Project 1: Recognizing Textual Entailment (80 points)

CS 6320: Natural Language Processing

Due date: 3/30/2020 02:30 pm

Problem Definition and Data

For Project-1, you will implement a deep learning model that recognizes the textual entailment relation between two sentences. Here, we are given two sentences: the premise, denoted by the letter t and the hypothesis, denoted by the letter h. We say that the premise entails the hypothesis i.e. $t \rightarrow h$ if the meaning of h can be inferred from the meaning of t [1]. To motivate the problem, consider some examples given below:

(1)
t: Eating lots of foods that are a good source of fiber may keep your blood glucose from rising too fast after you eat.
h: Fiber improves blood sugar control.
$t \rightarrow h$ as the meaning of h can be inferred from t.

(2)
t: Scientists at the Genome Institute of Singapore (GIS) have discovered the complete genetic sequence of a coronavirus isolated from a Singapore patient with SARS.
h: Singapore scientists reveal that SARS virus has undergone genetic changes
$t \not\rightarrow h$ as the meaning of h cannot be inferred from t.

The task of textual entailment is set up as a binary classification problem where, given the premise and the hypothesis, the goal is to classify the relation between them as Entails or Not Entails.

For conducting your experiments, you will use the RTE-1 dataset [2] that is provided as an addendum to this homework. The dataset contains 2 XML files: a train file and test file. The entailment relations are contained within pair tags in both files; examples of which are provided below:

```
<pair id="96" value="TRUE" task="IR">
    <t>The Massachusetts Supreme Judicial Court has cleared the way for lesbian and gay couples in
        the state to marry, ruling that government attorneys &amp;quot;failed to identify any
        constitutionally adequate reason&amp;quot; to deny them the right.</t>
    <h>U.S. Supreme Court in favor of same-sex marriage</h>
</pair>
<pair id="97" value="FALSE" task="IR">
    <t>In June the Supreme Court ruled that anti-sodomy laws are unconstitutional.</t>
    <h>U.S. Supreme Court in favor of same-sex marriage</h>
</pair>
<pair id="98" value="FALSE" task="IR">
    <t>Sharon warns Arafat could be targeted for assassination.</t>
    <h>prime minister targeted for assassination</h>
</pair>
<pair id="99" value="TRUE" task="IR">
    <t>Kurdistan Regional Government Prime Minister Dr. Barham Salih was unharmed after an
        assassination attempt.</t>
    <h>prime minister targeted for assassination</h>
</pair>
```

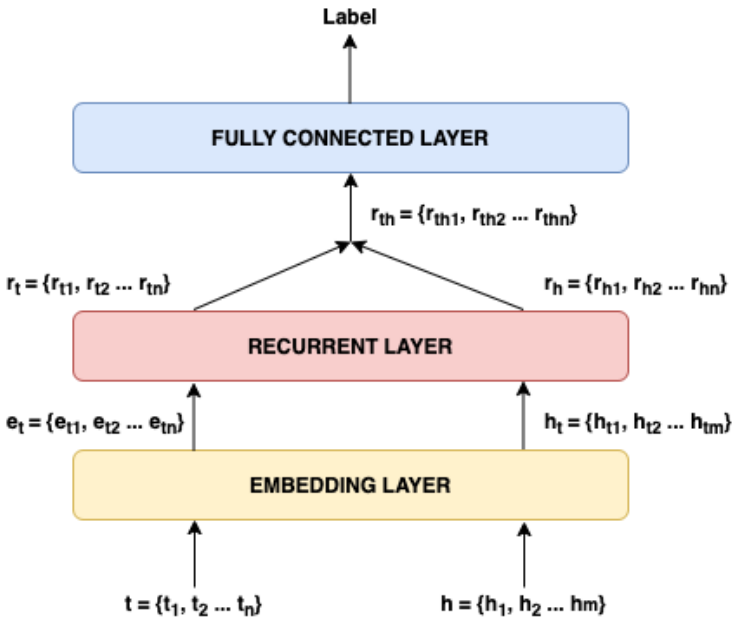
As you can observe, each pair tag contains the premise t and the hypothesis h contained within t and h tags respectively. The value attribute of the pair tag is a boolean indicating whether $t \rightarrow h$ or not.

Model Architecture

The architecture of our model for textual entailment is fairly simple. It contains the following layers:

1. Embedding layer: This layer transforms the integer-encoded representations of the sentences into dense vectors.
2. Recurrent layer: This is a stacked bi-directional LSTM layer that takes in the vector representation from the Embedding layer and outputs another vector.
3. Fully connected layer: This layer transforms the output of the RNN into a vector of 2 dimensions. (one corresponding to each label i.e. ENTAILS and NOT ENTAILS)

A schematic showing the architecture of the model is provided below:



We define the forward pass of our network as follows:

Let $t = \{t_1, t_2, ...t_n\}$ denote the premise and $h = \{h_1, h_2, ...h_n\}$ denote the hypothesis. We first obtain the dense vector representations for both sentences by passing them through the same embedding layer. Let $e_t = \{e_{t_1}, e_{t_2}, ...e_{t_n}\}$ denote the vector representations for the premise and $e_h = \{e_{h_1}, e_{h_2}, ...e_{h_n}\}$ denote the vector representations for the hypothesis where each vector e_{t_i} and e_{h_i} is of dimension d_1. Next, we pass the vector representations through the same LSTM to obtain temporal sequences r_t and r_h respectively, each vector having dimension d_2. The vectors r_t and r_h are concatenated together to obtain the vector r_{th} Finally, this concatenated representation r_{th} is passed through the fully connected layer to get vector f_{th}, with dimension $d_3 = 2$ (this is because you have 2 labels as discussed previously).

Implementation and Execution Framework

You are free to use any API like PyTorch, TensorFlow, DyNet, Caffe, etc. (with Python) for implementing this model. We recommend you use either TensorFlow or PyTorch for implementing your model as there is lot of help available online for writing code in these frameworks.
For executing your code, you will use Google Colab, a virtual environment provided by Google that allows you to edit and run your code on your browser. Additionally, Colab provides free access to GPUs making it convenient for you to train and test your model quickly.

Tasks to be performed

Here, we outline the tasks to be performed for this project.

Task - 1: Prepare dataset (15 points)

Write a method that takes in the path to (train or test) xml file as input and outputs three lists, one containing the lists of tokens for the premise, one containing the lists of tokens for the hypothesis, and one containing the label. For example, consider the given input and expected output:
Input file:

```
<pair id="593" value="TRUE" task="QA">
        <t>Oracle had fought to keep the forms from being released</t>
        <h>Oracle released a confidential document</h>
</pair>
<pair id="13" value="FALSE" task="IR">
        <t>iTunes software has seen strong sales in Europe</t>
        <h>Poor sales for iTunes in Europe</h>
</pair>
```

Output:

```
# premises list of lists
p = [['Oracle', 'had', 'fought', 'to', 'keep', 'the', 'forms', 'from', 'being', 'released'], ['
    itunes', 'software', 'has', 'seen', 'strong', 'sales', 'in', 'europe']]

# hypotheses list of lists
h = [['oracle', 'released', 'a', 'confidential', 'document'], ['poor', 'sales', 'for', 'itunes',
    'in', 'europe']]

# list of labels/classes
h = ['TRUE', 'FALSE']
```

Next, write a function to integer-encode the premises and hypotheses. In other words, replace each word in both lists by a unique integer (you may want to save this as dictionary to be used later). Likewise, integer-encode the labels also. For the same example given previously, the output will be:

```
# premises list of lists
p = [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [11, 12, 13, 14, 15, 16, 17, 18]]

# hypotheses list of lists
h = [[1, 10, 19, 20, 21], [22, 16, 23, 11, 17, 18]]

# list of labels/classes
h = [1, 2]
```

Neural networks work only when all inputs are of uniform length. Pad zeros at the end of premise and hypothesis lists so that they are of uniform length. For example, if the max allowed length is set to 10, the lists will change to:

```
# premises list of lists
p = [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [11, 12, 13, 14, 15, 16, 17, 18, 0, 0]]

# hypotheses list of lists
h = [[1, 10, 19, 20, 21, 0, 0, 0, 0, 0], [22, 16, 23, 11, 17, 18, 0, 0, 0, 0]]
```

Task - 2: Preparing the inputs for training/testing (10 points)

Look into how to create batches for training and testing your model. For example, if you are using PyTorch, you can look into TensorDatasets and DataLoaders for effective training and testing. Note that while training, you will use the RandomSampler and for testing, you will use the SequentialSampler class.

Task - 3: Define the model (20 points)

Create the model, following the architectural specifications provided in the previous section. Be careful when defining the parameters of each layer in the model. You may want to use the token-integer mapping dictionary saved previously to define the size of the embedding layer.

Task - 4: Train and Test the model (25 points)

Look into how the model can be trained and tested. Define a suitable loss and optimizer function. Define suitable values for different hyper-parameters such as learning rate, number of epochs and batch size. To test the model, you may use scikit-learn's classification report to get the precision, recall, f-score and accuracy values. Additionally, also report the throughput of your model (in seconds) at the time of inference.

Task - 5: Prepare a report (10 points)

Prepare a report summarizing your results. Specifically, observe the effect of hyper-parameters such as number of LSTM layers considered, embedding dimension, hidden dimension of the LSTM layers, etc. on model performance and throughput. To get a better understanding of how results are analyzed/summarized, consider the reference paper provided on the webpage.

To Submit

Submit the following:

1. Your source code (either as a Python notebook or regular Python file)
2. Instructions on how to run the code
3. Report

External Links

1. How to use Google Colab
2. PyTorch documentation
3. TenforFlow documentation
4. A very simple TensorFlow tutorial
5. A very simple PyTorch tutorial
6. Textual Entailment with TensorFlow
7. Textual Entailment with PyTorch

References

[1] Daniel Z Korman, Eric Mack, Jacob Jett, and Allen H Renear. Defining textual entailment. *Journal of the Association for Information Science and Technology*, 69(6):763–772, 2018.

[2] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.