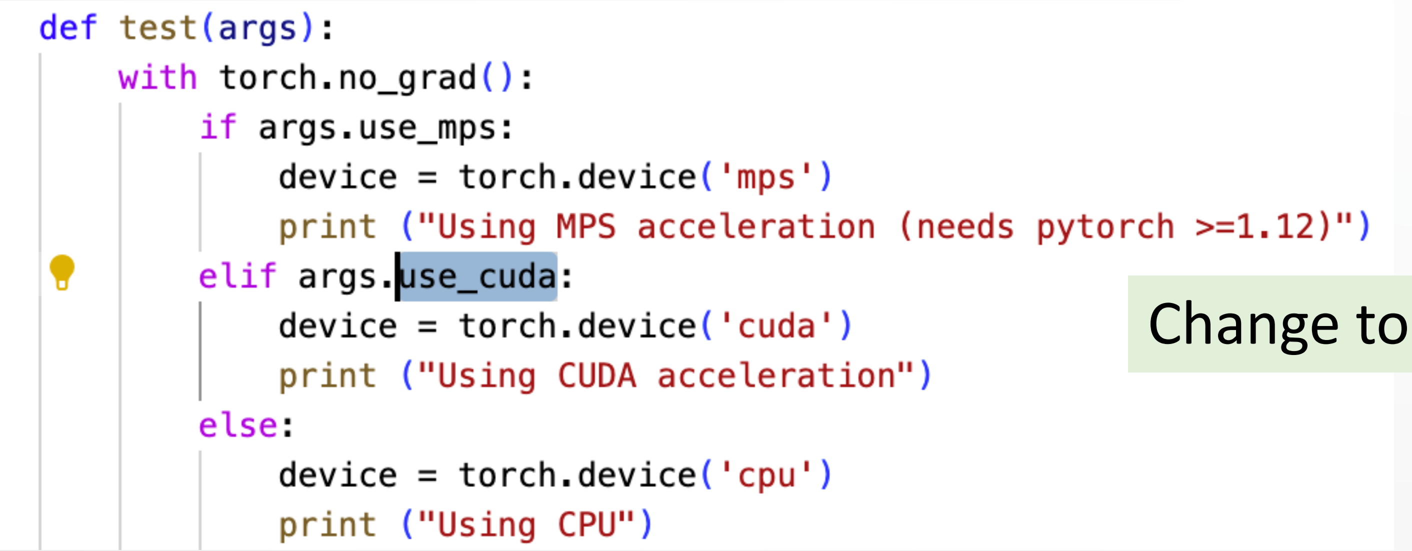
- Today’s lecture will only last 35 minutes
  - 10am session: The last 15 minutes is a survey.
  - 11am session: The first 15 minutes is a survey.

- Contents: sentiment analysis with a huggingface model.
  - I’ll introduce some key features of huggingface.

- After today’s lecture, you will be able to start working on Assignment 3.

Assignment 3 update 1: cuda

- In the test() function for classifier.py: change args.use_cuda to args.use_gpu



```
def test(args):
    with torch.no_grad():
        if args.use_mps:
            device = torch.device('mps')
            print ("Using MPS acceleration (needs pytorch >=1.12)")
        elif args.use_cuda:
            device = torch.device('cuda')
            print ("Using CUDA acceleration")
        else:
            device = torch.device('cpu')
            print ("Using CPU")
```

Change to **use_gpu**

Assignment 3 update 2: package

- Currently, the package importlib-metadata is not present on the wolf server – I asked the instruction support to install.

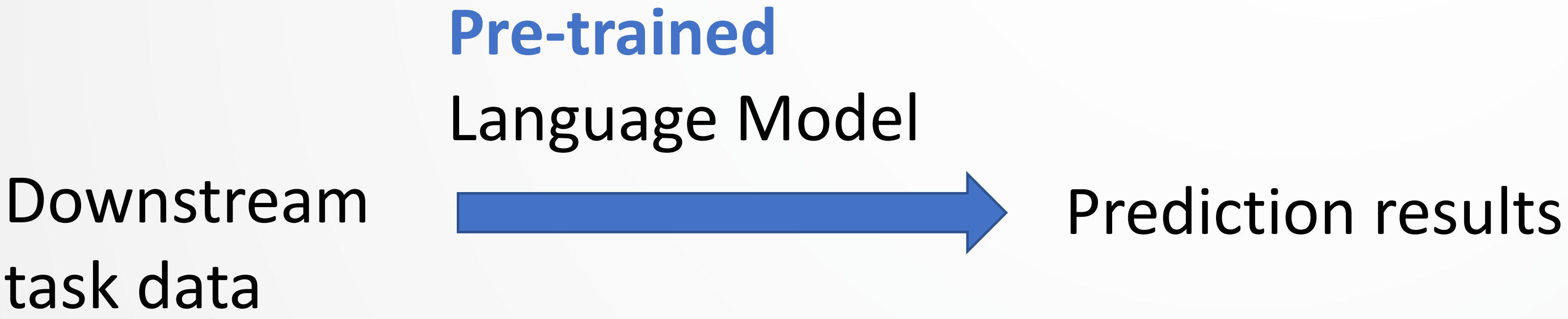
- A walkaround is to modify the lines in utils.py:
  - **Comment out** line 14 “import importlib_metadata”
  - Change line 20 into _torch_version = torch.__version__

Recap: Sentiment Analysis

- Is this IMDB movie review a positive one?

This is not a movie for fans of the usual eerie Lynch stuff. Rather, it's for those who either appreciate a good story, or have grown tired of the run-of-the-mill stuff with overt sentimentalism […]<br /><br />The story unfolds flawlessly, and we are taken along a journey that, I believe, most of us will come to recognize at some time. A compassionate, existentialist journey where we make amends for our past when approaching ourt inevitable demise.<br /><br />Acting is without faults, cinematography likewise (occasionally quite brilliant!), and the dialogue leaves out just enough for the viewer to grasp the details od the story.<br /><br />A warm movie. Not excessively sentimental.

Recap: DNN-based NLU



Pre-trained

Language Model

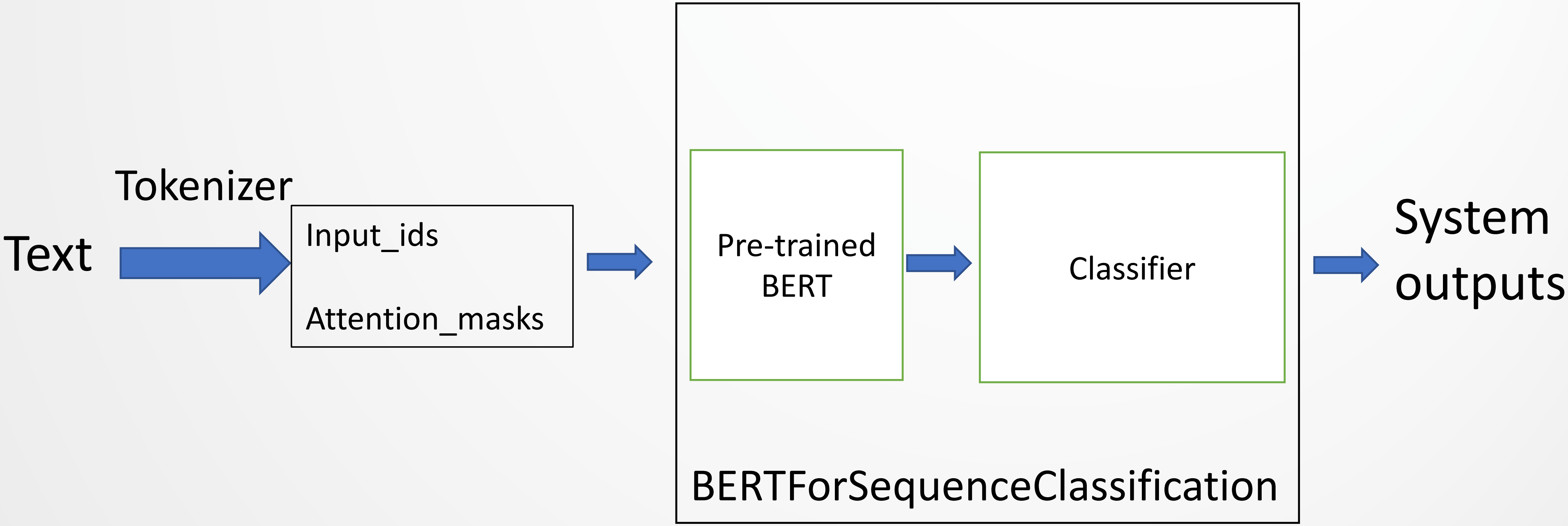
Downstream task data

Prediction results

Huggingface Transformers provides a convenient workflow for building DNN-based NLU systems.

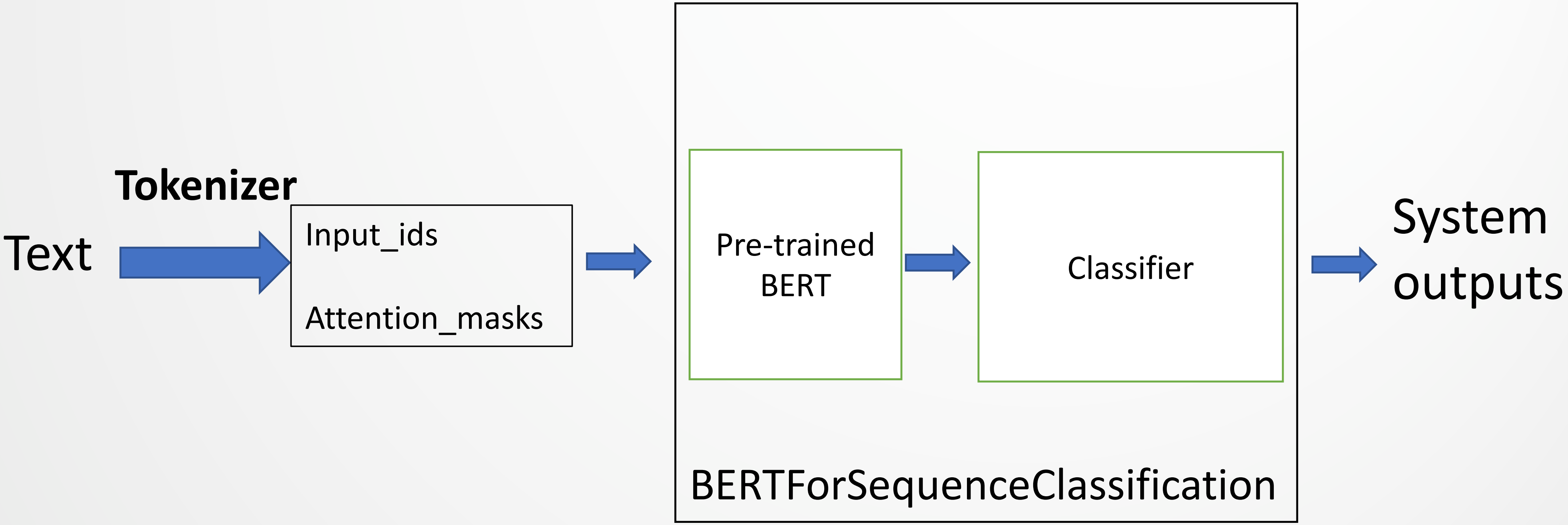
Overview of the pipeline

- An overview of the pipeline that you will use for A3:



Overview of the pipeline

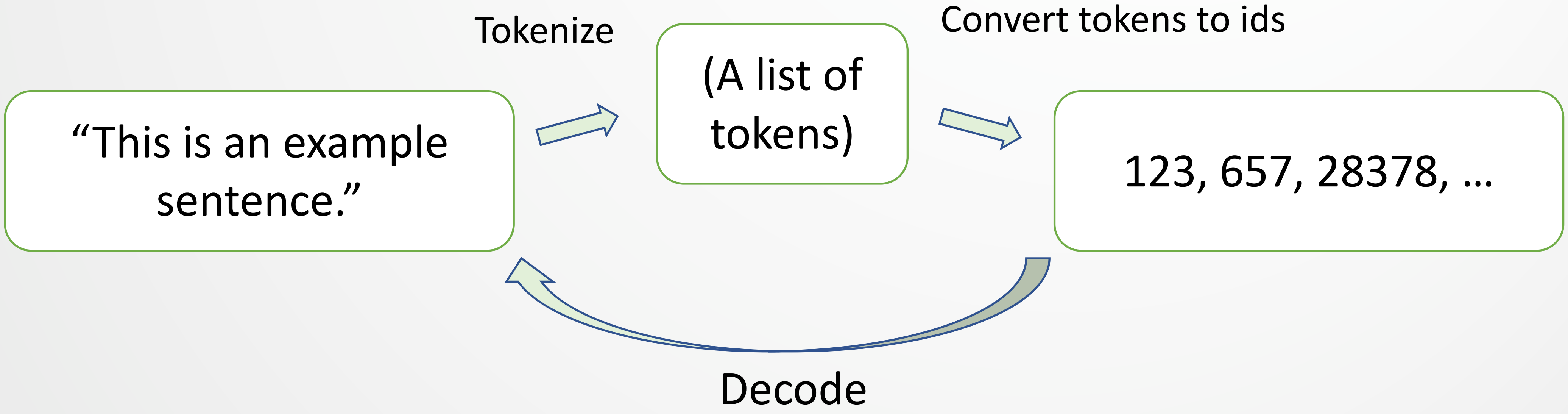
- An overview of the pipeline that you will use for A3:



Tokenizer

NLP systems need a **tokenizer** to **encode** texts into numbers.

Encode = **tokenize**, and then **convert_tokens_to_ids**



Tokenize

Convert tokens to ids

“This is an example sentence.”

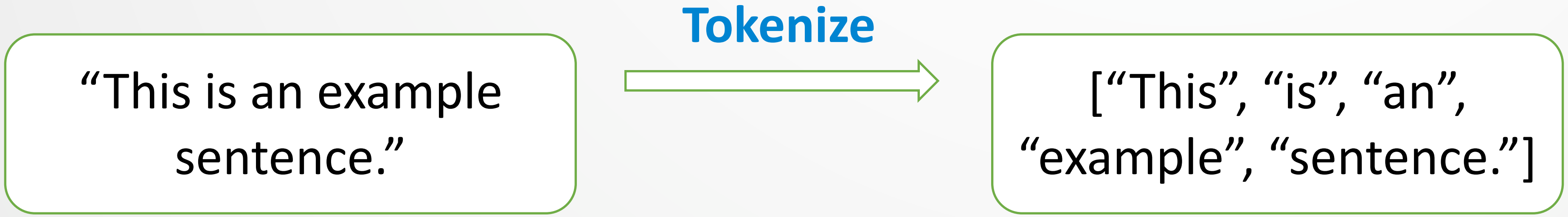
(A list of tokens)

123, 657, 28378, …

Decode

Tokenization: word splitting

- Method 1: .split(), then look up the word index in a dictionary.
  - Words with the same lemma forms are considered as different words.
    E.g., “convert” vs “converts”
  - Punctuations are not handled well.
    E.g., “The end of a sentence. The start of the other”



“This is an example sentence.”

Tokenize

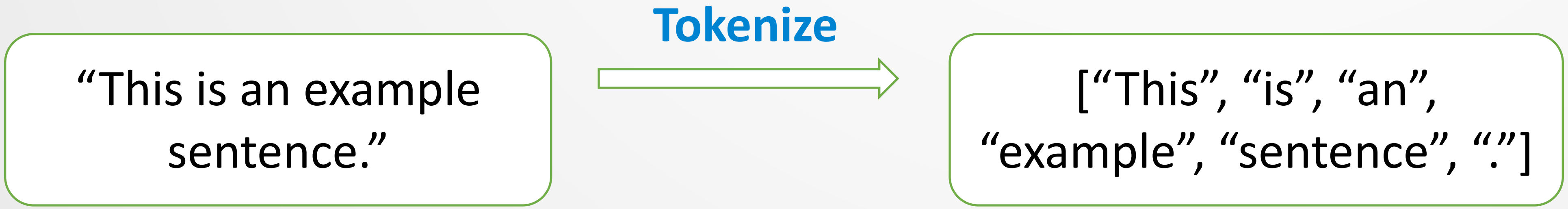
[“This”, “is”, “an”, “example”, “sentence.”]

Tokenization: better word splitting

- Method 2: Separate the words and the punctuations, then do .split(), then look up the word index from the dictionary.
  - Still, “convert” and “converts” are treated as different words.
  - The vocabulary sizes are unnecessarily large.
  - In multilingual tasks, the vocabulary sizes are even larger.

…although many English words have the same roots.

Some examples: geography, bibliography



“This is an example sentence.”

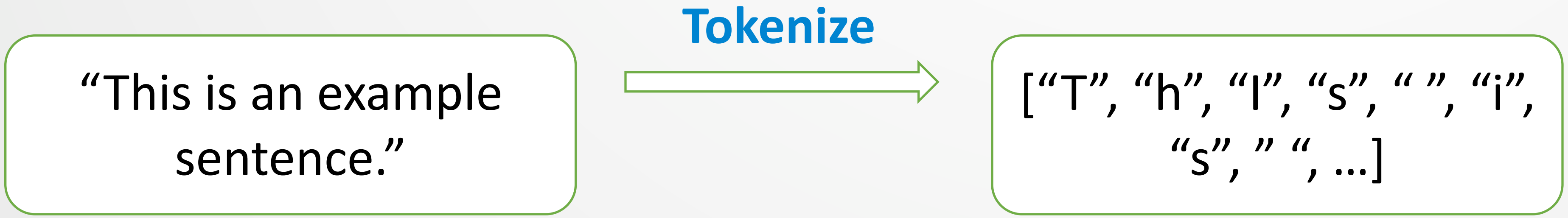
Tokenize

[“This”, “is”, “an”, “example”, “sentence”, “.”]

Tokenization: character encoding

- Method 3: Character / Byte-level encoding
  - Example: CANINE (Lecture 7)
  - The vocabulary size is significantly reduced.
  - but how long are your sequence going to be?

- Can we strike a balance between character-level encoding and word-level encoding?



“This is an example sentence.”

Tokenize

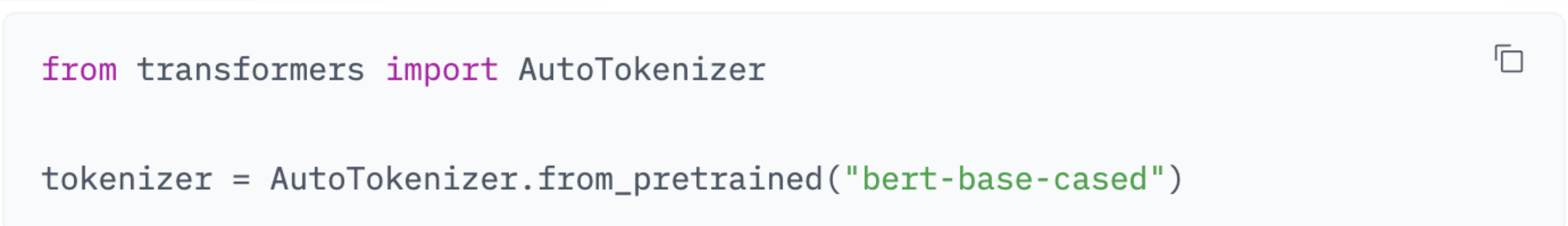
[“T”, “h”, “I”, “s”, “ ”, “i”, “s”, ” “, …]

Tokenization: subword

- Method 4: subword.
  - This is adopted by popular LMs, including BERT and *GPT.
  - The words to split, and the methods of splitting, differs.
    In CSC401/2511: don’t worry about that ^.
  - Each pretrained language model comes with its own tokenizer.

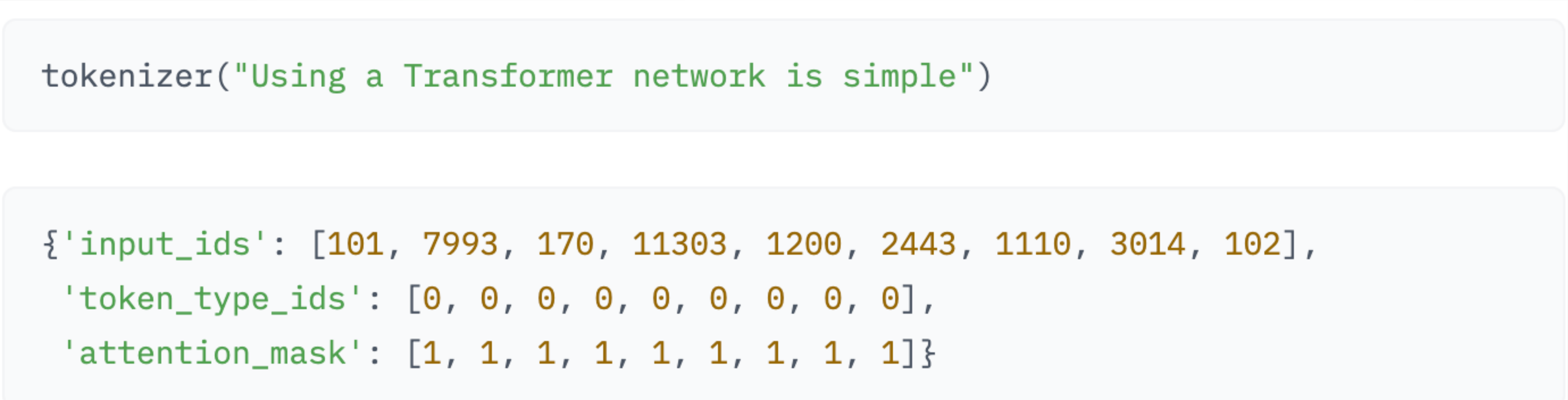
Let’s</w>	do</w>	token	ization</w>	!</w>

Loading and using the tokenizer



```python
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```



```python
tokenizer("Using a Transformer network is simple")
```

```
{'input_ids': [101, 7993, 170, 11303, 1200, 2443, 1110, 3014, 102],
 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0],
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

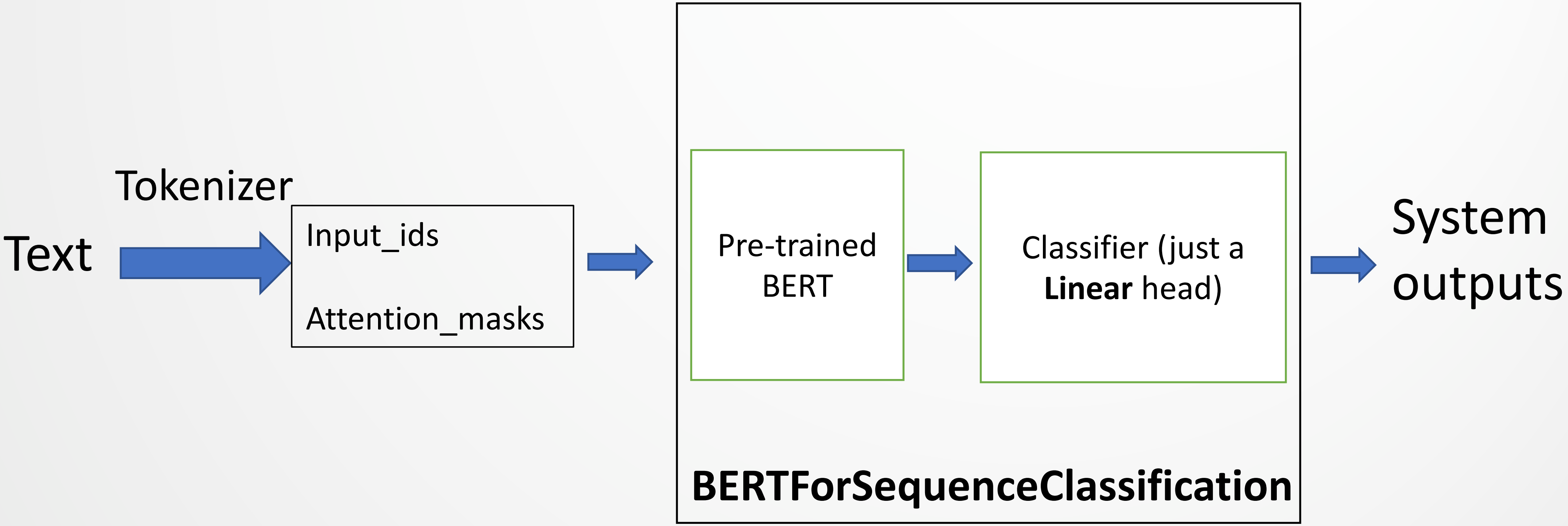
Two-step encoding process

- Calling tokenizer(sentence) is equivalent to:
  - tokens = tokenizer.tokenize(sentence), and then:
  - tokenizer.convert_tokens_to_ids(tokens)

- Details will be presented in Friday’s tutorial.

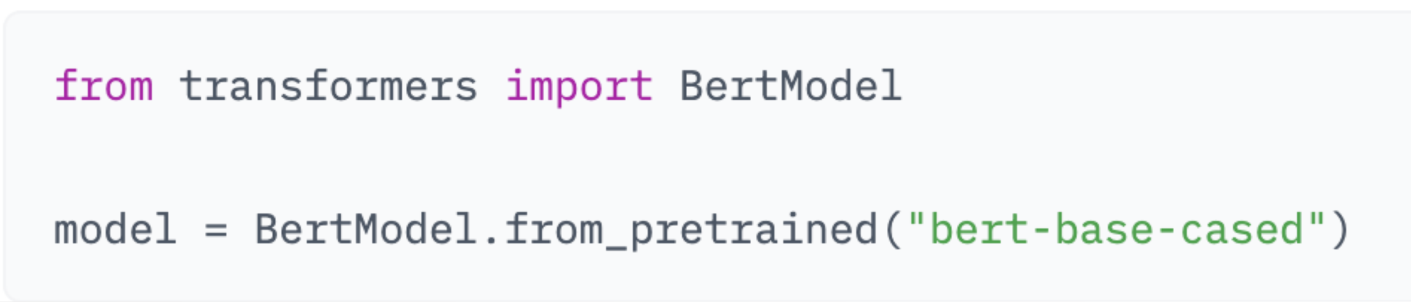
Overview of the pipeline

- An overview of the pipeline



Tokenizer
Text
Input_ids
Attention_masks
Pre-trained BERT
Classifier (just a **Linear** head)
BERTForSequenceClassification
System outputs

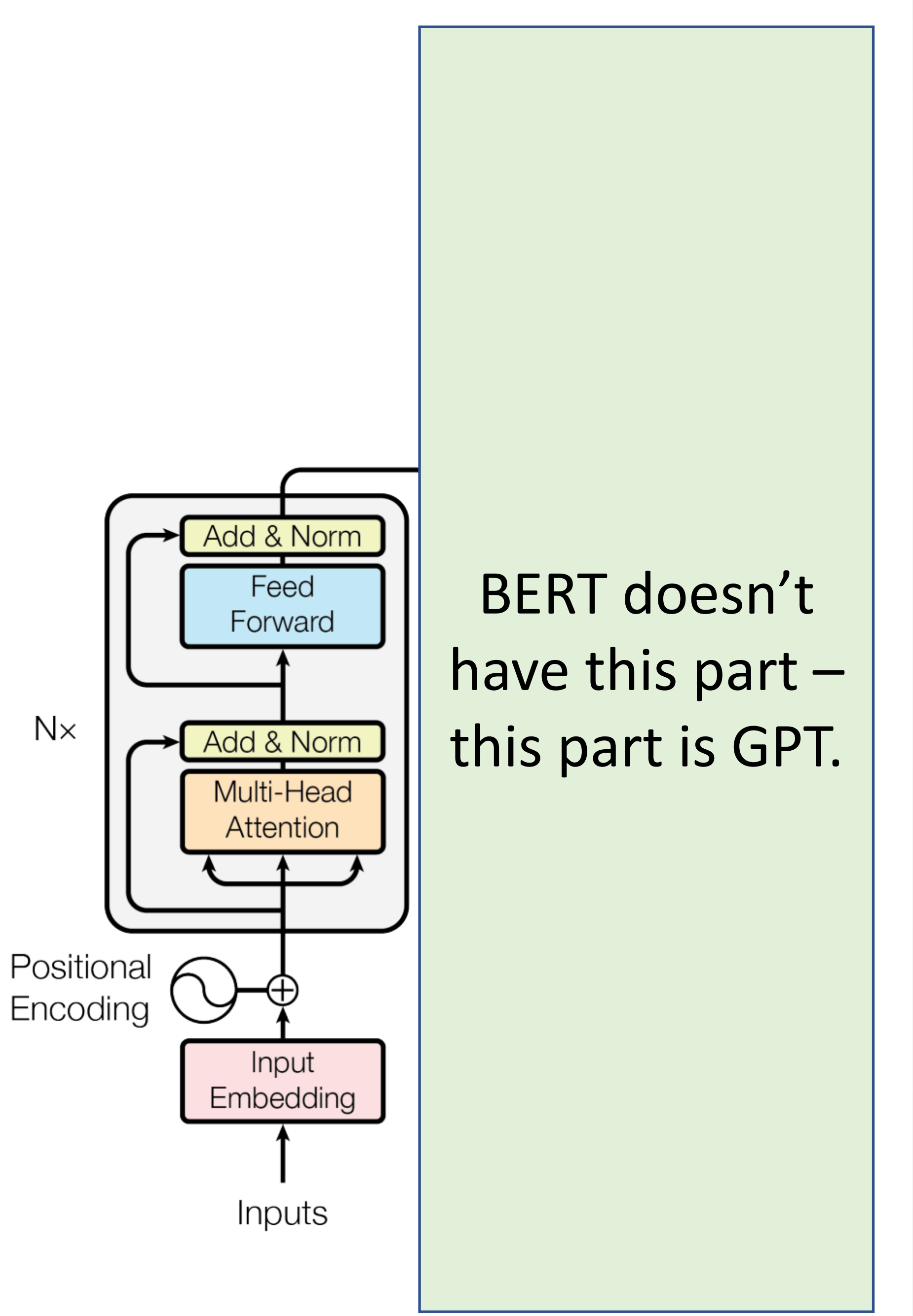
BERTmodel



```python
from transformers import BertModel

model = BertModel.from_pretrained("bert-base-cased")
```

- BERTmodel is the encoder part of the Transformer:
  - Also ref: Lecture 7



BERT doesn’t have this part – this part is GPT.

Figure 1: The Transformer - model architecture.

Lecture review questions

By the end of this lecture, you should be able to:

- Describe what is tokenization.
  - Use huggingface's tokenizer
- Describe a BERT for Sequential Classifier system.
- Start working on Q3 and Q4 in Assignment 3.
  - Friday's tutorial will also be helpful for Q3.
  - Q2: Not yet. Speech Recognition is in next week.

Anonymous feedback form: https://forms.gle/W3i6AHaE4uRx2FAJA