Assignment 3

RELEASE DATE: 03/24/2025

DUE DATE: 04/14/2025 11:59pm on Canvas

LaTeX Template: https://www.overleaf.com/read/tpmrnyhfpbcv#be6eb8

Name: First-Name Last-Name UIN: 000000000

This assignment consists of two parts: a writing section and a programming section. For the writing section, please use the provided LaTeX template to prepare your solutions and remember to fill in your name and UIN. For the programming section, please follow the instructions carefully.

When answering problems, please provide a ***detailed and step-by-step explanation*** *to justify your solutions.*

Discussions with others on course materials and assignment solutions are encouraged, and the use of AI tools as assistance is permitted. However, you must ensure that ***the final solutions are written in your own words****. It is your responsibility to avoid excessive similarity to others' work. Additionally, please clearly* ***indicate any parts where AI tools were used*** *as assistance.*

If you have any question, please send an email to csce638-ta-25s@list.tamu.edu

1 Event Detection with Sequential Labeling (Programming) [40pts]

We will solve a classic task in information extraction, called *event detection*. Given one piece of text, we will train a model to detect if certain events are happening in the text. Particularly, we consider the following 8 types of events which are common events for news articles.

Event Type	**Description**
Personnel	Anything related to election, nomination, starting working, stoping working
Conflict	Anything related to conflict, attacks, demonstration
Justice	Anything related to acquittal, appeal, arrest, conviction, execution, extradition, fine, lawsuit, pardon, sentence, trial hearing
Transaction	Anything related to transaction, transferring money, transferring ownership
Movement	Anything related to transport
Contact	Anything related to communication, meeting, phone call
Life	Anything related to birth, death, injury, marriage, divorce
Business	Anything related to bankruptcy, creating organization, merging organization, ending organization

We define that an event happens in the text as long as we can identify *a trigger span* in the text. Here are some examples:

Text	Events
America warns it will seek more **layoffs** if it does file for **Chapter 11**	Personnel (trigger: layoffs) Business (trigger: Chapter 11)
State proceedings could bring the **death penalty** if Nichols is **convicted**	Justice (trigger: death penalty) Justice (trigger: convicted)
Chalabi was the first top Iraqi opposition leader to be **airlifted** by the U.S. military into southern Iraq as the **fighting** wound down, and he and other top members of his group plan to **meet** soon in Baghdad.	Movement (trigger: airlifted) Conflict (trigger: fighting) Contact (trigger: meet)

We are going to fine-tune a BERT model with sequential labeling to solve this task.

`CSCE638-S25-HW3-1.ipynb`: Colab Notebook

`event-train.json`: Data

`valid.json`: Data

`test.json`: Data

Please use your *@tamu.edu* email to access the Colab Notebook. Copy the Colab Notebook to your drive and make the changes. The notebook has marked blocks where you need to code.

========= TODO : START =========
========= TODO : END =========

Please copy and paste your code (between `TODO:START` and `TODO:END`) **as part of the solution.** You can use the Minted package for code highlighting. Here is one example:

For this problem, you might have to *change the Colab runtime type* to enable GPU computation. Please choose the T4 GPU.

1.1 Creating BIO Examples [12pts]

We have to convert the raw data to BIO sequences for sequential labeling. The raw data is in the following JSON format:

```
{'text': ['america', 'warns', 'it', 'will', 'seek', 'more', 'layoffs', 'if', 'it',
          'does', 'file', 'for', 'chapter', '11', '.'],
 'events': [{'trigger': [6, 7], 'type': 'Personnel'},
            {'trigger': [12, 14], 'type': 'Business'}]
}
```

It means that we have two events for this text, a `Personnel` event with the trigger span (layoffs) from index 6 (inclusive) to index 7 (exclusive) and a `Business` event with the trigger span (chapter 11) from index 12 (inclusive) to index 14 (exclusive). We have to convert it to `input_ids`, `attention_mak`, and `output_ids` with `transformers.BertTokenizer` (link). Specifically, please load `google-bert/bert-base-uncased` for the tokenizer. Here is the result of the above example:

```
input_ids = [101, 2148, 4420, 1005, 1055, 6996, 2283, 2003, 5307, 2019, 4722,
             5290, 2923, 2655, 2005, 4487, 24137, 25780, 2993, 2000, 3443, 1037,
```

```
            2047, 2177, 2044, 1037, 28284, 4154, 2012, 2197, 2733, 1005, 1055,
            2011, 1011, 3864, 1010, 2283, 4584, 2056, 9857, 1012, 102, 0, 0, 0,
            0, 0, ...]
attention_mask = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, ...]
output_ids = [0, 0, 0, 0, 0, 0, 0, 1, 9, 0, 0, 0, 0, 0, 8, 16, 0, 0, 17, 17, 17,
              17, 17, 17, 17, 17, ...]
```

To better understand the result, the corresponding subtokens of `input_ids` and the labels of `output_ids` are shown below

```
subtokens = ['[CLS]', 'america', 'warns', 'it', 'will', 'seek', 'more', 'lay',
             '##offs', 'if', 'it', 'does', 'file', 'for', 'chapter', '11', '.',
             '[SEP]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', ...]
BIO_labels = ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-Personnel', 'I-Personnel',
              'O', 'O', 'O', 'O', 'O', 'B-Business', 'I-Business', 'O', 'O',
              'Ignore', 'Ignore', 'Ignore', 'Ignore', ...]
mapping = [('[CLS]', 'O'), ('america', 'O'), ('warns', 'O'), ('it', 'O'),
           ('will', 'O'), ('seek', 'O'), ('more', 'O'), ('lay', 'B-Personnel'),
           ('##offs', 'I-Personnel'), ('if', 'O'), ('it', 'O'), ('does', 'O'),
           ('file', 'O'), ('for', 'O'), ('chapter', 'B-Business'),
           ('11', 'I-Business'), ('.', 'O'), ('[SEP]', 'O'), ('[PAD]', 'Ignore'),
           ('[PAD]', 'Ignore'), ...]
```

Specifically, the `input_ids` should be the concatenation of all the subtokens with a `[CLS]` in the beginning of the text, a `[SEP]` in the end of text, and multiple `[PAD]` to fit the max length. The `attention_mak` should have the same length as `input_ids` and indicates which subtokens are not `[PAD]`.

The `output_ids` should have the same length as `input_ids` and define the corresponding BIO tags based on a `label_mapping` (defined in the code). For instance, the $[8, 16]$ in `output_ids` indicates the subtokens ['chapter', '11'] is the trigger span of a `Business` event, where 8 represents `B-Business` and 16 represents `I-Business` (based on `label_mapping`), corresponding to 'chapter' and '11', respectively. Similarly, the $[1, 9]$ in `output_ids` indicates the subtokens ['lay', '##offs'] is the trigger span of a `Personnel` event, where 1 represents `B-Personnel` and 9 represents `I-Personnel` (based on `label_mapping`), corresponding to 'lay' and '##offs', respectively. Please notice that, although in the raw data, the trigger span of the `Personnel` event is just one word, we have to process it as two subtokens due to the tokenization. You might notice that there are some `'Ignore'` token index (17) at the end of `output_ids`. They are similar to `[PAD]`, which won't be consider when calulating the loss.

Copy and paste your code as well as the output of `test_data2example`.

Solution:

Please enter your solution here.

1.2 Decoding BIO Examples [12pts]

For the training purpose, we have to implement a function to decode BIO sequences back to predictions, based on the original text and `output_ids`. For instance, given the following

```
text =  ['america', 'warns', 'it', 'will', 'seek', 'more', 'layoffs', 'if', 'it',
         'does', 'file', 'for', 'chapter', '11', '.']
output_ids = [0, 0, 0, 0, 0, 0, 0, 1, 9, 0, 0, 0, 0, 0, 8, 16, 0, 0, 17, 17, 17,
              17, 17, 17, 17, 17, ...]
```

We have to generate the following

```
predictions =  [{'trigger': [6, 7], 'type': 'Personnel'},
                {'trigger': [12, 14], 'type': 'Business'}]
```

Copy and paste your code as well as the output of `test_output2prediction`.

Solution:

Please enter your solution here.

1.3 Training [8pts]

We will fine-tune `transformers.BertForTokenClassification` (link) for sequential labeling. Please refer to the previous assignments and finish the code for training. Specifically, we will choose the best checkpoint based on the `compute_score` function (code provided). It calculates the `F1-score` (link) based on the ground truths and the predictions. Please load `google-bert/bert-base-uncased` for BERT.

Copy and paste your code (between `TODO:START` and `TODO:END`) as well as the output.

Solution:

Please enter your solution here.

1.4 Testing [8pts]

Please report the testing F1-score. This time, during grading, we will replace the test file with another hidden file to test the performance of your model. Please make sure that there won't be any runtime errors.

Solution:

Please enter your solution here.

2 Sentence Similarity with Triplet Loss (Programming) [32pts]

As mentioned in the lecture, pre-trained BERT is not good for measuring sentence similarity. In this problem, we are going to tine-tune BERT with triplet loss and make BERT more suitable for sentence similarity.

`CSCE638-S25-HW3-2.ipynb`: Colab Notebook

`sim-train.json`: Data

`sim-valid.json`: Data

`sim-test.json`: Data

Please use your *@tamu.edu* email to access the Colab Notebook. Copy the Colab Notebook to your drive and make the changes. The notebook has marked blocks where you need to code.

$### \ ========= \ TODO : START \ ========= \ ###$
$### \ ========= \ TODO : \ \ END \ \ ========= \ ###$

Please copy and paste your code (between `TODO:START` and `TODO:END`) **as part of the solution.** You can use the Minted package for code highlighting. Here is one example:

```
def hello_world():
    print("Hello World!")
```

For this problem, you might have to *change the Colab runtime type* to enable GPU computation. Please choose the T4 GPU.

2.1 Preparing Data [4pts]

Each example in the raw data consists of three sentences: (1) a reference sentence, (2) a positive sentence, and (3) a negative sentence. The reference sentence has semantics more similar to the positive sentence than the negative sentence. Therefore, we expect the sentence embedding of the reference sentence should be more similar to the sentence embedding of the positive sentence than the sentence embedding of the negative sentence.

Use `transformers.BertTokenizer` (link) to prepare the *input ids* and the *attention masks* for reference, positive, and negative sentences. Specifically, please load `google-bert/bert-base-uncased` for the tokenizer.

Copy and paste your code as well as the output of `test_bert_tokenize`.

Solution:

Please enter your solution here.

2.2 Testing Pre-Trained BERT [6pts]

We first test the performance of pre-trained BERT. Given an example `(ref_sent, pos_sent, neg_sent)`, we use the output embedding of `[CLS]` as the sentence embedding for all sentences. The similarity

will be computed by `cosine-similarity`. We test the percentage of examples where pre-trained BERT can have higher `Sim(ref_sent, pos_sent)` than `Sim(ref_sent, neg_sent)`.

Copy and paste your code (between `TODO:START` and `TODO:END`) and report the score.

Solution:

Please enter your solution here.

2.3 Preparing BERT for Similarity [6pts]

We create a `BERTSimModel` in the following way. The new sentence embedding will be obtained by

$$Tanh(W^\top E),$$

where E is the output embedding of `[CLS]` and W is a learnable linear projection. Please load `google-bert/bert-base-uncased` for BERT.

Copy and paste your code (between `TODO:START` and `TODO:END`).

Solution:

Please enter your solution here.

2.4 Training [8pts]

Please finish the code for training. Specifically, we consider the following triplet loss, which can be viewed as a simplified contrastive loss:

$$\mathcal{L}_{triplet} = -\log \frac{\exp(sim(e_{ref}, e_{pos}))}{\exp(sim(e_{ref}, e_{pos})) + \exp(sim(e_{ref}, e_{neg}))}$$

where e_{ref}, e_{pos}, and e_{neg} are the sentence embeddings of `ref_sent`, `pos_sent`, and `neg_sent`, and $sim(\cdot)$ is the cosine similarity function.

Copy and paste your code (between `TODO:START` and `TODO:END`) as well as the output.

Solution:

Please enter your solution here.

2.5 Testing [8pts]

Please report the testing performance. This time, during grading, we will replace the test file with another hidden file to test the performance of your model. Please make sure that there won't be any runtime errors.

Solution:

Please enter your solution here.

3 Adversarial Attacks for CLIP (Programming) [28pts]

We will implement two methods for attacking CLIP. Given an image, we consider two texts: `a photo of a cat` and `a photo of a dog`. By computing the CLIP scores between the image and the texts, we can classify the image as a cat or dog based on the scores. The provided code has already implemented the classification code. More information can be found here.

`CSCE638-S25-HW3-3.ipynb`: Colab Notebook

`cat.jpg`: Image

Please use your *@tamu.edu* email to access the Colab Notebook. Copy the Colab Notebook to your drive and make the changes. The notebook has marked blocks where you need to code.

$\#\#\#\ =========\ TODO: START\ =========\ \#\#\#$
$\#\#\#\ =========\ TODO:\ \ END\ \ =========\ \#\#\#$

Please copy and paste your code (between `TODO:START` and `TODO:END`) **as part of the solution.** You can use the Minted package for code highlighting. Here is one example:

```
def hello_world():
    print("Hello World!")
```

For this problem, you might have to *change the Colab runtime type* to enable GPU computation. Please choose the T4 GPU.

3.1 Visualizing Images [6pts]

Please use `matplotlib` to show the original image from `inputs["pixel_values"]`. The values in `inputs["pixel_values"]` are *after normalization* with $mean = [0.481, 0.457, 0.408]$ and $std = [0.268, 0.261, 0.275]$ for different channels. Therefore, you have to *denormaliza* the image before showing. More details about image nomalization can be found here.

Copy and paste your code as well as the visualization.

Solution:

Please enter your solution here.

3.2 Fast Gradient Sign Method (FGSM) Attack [8pts]

Fast Gradient Sign Method (FGSM) is a simple attacking method based on the gradient sign. Let x denote the normalized pixel values (`inputs["pixel_values"]`) and consider the cross-entropy loss $\mathcal{L}_{CE}$ based on the CLIP logits (`outputs.logits_per_image`). We can calculate the gradient of loss $\mathcal{L}_{CE}$ w.r.t. x, denoted as $\nabla_x \mathcal{L}_{CE}$. Then, we modify the normalized pixel values x by the following

$$x = x + \alpha \cdot \text{Sign}(\nabla_x \mathcal{L}_{CE})$$

where $\text{Sign}(\cdot)$ is the sign function and α is a hyper-parameter.

Please implement the FGSM. You can use `torch.Tensor.grad` (link) to get the gradient after `loss.backward()`.

Copy and paste your code, and report classification probabilities as well as the image visualization when $\alpha = [0.1, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 5.0]$. You can additionally report the results of other choice of α if you think they better demonstrate the effect of FGSM.

For reference, when $\alpha = 1$, the probabilities of cat and dog are $[0.5961, 0.4039]$, with the image shown below.

Solution:

Please enter your solution here.

3.3 Basic Iterative Method (BIM) Attack [14pts]

Basic Iterative Method (BIM) is an advanced version of FGSM. It is basically repeating FGSM several iterations, which is

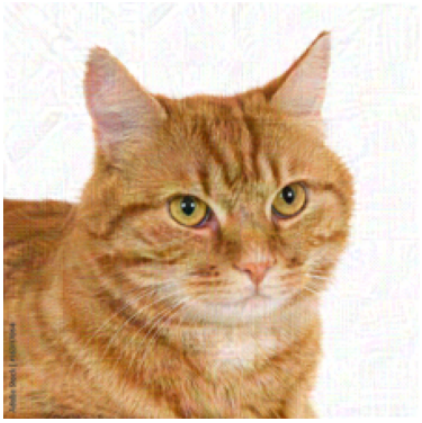
$$x^{(i+1)} = x^{(i)} + \alpha \cdot \text{Sign}(\nabla_{x^{(i)}} \mathcal{L}_{CE})$$

where $x^{(0)}$ is the original normalized pixel values (`inputs["pixel_values"]`). If we set the number of iterations N to 5, we will get $x^{(5)}$ as the adversarial example.

Please implement the BIM. You can use `torch.Tensor.grad` (link) to get the gradient after `loss.backward()`.

Copy and paste your code, and report classification probabilities as well as the image visualization for *all combinations* of $\alpha = [0.05, 0.1, 0.2]$ and $N = [1, 2, 3, 4, 5]$ (15 combinations in total). You can additionally report the results of other choice of α and N if you think they better demonstrate the effect of BIM.

For reference, when $\alpha = 0.05$ and $N = 4$, the probabilities of cat and dog are $[0.4192, 0.5808]$, with the image shown below.



Solution:

Please enter your solution here.

Submission Instructions

You have to upload a `.zip` file to Canvas, which contains the following:

- `submission.pdf`: The `.pdf` file generated by the LaTeX template.
- `submission1.py`: Please export the Colab Notebook for problem 1 to a `.py` file by clicking "File" → "Download" → "Download .py"
- `submission1.ipynb`: Please export the Colab Notebook for problem 1 to a `.ipynb` file by clicking "File" → "Download" → "Download .ipynb"
- `submission2.py`: Please export the Colab Notebook for problem 2 to a `.py` file by clicking "File" → "Download" → "Download .py"
- `submission2.ipynb`: Please export the Colab Notebook for problem 2 to a `.ipynb` file by clicking "File" → "Download" → "Download .ipynb"
- `submission3.py`: Please export the Colab Notebook for problem 3 to a `.py` file by clicking "File" → "Download" → "Download .py"
- `submission3.ipynb`: Please export the Colab Notebook for problem 3 to a `.ipynb` file by clicking "File" → "Download" → "Download .ipynb"