Task 1. Natural Language Processing. Named entity recognition

In this task, we need to train a named entity recognition (NER) model for the identification of mountain names inside the texts

```
text = "Everest is a mountain."
text_to_tags(text)
```

```
{'Everest': 'mountain',
 'is': 'other',
 'a': 'other',
 'mountain': 'other',
 '.': 'other'}
```

```
text = "Mountains."
text_to_tags(text)
```

```
{'Mountains': 'mountain', '.': 'other'}
```

In the earlier example, 'Classification Mountains' denoted the name of a mountain range, requiring both words to be tagged as 'mountain.' In this sentence, the term 'mountain' does not signify a specific mountain name but rather represents a category. Therefore, it is appropriate that the model does not label it as a 'mountain.'

```
text = "Mountains can be of varying height."
text_to_tags(text)
```

```
{'Mountains': 'other',
 'can': 'other',
 'be': 'other',
 'of': 'other',
 'varying': 'other',
 'height': 'other',
 '.': 'other'}
```

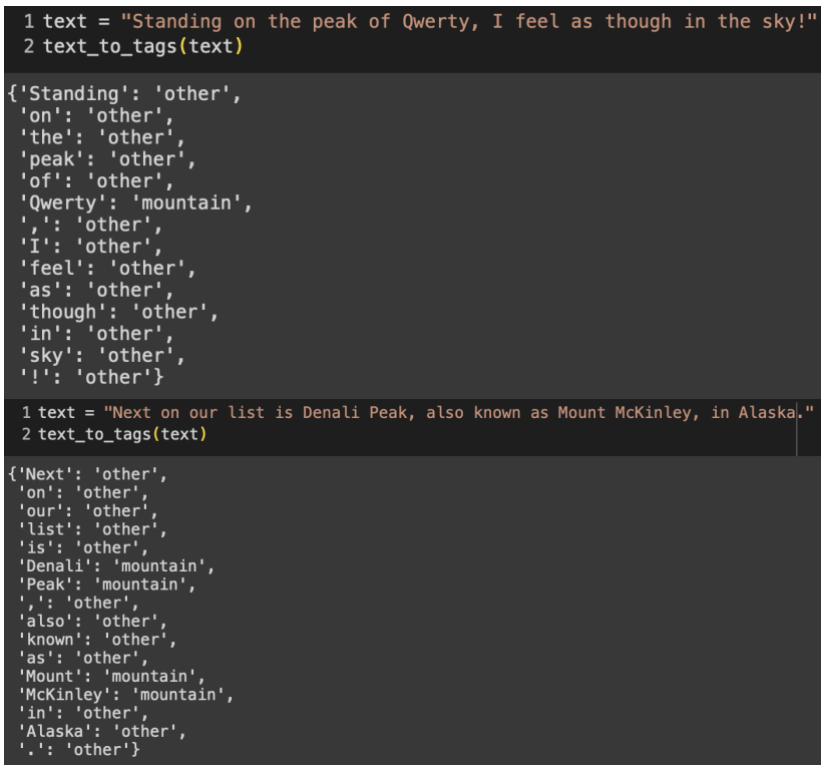
```
text = "Finally, at the top of our list is Mount Fuji in Japan."
text_to_tags(text)
```

```
{'Finally': 'other',
 ',': 'other',
 'at': 'other',
 'the': 'other',
 'top': 'other',
 'of': 'other',
 'our': 'other',
 'list': 'other',
 'is': 'other',
 'Mount': 'mountain',
 'Fuji': 'mountain',
 'in': 'other',
 'Japan': 'other',
 '.': 'other'}
```

```
text = "I visited the Classification Mountains when I was a child."
text_to_tags(text)
```

```
{'I': 'other',
 'visited': 'other',
 'the': 'other',
 'Classification': 'mountain',
 'Mountains': 'mountain',
 'when': 'other',
 'was': 'other',
 'a': 'other',
 'child': 'other',
 '.': 'other'}
```

I've successfully tricked the model. The term 'Mountains' indicates a specific mountain name only when accompanied by another word, as in the case of the Rocky Mountains. Without contextual clues, it's reasonable to interpret it as a general category. The likely factor influencing this behavior is the capitalization. As demonstrated below, the same sentence with 'mountains' in lowercase produces the correct result.



```
1 text = "Standing on the peak of Qwerty, I feel as though in the sky!"
2 text_to_tags(text)
```

```
{'Standing': 'other',
 'on': 'other',
 'the': 'other',
 'peak': 'other',
 'of': 'other',
 'Qwerty': 'mountain',
 ',': 'other',
 'I': 'other',
 'feel': 'other',
 'as': 'other',
 'though': 'other',
 'in': 'other',
 'sky': 'other',
 '!': 'other'}
```

```
1 text = "Next on our list is Denali Peak, also known as Mount McKinley, in Alaska."
2 text_to_tags(text)
```

```
{'Next': 'other',
 'on': 'other',
 'our': 'other',
 'list': 'other',
 'is': 'other',
 'Denali': 'mountain',
 'Peak': 'mountain',
 ',': 'other',
 'also': 'other',
 'known': 'other',
 'as': 'other',
 'Mount': 'mountain',
 'McKinley': 'mountain',
 'in': 'other',
 'Alaska': 'other',
 '.': 'other'}
```

I employed the dslim/bert-large-NER model and tokenizer in my solution. The training process involved:

- Tokenization: I tokenized the texts, converting existing word tokens into subword tokens suitable for the model. Adjustments to labels were made accordingly.
- Metrics: Due to data imbalance, particularly with mountain names being less frequent, I utilized the F1 Score as my evaluation metric.
- Class weights: To address data class imbalance, I implemented a custom trainer that applied balanced class weights.
- Training: The model underwent training for 5 epochs with a learning rate set at 2e-5.