Homework 5: Sequence Labeling

Due March 3, 2022 (11:59 PM).

1 Introduction and Task

For this homework, we'll provide you with a neural sequence labeling model, an implementation of **BERT** (Devlin et al. 2019) from the **Transformers** library (Turc et al. 2019). Your task is to extract span labels from the token-level predictions and calculate span F1. Refer to the 2/24 lecture on sequence labeling for conceptual help on the assignment.

The Colab notebook for this homework is located here: https://github.com/dbamman/nlp22/tree/main/HW5/HW_5.ipynb

2 NER and BIO Notation

NER

In class we've explored different ways to apply useful labels to text in the course of natural language processing tasks. If you are analyzing syllabi from 100 different Berkeley courses, you might carry out part-of-speech tagging and compare the different adjectives that instructors use.

But what if you wanted to identify all of the *entities* (persons, locations, organizations) mentioned in those syllabi? The goal of **Named Entity Recognition (NER)** is to identify **spans** of text that constitute proper names and tag the type of the entity (Jurafsky and Martin, 2021). Some example tags include `PER` (People) and `LOC` (Location).

This is a challenging task. To start, most words in a text will not be named entities which limits the size of relevant training data. Also, we're faced with perplexing ambiguities — e.g. is a mention of “Reagan” referring to the Washington, D.C. airport or the United States' 40th president? Finally, unlike parts of speech tagging, where we know that we will tag each word in our input with just one, single label, NER requires us to segment text into *spans* of indeterminate lengths and label those spans.

To summarize, to carry out NER, we have need to decide:

1. what's an entity and what isn't
2. where the boundaries are

BIO Notation

The standard method to sequence labeling for NER is **BIO** tagging (Ramshaw and Marcus, 1995). This method allows us to treat NER like a word-by-word sequence labeling task, with tags that capture both the boundary and the named entity type.

BIO notation assigns a label to every word marking its relation to a given entity. There are three possibilities:

1. Beginning of Entity `B`
2. Inside Entity `I`

3. Outside Entity `O`

We label any token that *begins* a span of interest with the label `B`. Tokens that occur *inside* a span are tagged with an `I`. Lastly, there is only one `O` tag, which generically captures any tokens outside of any span of interest. These entity "prefix" tags (`B` and `I`) are then combined with one of the NER type tags that we saw above (e.g. `PER`, `LOC`) to capture both the boundary and the named entity type.

As an example, let's take a sentence like:

tim cook is the ceo of apple.

The overall goal of NER is to create this representation:

[tim cook]$_{\text{per}}$ is the ceo of [apple]$_{\text{org}}$

We can use BIO notation to label the sentence like so:

B-PER	I-PER	O	O	O	O	B-ORG
tim	cook	is	the	ceo	of	apple

Why is this the case? Each word in the input was tagged with its requisite label:

`Tim` is the *Beginning* of Entity so it gets a `B` prefix. And it refers to a person, so `PER` is the type label. `Cook` refers back to the same entity, `Tim`, so it's an *Inside* entity and gets a `I` prefix. Moving through the rest of the sentence, the last entity of interest is `apple`, which refers to the software company -- not the fruit! So we'll label as the *Beginning* of a new entity. Every other token is not a part of any entity, and so gets a label of `O`.

From this small example, you can see the complexity this task presents. For this assignment, we'll be asking you to extract the span labels that are outputted by an NER model and evaluate how well it makes predictions.

3 Deliverable 1: Span Label Extractions

The first deliverable for this homework is completing the `get_spans()` method. This method will be used evaluate the performance of our neural NER model by comparing its adjudged BIO tags to gold-standard labels.

As input, it will take in a list of strings in BIO notation. The method should parse each element and return the entities contained therein.

For example, let's say that our model doesn't realize that the "apple" in our example sentence *tim cook is the ceo of apple* refers to the software company and predicts its BIO tags as:

```
["B-PER", "I-PER", "O", "O", "O", "O", "O"]
```

Your method should return a set containing the relevant entities, where each element in the set is a tuple (*start, end, category*) containing information about one entity, where *start*=the start token position for that entity, *end*=the end token position for the entity, and *category*=its NER category. For the example above, your method should output the following set:

```
{ (0, 1, PER) }
```

This corresponds to an entity starting at token position 0, ending at token position 1 (inclusive), and of category `PER`.

4 Deliverable 2: F1

Your second deliverable is completing the `get_span_f1()` method to evaluate the performance of our model with the F1 Measure.

As input, this method will take in `sentence_preds`, a list containing every sentence tagged by the model along with its predicted and true token-level labels. As output, this method should return the overall F1 score for the entire model. For NER systems, the entity rather than the word is the unit of analysis. This means that the F1 score this method calculates represents the performance at the *span* level.

F1 is the harmonic mean of **recall** and **precision**:

$$F1 = \frac{2 * P * R}{P + R}$$

Where:

- **recall** is the ratio of the number of correctly labeled spans divided by number that should have been labeled
- **precision** is the number of corrected labeled spans divided by the number of labels that we applied

Looking back to our example sentence and tags, where we failed to label *apple* correctly, here is an example of how to calculate F1.

	0	**1**	**2**	**3**	**4**	**5**	**6**
	tim	cook	is	the	ceo	of	apple
gold	`B-PER`	`I-PER`	O	O	O	O	`B-ORG`
model	`B-PER`	`I-PER`	O	O	O	O	O

We correctly identified one of the two true spans (leading to a recall of 1/2); and for the only entity we predicted, we got it right (leading to a precision of 1/1). This works out to a span F1 score of 0.667

Finally, when part of a predicted span is correct but another part is incorrect, we consider that to be a miss rather than "partially" correct. If we take our example sentence and adjust the prediction of "cook" from `I-PER` (correct) to `B-PER` (wrong), our F1 score needs to change accordingly.

	0	**1**	**2**	**3**	**4**	**5**	**6**
	tim	cook	is	the	ceo	of	apple
gold	`B-PER`	`I-PER`	O	O	O	O	`B-ORG`
model	`B-PER`	`B-PER`	O	O	O	O	O

We know that the correct span is `{ (0, 1, PER) }`. Our model, however, predicted two separate entities: `{ (0, 0, PER), (1, 1, PER) }`. We did not correctly identify either of the two true spans (leading to a recall of 0/2); and neither of the predictions were correct (leading to a precision of 0/2). This resolves to an F1 of 0.

5 How to Submit

- Submit your work to Gradescope
  - Download your Colab notebook as a `.ipynb` file
    - (File --> Download .ipynb)
  - Submit **HW_5.ipynb**
    - Your file must be named this way for the Autograder to work
    - Please do not include print statements, test cases etc. in your final submission as this can cause issues for the Autograder
  - Recall that when the Autograder is working correctly, you'll be presented with a blank screen (that is ok!)