BME646 and ECE60146: Homework 11

Spring 2025
Due Date: 11:59pm, May 2, 2025
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via Gradescope. Post questions to Piazza. Additional instructions can be found at the end. No late submissions will be accepted.

1 A Special Note

Our original goal was to give you a homework based on babyGPT that would get you to train a small enough model with a small enough corpus of text data. Unfortunately, the smallest babyGPT based models that do anything meaningful at all require more than 300 million learnable parameters.

So, for this year, we are going back to the last homework that was given last year and creating a variant thereof that is described in the rest of this document. That is, as last year, the focus is again on tokenization. The main difference is that you are now required to compare the BERT tokenizer with the tokenizer that comes bundled with babyGPT.

When you do the comparison, keep in mind the fact that the `104_babygpt_tokenizer_49270.json` tokenizer in babyGPT was trained on a corpus of athlete news data. In that sense, it is a 'specialty domain' tokenizer as opposed to the general purpose BERT tokenizer.

2 Introduction

LLM (Large Language Modeling) would not work without tokenization. Tokenization guarantees a fixed-sized vocabulary of symbolic tokens irrespective of the actual size of the word vocabulary in your text corpus. LLM requires a fixed size for the vocabulary because, for autoregressive modeling, the last layer of the network must estimate the probabilities over the

entire vocab in order to predict the most probable next token. It would be impossible to do this estimation unless the size of the vocab is known in advance.

3 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. Install `transformers` library into your conda environment, as this will be used to extract subword tokens.
2. Please download the babyGPT module from the following link:

   https://engineering.purdue.edu/kak/distBabyGPT/babyGPT-1.0.5.html
3. In order to run the required script, you are going to need a text corpus. For your run you will be using the smaller of the two datasets provided through the link "Download the text datasets for babyGPT".

4 Programming Tasks

4.1 Word-level Tokenization

Your first task in this HW would be to tokenize the data. The steps are:

1. Depending on the specific task at hand, text can be tokenized at various levels such as character, subword, word, or sentence level. For this assignment, we will focus on tokenization at the word and subword levels.
2. To tokenize text at the word level, each word is separated at whitespace boundaries. This can be achieved using the built-in `split()` function. The following code snippet illustrates how this process can be implemented:

```
import csv

# this is an example of how to read a csv file line by line
# This snipped only shows the processing on the first 4 entries
sentences = []
sentiments = []
count = 0
with open('data.csv', 'r') as f:
    reader = csv.reader(f)
    # ignore the first line
    next(reader)
    for row in reader:
        count += 1
        sentences.append(row[0])
        sentiments.append(row[1])
        if count == 4:
            break

print(sentences)
# ["The GeoSolutions technology will leverage Benefon 's GPS solutions by providing Location Based Search Technology , a Communities Platform , location relevant multimedia content and a new and powerful commercial model .",
# '\$ESI on lows, down \$1.50 to \$2.50 BK a real possibility',
# "For the last quarter of 2010 , Componenta 's net sales doubled to EUR131m from EUR76m for the same period a year earlier , while it moved to a zero pre-tax profit from a pre-tax loss of EUR7m .",
# 'According to the Finnish-Russian Chamber of Commerce , all the major construction companies of Finland are operating in Russia .']

print(sentiments)
# ['positive', 'negative', 'positive', 'neutral']

# tokenize the sentences word by word
```

```
word_tokenized_sentences = [sentence.split() for sentence in sentences]
print(word_tokenized_sentences[:2])
# [['The', 'GeoSolutions', 'technology', 'will', 'leverage', 'Benefon', "'s", 'GPS', 'solutions', 'by', 'providing', 'Location', 'Based', 'Search', 'Technology', ',', 'a', 'Communities', 'Platform', ',', 'location', 'relevant', 'multimedia', 'content', 'and', 'a', 'new', 'and', 'powerful', 'commercial', 'model', '.'], ['$ESI', 'on', 'lows,', 'down', '$1.50', 'to', '$2.50', 'BK', 'a', 'real', 'possibility']]
# pad the sentences to the same length
# here I chose the max of all the sentences. You may set it to a hard number such as 64, 128 etc.
max_len = max([len(sentence) for sentence in word_tokenized_sentences])
padded_sentences = [sentence + ['[PAD]'] * (max_len - len(sentence)) for sentence in word_tokenized_sentences]
print(padded_sentences[:2])
#[['The', 'GeoSolutions', 'technology', 'will', 'leverage', 'Benefon', "'s", 'GPS', 'solutions', 'by', 'providing', 'Location', 'Based', 'Search', 'Technology', ',', 'a', 'Communities', 'Platform', ',', 'location', 'relevant', 'multimedia', 'content', 'and', 'a', 'new', 'and', 'powerful', 'commercial', 'model', '.', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]'], ['$ESI', 'on', 'lows,', 'down', '$1.50', 'to', '$2.50', 'BK', 'a', 'real', 'possibility', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[
```

```
                                    PAD]', '[PAD]', '[PAD]',
                                    '[PAD]', '[PAD]', '[PAD]',
                                     '[PAD]', '[PAD]', '[PAD
                                    ]', '[PAD]', '[PAD]', '[
                                    PAD]', '[PAD]', '[PAD]',
                                    '[PAD]', '[PAD]', '[PAD]',
                                     '[PAD]', '[PAD]', '[PAD
                                    ]', '[PAD]', '[PAD]']]
```

4.2 Sub-Word-level Tokenization

3. Subword level tokenization, also known as wordpiece tokenization employed in models like BERT, offers the advantage of breaking down less frequent words into subwords that occur more frequently. Below is code snippet demonstrating how one can perform subword tokenization as used in BERT:

```
from transformers import DistilBertTokenizer
model_ckpt = "distilbert-base-uncased"
distilbert_tokenizer = DistilBertTokenizer.from_pretrained
                                    (model_ckpt)

# bert encode returns the tokens as ids.
# i have set the max length to what we have padded the
                                    sentences to in word
                                    tokens
# you are free to choose any size but be consistent so
                                    that you may use the same
                                    model for training.
bert_tokenized_sentences_ids = [distilbert_tokenizer.
                                    encode(sentence, padding='
                                    max_length',
                                    truncation=True,
                                    max_length=max_len)
                                    for sentence in sentences]

print(bert_tokenized_sentences_ids[:2])
# [[101, 1996, 20248, 19454, 13700, 2015, 2974, 2097,
                                    21155, 3841, 12879, 2239,
                                    1005, 1055, 14658, 7300,
                                    2011, 4346, 3295, 2241,
                                    3945, 2974, 1010, 1037,
                                    4279, 4132, 1010, 3295,
                                    7882, 14959, 4180, 1998,
                                    1037, 2047, 1998, 3928,
                                    3293, 2944, 102], [101,
```

```
                                    1002, 9686, 2072, 2006,
                                    2659, 2015, 1010, 2091,
                                    1002, 1015, 1012, 2753,
                                    2000, 1002, 1016, 1012,
                                    2753, 23923, 1037, 2613,
                                    6061, 102, 0, 0, 0, 0, 0,
                                    0, 0, 0, 0, 0, 0, 0, 0, 0,
                                     0, 0]]
bert_tokenized_sentences_tokens = [distilbert_tokenizer.
                                    convert_ids_to_tokens(
                                    sentence) for sentence in
                                    bert_tokenized_sentences_ids
                                    ]
print(bert_tokenized_sentences_tokens[:2])
# [['[CLS]', 'the', 'geo', '##sol', '##ution', '##s', '
                                    technology', 'will', '
                                    leverage', 'ben', '##ef',
                                    '##on', "'", 's', 'gps', '
                                    solutions', 'by', '
                                    providing', 'location', '
                                    based', 'search', '
                                    technology', ',', 'a', '
                                    communities', 'platform',
                                    ',', 'location', 'relevant
                                    ', 'multimedia', 'content
                                    ', 'and', 'a', 'new', 'and
                                    ', 'powerful', 'commercial
                                    ', 'model', '[SEP]'], ['[
                                    CLS]', '$', 'es', '##i', '
                                    on', 'low', '##s', ',', '
                                    down', '$', '1', '.', '50
                                    ', 'to', '$', '2', '.', '
                                    50', 'bk', 'a', 'real', '
                                    possibility', '[SEP]', '[
                                    PAD]', '[PAD]', '[PAD]',
                                    '[PAD]', '[PAD]', '[PAD]',
                                     '[PAD]', '[PAD]', '[PAD
                                    ]', '[PAD]', '[PAD]', '[
                                    PAD]', '[PAD]', '[PAD]',
                                    '[PAD]', '[PAD]']]
```

4. You will observe that tokens have an extra `[CLS]` and `[SEP]` token when used with the bert tokenizer. Some words are split into subwords. Example: `"solution"` is split into `"##sol"` and `"##ution"`.

4.3 Baby-GPT Tokenizer

To better understand the behavior and structure of the tokenizer used in the BabyGPT model, we begin by training a new tokenizer on a large-scale corpus contained within the directory `saved_articles_dir_12M`. This process involves learning byte pair encoding (BPE) merges and constructing a vocabulary that captures the most frequent subword patterns present in the dataset. To do this, run the script:

`train_tokenizer.py`

with the `saved_articles_dir_12M`.

Once trained, the newly learned tokenizer is serialized into a JSON format for compatibility and reuse. We then perform a comparative analysis between the tokenized outputs produced by this custom tokenizer and those generated by the pre-existing tokenizer included with the BabyGPT module: `104_babygpt_tokenizer_49270.json`.

Here is a code snippet for you to perform this comparison. Show your results on the given sentences and 5 strings of your choice. Make sure your strings are atleast 25 word long. You can pick the sentences from any online resource, just make sure you cite it.

You may try diverse online sources (news, academic articles, and encyclopedic content). The objective is to observe variations in subword segmentation, vocabulary coverage, and overall token counts. The tokenizer behavior may vary due to differences in the underlying training corpora and resulting vocabulary merges.

```
1 from transformers import PreTrainedTokenizerFast
2
3 sent = ["The GeoSolutions technology will leverage Benefon 's
                                GPS solutions by providing
                                Location Based Search
                                Technology , a Communities
                                Platform , location relevant
                                multimedia content and a new
                                and powerful commercial model .
                                ",
4 '\$ESI on lows, down \$1.50 to \$2.50 BK a real possibility',
```

```
"For the last quarter of 2010 , Componenta 's net sales doubled to EUR131m from EUR76m for the same period a year earlier , while it moved to a zero pre-tax profit from a pre-tax loss of EUR7m .",
'According to the Finnish-Russian Chamber of Commerce , all the major construction companies of Finland are operating in Russia .']
tokenizer_json  =  '104_babygpt_tokenizer_49270.json'
tokenizer = PreTrainedTokenizerFast(tokenizer_file=tokenizer_json)

encoded = tokenizer(sent)

for i in range(4):
    print(tokenizer.decode(encoded['input_ids'][i]))

# The Ge o S ol uti on s tech no lo gy will l ever age Ben ef on's GPS solution s by provi ding Lo ca ti on Base d Se ar ch Te ch no lo gy, a Com m unit i es Pla t for m, lo ca ti on relev ant multi medi a content an d a new an d power ful commercial model.
# \ $ E SI on low s, down \ $ 1. 50 to \ $ 2. 50 B K a real possi bi lit y
# For the last quarter of 2010, Com pon en ta's net sale s dou ble d to EU R 13 1m from EU R 76 m for the sa me peri od a year earlier, whi le it mov ed to a zero pre - tax pro fit from a pre - tax loss of EU R 7 m.
# Accor ding to the Fin ni sh - Russian Cham be r of Com merce, all the major construction compani es of Fin land are operat ing in Russia.
```

Finally, compare word-, subword- and babyGPT-tokenizer with all the sentences. These are emperical observations only.

5 Submission Instructions

Include a typed report explaining how you solved the given programming tasks. You may refer to the homework solutions posted at the class website for the previous years for examples of how to structure your report

1. **Turn in a PDF file and mark all pages on gradescope**.
2. Submit your code files(s) as zip file.
3. **Code and Output Placement:** Include the output directly next to the corresponding code block in your submission. Avoid placing the code and output in separate sections as this can make it difficult to follow.
4. **Output Requirement:** Ensure that all your code produces outputs and that these outputs are included in the submitted PDF. Submissions without outputs may not receive full credit, even if the code appears correct.
5. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert code to `.py` and submit that as source code. **Do NOT submit .ipynb notebooks.**
6. You can resubmit a homework homework as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.
7. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**

References