CSE481 NLP Capstone: Blog Post 4

Team name: Bitmaps - **BI**narized **T**ransformers for **M**aking f**A**st **P**rediction**S**
List of members: Tobias Rohde, Anirudh Canumalla
GitHub URL: nlp-capstone

Baseline Approach

Before attempting any binarization, we ensured that we are using BERT correctly by having it predict a masked token in a few sample sentences. For instance, in the sentence “I am taking a [MASK] on Natural Language Processing.”, BERT predicts course, class, phd and degree as its top 4 choices. Note that we are using bert-base. Originally we used bert-large, but decided this is unnecessary. Next we started building our baseline model. Our baseline approach was to naively binarize the weights of BERT using a fixed threshold. That is, for all parameters in BERT, we independently set the parameter to one of two values, depending on if the parameter is above or below a cutoff. We tried to use -1 and 1, which was used in the XNOR-Net paper and we tried 0 and 1. In both cases we used a cutoff of 0, so if a parameter is less than 0 we set it to the first value and if it is greater than 0 we set it to the second value. We did not expect this approach to work very well, but tried it just in case since this is the most basic approach to binarization we could think of.

We tested the naive binarization on the previous sentence to get a feeling for whether the model still works with binary parameters. As expected, the top-4 predictions were completely off and seemingly random. In order to quantify this loss, we proceeded to set up our evaluation framework. We checked the BERT paper and found that BERT was pre-trained using the BooksCorpus and English Wikipedia. This ruled out our initial idea of using WikiText as a dataset for evaluation and left us with the Penn Tree Bank data. We implemented the necessary code to load this dataset. Next we wrote code to randomly mask out 15% of the tokens in the dataset. In the BERT paper, a fraction of those tokens is also set to other random tokens and another fraction is left unchanged. We did not implement this yet but will do so soon. Next we implemented the cross entropy loss, which was slightly tricky since we only need consider the cross entropy loss of the randomly masked tokens. Now we reran original BERT and the two binarized versions of BERT on the training data of the Penn Tree Bank (since this is the largest split and we are not planning on training on this data anyways) and calculated the mean cross entropy. BERT achieved a mean cross-entropy loss of 4.5788. This seems higher than expected. We think that this is because the Penn Tree Bank text was tokenized with the PTBTokenizer, which might cause problems with the BERT tokenizer. For instance, in Penn Tree Bank text, all apostrophes are separated by a space from their corresponding word (for instance “city ‘s”). We performed some basic data cleaning, such as replacing unk tokens with the corresponding token that BERT understands, but there is more sanitary work that needs to be done on the data. Next we evaluated the -1, 1 binarized model and it achieved an average cross entropy of 261.2058, which is extremely high as expected.

To our surprise, the 0, 1 binarized model achieved a mean cross entropy of 124.2436, which is still bad but much lower than that of the -1, 1 model. At this point we have no reasonable explanation for this and it contradicts our original expectations. However, we still think that the -1, 1 binarization will perform better once properly implemented.

(Note that we are not storing the masked dataset yet, so every time we are running the experiment we randomly mask out different tokens. After repeating our experiments several times, we found the cross entropy only varies slightly. We will soon write code to create a masked dataset once and store it so that we have a fixed evaluation dataset)

You can find our baseline code here.