Spring 2025

EDS 6397 – Natural Language Processing

Assignment #4 – Named Entity Recognition using a Transformer-based model

Use of artificial intelligence assistant such as ChatGPT in developing the code for this assignment is allowed. You might still choose not to use AI assistant. However, ***the report needs to be completely written by you****. Use of grammar correction tools is disallowed.*

Assignment Overview

The objective of this assignment is to familiarize you with DistilBERT transformer model. You will use this bidirectional encoder to perform Named Entity Recognition (NER) classification on tweet data.

Note: You need to work with distilbert_base_cased model from huggingface for this assignment.

Assignment Description

You will be given 11,090 short documents (tweets) with tags specified with character spans. You are asked to train a classifier for named entity recognition using 7 categories of tags. Essentially, this is a sequence labeling task. But you will soon see that you have to work with tokens not words, and you have to consider 15 classes, not just 7.

At the end, to get realistic performance metrics, you will reconcile the 15 classes back to 7, and tokens to words. Then you will report performance statistics.

A few reminders:

- Make sure you follow the instructions below very carefully.
- Make sure you cite any online resource or paper that you used that helped you develop the code and/or report for this assignment.
- You need to use DistilBERT tokenizer. Use of other tokenizers such as Spacy's tokenizer is not allowed.
- DistilBERT comes with its own embedding. Use of Other word vectors such as Word2Vec is not allowed.

Input Data

You are given a JSON file named ***assignment5_data.json*** that contains 11,090 tweets that are tagged for NER task similar to assignment 1. The tags represent seven categories: **PER** (person), **NORP** (nationality or religious/political group), **ORG** (organization), **GPE** (geopolitical entity), **LOC** (location), **DATE**, and **MONEY.**

Step 1: Pre-processing

No data cleanup is necessary, as it has already been performed. However, you should plot the total number of instances for each tag category.

Additionally, note that some tweets contain multiple instances of the same tag, while others may lack instances of certain tags. To gain better insight into the data, analyze and discuss the distribution of tag occurrences across tweets. You may achieve this by looking at what percentage of tweets include at least one tag from each of the 7 categories. What percentage may have 2 or more.

Step 2: Converting to IOB tags:

As covered in Chapter 11 of the book, NER typically uses IOB tagging. You need to **write *a function*** that converts the input tags (from the data) into IOB tags. Each document (tweet) should be fed into your function, and the output tags should be recorded. Here are the steps:

1- The input to your function is a document and human-level NER tags provided as character spans
2- Use DistilBERT's tokenizer to tokenize the input document
3- Using the tokens, the original input document, and the tag character spans, create IOB tags as follows:
    a. Each token outside the identified spans gets an **O** tag.
    b. The first token within a span gets a **B- tag**, corresponding to its category.
    c. All subsequent tokens of the span get an **I- tag**, corresponding to their categories.
4- Make sure you include **O** tags for DistilBERT's special character tokens [CLS] and [SEP].

Step 3: Finetuning DistilBERT - Training a sequence labeling model:

- Split the **data** into training and test sets with an 80-20 ratio with scikit-learn's train-test-split function with random state of 4. Set aside the test data for final evaluation.
- Further split the **training set** into training and validation subsets using the same function and split ratio with a random state of 4.
- Add a sequence labeling head to the pre-trained DistilBERT transformer model.
- Ensure that the transformer weights are not fine-tuned (are frozen).
- Fine-tune only the parameters in the sequence labeling head for at least 30 iterations.
- Use categorical cross-entropy as the loss function.
- Calculate and report the precision, recall, and F1 scores per class for each of the 15 classes using the validation dataset.
- In your report, include the reasoning behind your choice of number of epochs, learning rate, dropout, and other hyperparameters.

Step 4: Converting IOB to plain tags:

0- *Run inference*
With your finetuned model, run inference on the 20% test data that you had set aside.

1- *Prepare Tags for Reconciliation:*
Before reconciling subword tags, convert all I- tags to their corresponding B- tags for consistency. This step simplifies the tag structure and makes it easier to combine subwords accurately. Specifically:

- For any tag in the even-numbered series (2, 4, 6, 8, 10, 12, 14), convert it to its corresponding odd-numbered tag (1, 3, 5, 7, 9, 11, 13) which aligns with B-PER, B-NORP, B-ORG, B-GPE, B-LOC, B-DATE, and B-MONEY.
- Leave 0 tags (non-entity tokens) unchanged.

2- *Reconcile Subword Tokens Using Majority Voting:*
Write a function that takes two lists—one containing tokens (as tokenized by the DistilBERT tokenizer) and another containing tags corresponding to each token. This function should reconcile the tags for subwords by combining them into full words and determining the final tag based on majority voting.

Overall, the function should operate as follows:

- If tokens and tags are ['Ta', '##ku', '##ya', 'Ta', '##ka', '##gi'] and [1, 2, 2, 2, 2, 0], first convert all even tags to odd, resulting in [1, 1, 1, 1, 1, 0].
- Group subword tokens based on prefixes (like ##) and form words from combined tokens: e.g., ['Takuya', 'Takagi'].

Now develop **two more functions** that assigns a single tag to each word (using tags of subwords):

3- *Evaluate Model Performance Using Majority Voting Results:*
- Assign tags based on majority voting within each word's subword span.
- Evaluate model performance per class and at a word level.

4- *Reconcile Subword Tokens Using First Tag of Span:*
- Assign each word the tag of its first token rather than using majority voting.
- Evaluate model performance per class and at a word level.

The report

- Your report should be up to four pages and must include the following sections (failure to do so will result in a deduction of points):
- Performance metrics for the last epoch, per class.
- Performance metric at human-level tags, also known as word level (step 4, subtasks 3, and 5)
- Explain why provided token tags have to be converted to IOB tags.
- Justify your choice of number of epochs, learning rate, dropout (if used), and any other hyper parameter used.
- Description of the fine-tuning approach and any challenges encountered.
- Evaluation results with discussion on model performance.

- Include any additional insights you gained from this assignment.

Questions

1- What is the size of the weight matrix for this finetuning project that you trained?
2- What hyper parameters (if any) did you set, and how did you determine their values?
3- Explain what is achieved in step 4, part 1. Why do we need to do this?
4- Describe the impact of tokenization on tagging accuracy and the handling of subwords in Step 4. Which reconciliation method worked better? Majority voting or the first tag of a span?
5- Discuss any limitations of the DistilBERT-base_cased NER approach you noticed.
6- Overall, how do you see the performance? What do you think of the data?
7- Which categories of named entities you would exclude if this was your research, and why?

What to submit:

1- Your Python Jupyter Notebook should include plenty of comments and explanatory cells to ensure that your code is easy to read and understand. The Notebook should also contain the results of each cell executed by you. It should be named as follows: **[Last_name]_HW4.ipynb**. You can instead submit a Google Colab file. Use a similar naming convention.
Make sure the 4 functions that you were asked (in highlighted text) are clearly named function 1, 2, 3, and 4.
2- A report (2-4 pages) **in PDF format** that briefly discusses what you did for this assignment as well as the results. Make sure you include everything listed under subsection *the report* above as well as answer to all questions above.

Late submissions will be penalized at a rate of 1 point per hour.

We round up the time to the nearest hour.

Make sure you click on submit so your assignment is "Turned In". Uploading the files doesn't mean you turned the assignment in.

Due Date: April 10, 2025 by 11:59 PM (submit through Teams)