Homework 6: Semantic Role Labeling

Due April 7, 2021 (11:59 PM)

1 Introduction

In this homework, you will be using a BERT-based classifier to perform semantic role labeling (SRL).

We will be using PropBank data for this assignment, which consists of sentences (from the Penn Treebank) annotated with proto-roles that are verb-specific. In general, `ARG0` represents the proto-agent and `ARG1` the proto-patient, while the semantics of other roles are somewhat less consistent and more verb-specific. For this assignment, you will be training a system that predicts whether an argument in a given predicate-argument pair is `ARG0`, `ARG1`, or `O` (neither).

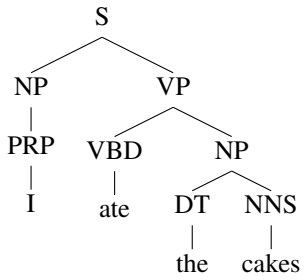
Similarly to HW4, we'll be using the Transformers Python library. The Colab notebook for this homework is located here:
https://github.com/dbamman/nlp21/blob/main/HW6/HW_6.ipynb

2 Deliverable 1: Complete `BERTSRLClassifier`

The first task is to predict whether a given candidate phrase is an `ARG0`, `ARG1` or `O`, with respect to a given predicate. As an example, consider the following sentence, where the target predicate *ate* is underlined:

I ate the cakes

The phrase structure tree for this sentence looks like the following:



For this task, you are given one candidate phrase from this tree (e.g., the NP corresponding to *the cakes*) and will classify that candidate as being either `ARG0`, `ARG1` or neither (`O`) with respect to the target predicate (here, *ate*).

To do this, a class called `BERTSRLClassifier` has been provided for you. This BERT-based classifier takes in the words of a sentence along with information about the beginning and end of an argument span and predicate to classify

those predicate-specific arguments into `ARG0`, `ARG1` or `O`. The location information is provided to you in the form of the WordPiece token positions (remember that BERT uses WordPiece tokenization to break down complex words into smaller subpieces). The WordPiece tokenization for the sentence above would be the following:

$$[\text{CLS}]_0\ \text{I}_1\ \underline{\text{ate}}_2\ \text{the}_3\ \text{cake}_4\ \#\text{s}_5\ [\text{SEP}]_6$$

In making a decision about the label for *the cakes* with respect to *ate*, you are given the following position information:

- Predicate start WordPiece token position: 2
- Candidate span start WordPiece token position: 3
- Candidate span end WordPiece token position: 5

Using this information, You will make a prediction about a candidate/predicate pair by concatenating the 768-dimensional BERT vectors indexed by these positions, passing them through a linear transformation into the size of the 3-dimensional output space (for the three classes `ARG0`, `ARG1` and `O`) and passing that through a softmax function to yield a probability distribution over the space.

Formally, let b be the final BERT layer (for this example, $b \in \mathbb{R}^{7\times 768}$) and let W be a linear layer ($\in \mathbb{R}^{3*768\times 3}$). For this one example:

$$o = \text{concat}[b_2; b_3; b_5] \tag{1}$$

$$y = \text{softmax}\left(o^\top W\right) \tag{2}$$

Your deliverable is to complete the indicated section of the 'forward' function in `BERTSRLClassifier` to reflect this model. Code for reading in the data and training the model has been provided for you, along with an example of how to index into multidensional pytorch tensors.

(Note: don't overthink this one; the solution here is quite simple, and involves only a few lines of code).

3 Deliverable 2: Find `ARG0` from a list of all non-terminal phrases in a parse tree

The code above lets you classify the `ARG0`/`ARG1`/`O` status of a given candidate with respect to a predicate, but it doesn't directly tell you what the best `ARG0` for a given predicate is; one way of doing that would be to perform this classification over every possible candidate in a phrase structure tree and then return the `ARG0` (for example) as the candidate that has the highest `ARG0` score. In the example above (and reindexing the sentence according to its non-WordPiece tokens), that would require generating an `ARG0` score for each of the following candidates (where each candidate is given by (NP category, start token position, end token position):

$$\text{I}_0\ \underline{\text{ate}}_1\ \text{the}_2\ \text{cakes}_3$$

- (NP, 0, 0), "I"
- (PRP, 0, 0), "I"
- (VP, 1, 3), "ate the cakes"

- (VBD, 1, 1), “ate”
- (NP, 2, 3), “the cakes”
- (DT, 2, 2), “the”
- (NNS, 3, 3), “cakes”

In this section, you will be given a sentence with a list of all its non-terminal phrases (candidates), and a predicate. Using your completed `BERTSRLClassifier`, your task is to find the correct `ARG0` argument among the list of candidates, for the given sentence and predicate.

In order to do this, you would generate an `ARG0` score for all the candidates using the `BERTSRLClassifier` and select the candidate with the highest score (highest likelihood of being `ARG0`). The basic setup has been provided for you, and your deliverable is to finish the ‘predict_arg0‘ function so that it returns **the start and end position** of the `ARG0` predicted by your model (not the syntactic category). The true `ARG0` for the example above would be (0,0).

4 How to Submit

1. Download your Colab notebook as an .ipynb file (File → Download .ipynb)
2. Submit **HW_6.ipynb** to the Homework 6 assignment on Gradescope. The files must be named **HW_6.ipynb** for the Gradescope autograder.