CS 5/7322
Introduction to Natural Language Processing
Fall 2025

Programming Homework 2: Examining contextualized Word Embeddings for BERT

Due date: 11/5 (Wed) 11:59pm

In this program, you are to explore "contextualized" word embeddings and examine the "quality" of the vectors that are generated.

This is an individual effort, not a group project.

BERT (via pytorch and transformer modules)

BERT is a contextualized word embedding. That is, if one passes a sentence to BERT, it can output a vector corresponding to each word. However, even for the same word, the vector corresponding to it will be different for the same word in two different sentences.

Here is a short sample code of using BERT via the transformer and torch module to extract the embedding of words in a sentence. Remember that BERT always put a [CLS] token at the beginning of the sentence.:

```
from transformers import BertModel, BertTokenizer
import torch

# Load pre-trained BERT model and tokenizer
model_name = 'bert-base-uncased'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertModel.from_pretrained(model_name)

# Sample sentence
sentence = "This dog is big"
sentence2 = "This dog is lovely"

# Tokenize the input sentence
inputs = tokenizer([sentence, sentence2], padding=True,
return_tensors="pt")

print(inputs)

# Forward pass through the BERT model
with torch.no_grad():
    outputs = model(**inputs)
```

```
# Get the hidden states (embeddings) of the last layer
last_hidden_states = outputs.last_hidden_state

# Convert tensor to numpy array
last_hidden_states = last_hidden_states.numpy()

# Print the shape of the last hidden states
print(last_hidden_states.shape)  # Shape will be (batch_size,
sequence_length, hidden_size)

for x0 in last_hidden_states:
    print("-----")
    for x in x0:
       print(x)
```

Part 1: Comparing BERT vectors for same word + same meaning across sentences

The goal of a word embedding is to map words into vectors such that words that have similar meaning will be mapped to vectors that are "close" to one another. Since BERT can map the same word into different vectors, it will be interesting to see whether the vectors for the same word (that has the same meaning) will be mapped to similar vectors.

You should take the following steps:

1. Pick 8 words. They should be quite different in meaning. We will denote each word by w1, w2, …, w8 respectively.
2. For each word, make up 10 sentences that contain the word. The chosen word should have the same meaning in the 10 sentences. Also try to avoid generating sentences that are similar to one another. Also please keep the same wordform for all sentences.
    a. For example, for the word "park". You can generate the following sentences
        i. The park next to my house offers a nice walk.
        ii. The residents is trying to build a park for the kids to play.

       Given the about sentences, you should NOT generate the following sentences:
        iii. The park next to my street offers a nice walk. (too similar to the first sentence)
        iv. You should put the car in park and got out. (park here means different things)
3. For each sentence, put it to BERT and obtain the vector for the word. For word wi, we will call the vector wi1, wi2, wi3, …, wi10 respectively. In this case, take the first 50 dimensions for each vector.
4. For each word, calculate the cosine similarity between every pair of vectors. (so each word has 45 pairs of vectors [10*9/2])
5. For each word, you should report the following:
    a. The ten sentences
    b. The mean and standard deviation of the cosine similarity
    c. A histogram plotting the distribution of cosine similarity (you should have ten columns, with ranges of 0-0.1, 0.1-0.2, 0.2-0.3,…, 0.9-1)

6. Also write a paragraph in your report to suggest whether BERT consistently generates vectors with high similarity for the same word.
7. After this you should have one histogram that combines the results of each word (i.e. a total of 45 * 8 = 360 pairs). And calculate the overall mean and standard deviation of the cosine similarities (include that in your report).

Part 2: Comparing BERT vectors for different words and different meaning across sentences

To get a fuller picture, we should also compare vectors of different words to see if the similarities between those vectors are different.

1. Use the vectors generated in part 1, randomly pick 360 pairs of vectors such that each vector of the pair is from different words. Calculate the cosine similarity of each pair.
2. Calculate the mean and standard deviation of all the similarities in step 1. Put it in the report
3. Also draw a histogram as in the one in step 7 of part 1. Put it in the report.
4. Write a paragraph to compare the two histograms (the one in step 7 of part 1, and step 3 of part 2).

Part 3: Compare BERT vector for different words and same meaning across sentences

If the embedding vectors are BERT is good in maintaining meanings, different words that means the same thing should be mapped to similar vectors. We want to explore this in this part of the homework:

1. Pick eight pairs of words that have a similar meaning. You should do this by looking at synsets in Wordnet. Pick 8 synsets that contain multiple words, and for each of those synsets, pick 2 words. (Notice WordNet allows phrases to be in a synset. For this project we should keep to single words).
    a. For example, there is a synset that contains the words (car, auto, automobile, machine, motorcar). You can pick (car, automobile) as a pair.

    While the GUI is no more, you can use the NLTK api for WordNet to look for synset. In fact, Wordnet "database" is just a few text files you can read directly to look for information you need.
2. For each pair, make up 5 sentences for each word. The word(s) should be used as the meaning described in the synset. The sentences should be different (and not just substitute one word with the other). For example, you can generate:
    i. The red car is very fast.
    ii. This is a turbocharged automobile

    Please do not substitute the word in the same sentence structure. i.e. do not use the sentence "This is a turbocharged car".
3. For each sentence, run it through BERT to generate the vectors for each word (in the example above, 5 vectors for car, and 5 vectors for automobile).
4. Calculate the similarity between every pair of vectors where one vector is from each word respectively. (i.e. for each vector of car, calculate the similarity with all 5 vectors of automobile).
5. Repeat part 5-7 of part 1 for this set of data

Implementation

You should implement the following function and put it in “hw2.py”

- genBERTVector(model, tokenizer, word, list)
    - You are given a BERT model, the corresponding tokenizer, a word, and a list of strings, where each string is a sentence that should contain the word.
    - The output should be a list of vectors, where each vector corresponds to the vector for the word in each sentence in the list. If the word appears more than one in a sentence, take the vector corresponds to the first appearance in the sentence. If the word does not appear in the sentence, you should return an empty list.
    - For example: consider the following:

```
model_name = 'bert-base-uncased'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertModel.from_pretrained(model_name)
v = genBERTVector(model, tokenizer, “dog”, [“This dog is barking”, “This dog is barking loudly”, “How are you today?”, “My dog is a big dog”])

Then v = [ v1 v2 v3 v4]
Where the first 8 dimension of v1 and v2 are:

V1[:8] = [ 0.37257612  0.7042742   0.16273095  0.02697564  0.07547668 -0.15100406 0.52527225  0.05713326]
V2[:8] = [ 0.6084792   0.7719362   0.28940228 -0.10523753  0.14368033 -0.3067458 0.37029967  0.1639219]
V3 = []
V4[:8] = [-0.07936206  0.6111502   0.10624825 -0.6505449   0.83996457  0.32269466  -0.2398438   0.11347568]
```

You should use this function in your main program for the task above.

What to hand in

You should hand in the python source code (NO juypter notebook please) that contains the function specified, together with a pdf file that reports all the results. You should zip the file and upload it.

Extra Credit (15 points)

Repeat part 1 and 2 with a domain specific version of BERT, you can choose between LegalBERT, BioBERT, SciBERT or FinBERT.