BERT based text classifier

Here is a notebook for training a BERT based text classifier. This model loads a pre-trained BERT encoder and fine-tunes it for text classification (learns a classification layer on top of the encoder).

a) Study the notebook to see how the model is created. Summarize the model creation part (note that most of the code we have already seen previously in CNN/RNN notebooks, so no need to summarize these parts anymore). Especially, you should explain what AutoModel and ForSequenceClassification means in transformers.AutoModelForSequenceClassification.from_pretrained().

b) How the accuracy compares with your previous CNN/RNN experiments?

WEEK 4 EX 7 BERT BASED TEXT CLASSIFIER

A

To summarize the BERT mode notebook, i'll skip over most of the function definitions. One thing to mention is in the `def tokenize`, where parameter `max_length` is set to 512, which is the maximum the BERT models can handle due to the position embeddings.

With the rows:

```
model_name = "bert-base-cased"
model = 
transformers.AutoModelForSequenceClassification.from_pretrained(model_name
, num_labels=2)
```

First the bert model is chosen and inserted in to the variable `model_name` and after that the chosen *pre-trained* model is downloaded AS A GENERIC MODEL which has bulk of the work done already by bigger instances for sequence classification from the Huggingfaces' Amazon S3 service. This model needs to be finetuned for the task in hand.

First the model must be configured just like CNN/RNN cells with hyperparameters. The ones were interested are: `learning rate`, `pre_device_train_batch_size`, `per_device_eval_batch_size` and `max_steps` which are set as transformers `TrainingArguments`. Early stopping and logging is the same as CSS/RNN cells. So the training is pretty straight forward operation in theory, my own experiments with finetuning is on the contrary.

B

The results of transformer based classifier compared to earlier work with CNN/RNN cells is greatly improved from 71.x% accuracy to 91%.