Domain Training BERT to Improve Performance of the Pronoun Disambiguous Problem

April 29th, 2020

Jeremie Fraeys de Veubeke
0892019
Masters Student
School of Computer Science
University of Guelph

Presented to Fei Song
For: CIS*6650 - Special Topics in Computer Science
Natural Language Processing

Table of Content

Introduction

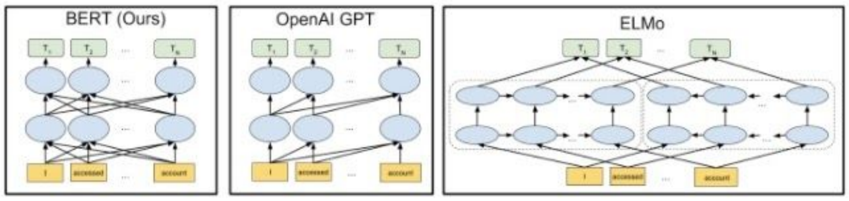
As the world adopts more and more the need for automation and learning, an increased interest for techniques have been adopted by the industry. These help them understand their environment and consumers better than ever. From a marketing analyst using the AI to develop target markets for a new campaign that will not only help focus the proper messages to the right potential consumer. To a technology company using the state-of-the-art technique in their product to increase the automation of divers' reaction to users behaviour. In 2018, Google did just this as it sees that their search engine encounters 15% search queries that the engine has never seen before. Such as, users that may be uncertain of the proper way to formulate the query or the spelling of a word. Over a day this engine is used billions of times around the world resulting in a large quantity of novel queries that will not be optimizely answered. BERT was developed by the Google Lab team to remedy this, as a novel state-of-the-art algorithm. This algorithm is aimed at question and answer problems but offers the possibility to be used for numerous other problems. In this project, we will be looking at using BERT to answer the Pronoun Disambiguous Problem(PDP). It is a commonsense reasoning conference challenge proposed to four participants during the IJCAI 2016 conference. The challenge was proposed by Hector Levesque from the University of Toronto to test machine intelligence. The question is to use AI to solve for a pronoun in a sentence. The algorithm will be given a sentence and a pronoun within that sentence that must be assigned to one of the given choices of nouns in the sentence. The rules of the competitions were that the participant to achieve 90% accuracy would win a cash prize. That year the highest accuracy was 58%. And still, these days as the competition is over individuals and groups are still researching possible solutions to the problem. A Google competition has been launched on a data science competition website named: Kaggle. This competition is also over, it was running in early 2019. No pre-posted accuracy had to be achieved, group up to 5 members could attempt. The Kaggle competition although not specific to BERT saw a large utilisation of the powerful model. Incorporating this model as a classifier was a popular approach, adding it to a Multi-Layer Perceptron, often referred to as a Forward Feeding artificial Neural Network(FNN). The results were given as a score instead of accuracy or f1 score, therefore, is hard compared to other models' performance. The problem is still a great challenge for models and must be researched further.

Applications of this problem are growing as the IoT and human to computer communication are moving to conversation based talks. As virtual assistants are becoming more used in mobile phones and integrated into more and more processedures, the fluidity of these systems to understand more complex language will improve the user experience and performance.

In this project, we will be looking at the “Attention Is (not) All You Need for Commonsense Reasoning” article that tries to use BERT to solve the PDP. This proves to be performing well when compared to techniques used in the IJCAI-2016 conference. However, it is being surpassed by numerous attempts to perform at 90% accuracy. We attempted to improve on the experiments of the article by using other features and models offered by BERT. Such as, different type of pre-trained models and the run_pretraining function that made it possible to domain train any model further.

BERT: Bidirectional Encoder Representations from Transformation

In 2018, Google's research team published a article name Attention is all you need article[1]. This article presented a novel state-of-the-art model based on Attention. This model was first thought to help google with search engine queries. As 15% of these queries have never been seen before Google was looking for an AI that could better read context in a sentence. Hence, the creation of BERT, BERT as the name says is capable of di-direction analysis, as the nodes in the model are capable of looking at all other elements to determine context of a word.



Deeply bidirectional unsupervised language representations with BERT

This multi-head attention, as it is called in the paper, allows the model to understand words such as "for" and "to" better. As such words may be related to words before or after itself, which other models would not have an easy time differentiating. Attention is calculated by the formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The attention input is a pair of Key-value and a query, the output is three vectors of all of these. The output is calculated as the weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O, \ where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

The multi-head attention using the first dot-product equation finds the attention for all words(Z), but will get multi-weighted vectors WQ, Wk, WV. These represent different subspaces at different positions. Finally reduce the dimension as the feed forward NN expects one vector per word. All these attention layers are running in parallel.

[1] https://arxiv.org/pdf/1706.03762.pdf

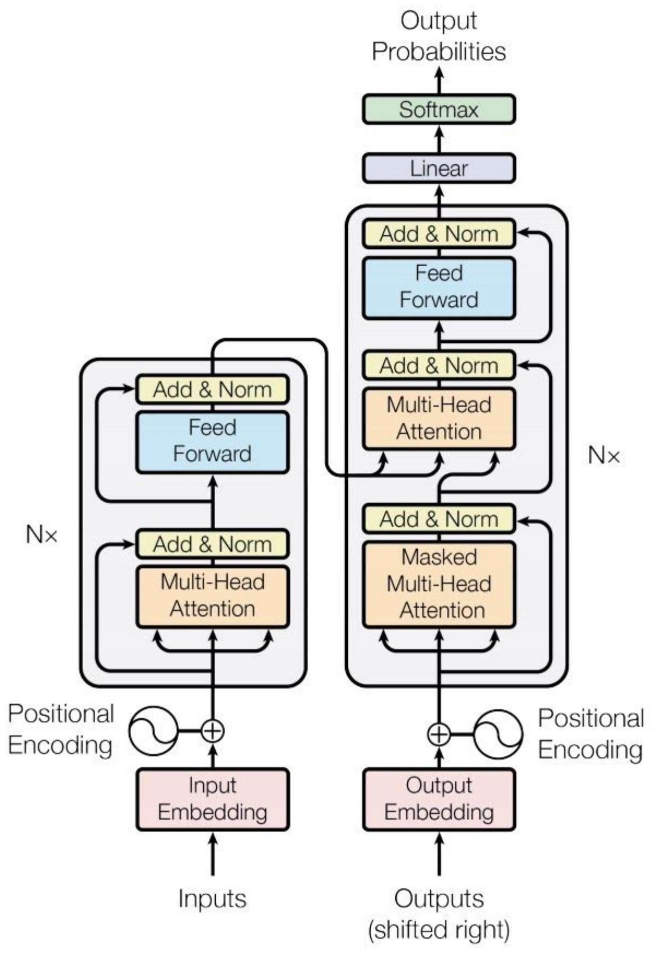


Figure 3: Diagram of the BERT model, presenting the transformer.

The encoders and decoders are part of the transformers that are highlighted in grey in the diagram. Each has a stack of 6 identical layers. The encoder is composed of a multi-head attention mechanism and a position wise fully connected FNN. The FNN is used as two linear transformations with ReLU activation in between, Each of the layers in the encoders and the decoders are followed by an Add/normalizer which is adding the input and reducing the dimensions of the output to be later used in another layer as input. In the decoder, the first step unlike in the encoder, which is the Multi-head Attention function, a Maked Multi-head Attention function is found this is used in the training process and processes the data with needed tags such as [CLS], [SEQ], and [Mask] for training. Finally, the transformation is finished with a linear which converts the output tokens into vectors, which the Softmax uses to predict the next-token probabilities. The weight matrix between the two embedding layers and the softmax is shared, in the embedding the weights are multiplied by the sqrt(dimension of the model).

BERT is trained on a general-domain corpuses:

1. BookCorpus (800M words)
2. English Wikipedia (2,500M words)

    ○ Removed all tables, list and headers in order to keep document-level corpus in order to extract long continuous sequences.

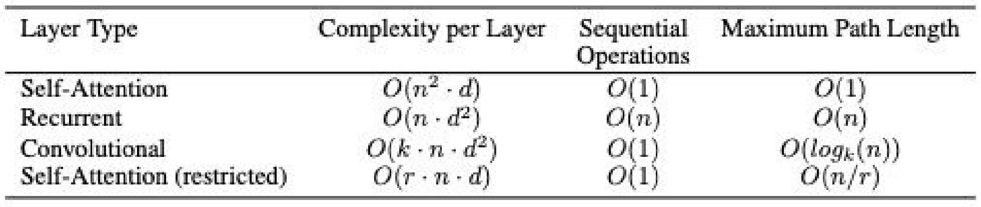
Objective of training:

- Masked language model(MLM): predict the masked token based on its context only. 15% of the corpus is replace with[MASK]
- Next sentence prediction(NSP) task: predict if 2 sentences

MLM and NSP are trained together, with the goal of minimizing the combined loss function of the two strategies.

The prediction of the output word for MLM requires: adding classification layer on top of the encoder output, multiplying the output vector by the embedding matrix, transforming them into vocabulary dimension, calculating the probability of each word in the vocabulary with softmax. To predict second sentence connectivity: the entire input sequence goes through the transformer model, the output of [CLS] token is transformed into a 2x1 shaped vector, using a simple classification layer(learned matrices of weights and biases), calculating the probability of IsNextSequence with softmax.

The complexity of the model is taken from the research.



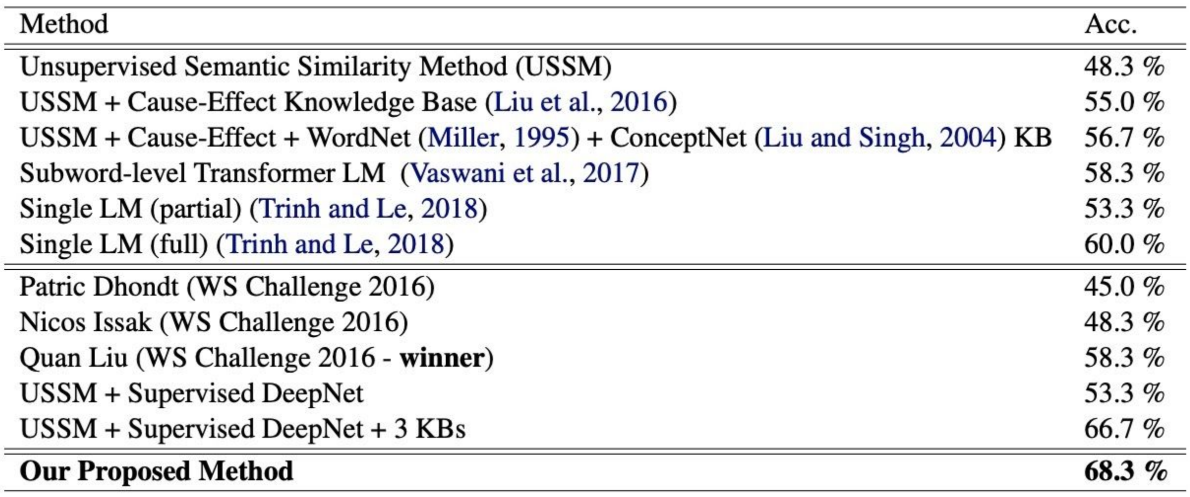
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Figure 4: the complexity table of BERT. Where n = sequence length, d = representation dimension, k = kernel matrix size of convolution and, r = size of the neighborhood in restricted self-attention

This shows a faster complexity compared to recurrent models as self-attention is constant in sequential operations. It is also slightly faster in Maximum Path Length, though both are linear. And Finally, is equivalent in complexity of layers, where self-attention takes into account the size of the neighbourhood, whereas recurrent squares the representation dimension. Self-attention therefore, will perform better in a higher dimensional problem.

Purpose of this project

The purpose of this project is to explore the possibilities of the BERT and to improve on the “Attention Is (Not) All You Need for Commonsense Reasoning.”[T., Klein, 2019] experiment that tries to solve the PDP problem using BERT. This article solely looks at the base uncased model of BERt, however the BERT team periodically adds models to the library, in fact in March of this year smaller models were added to be used on mobile and less resourceful devices. The model used for the experiment yielded an accuracy of 68.3% and surpassed the other model from the 2016 IJCAI conference.



Method	Acc.
Unsupervised Semantic Similarity Method (USSM)	48.3 %
USSM + Cause-Effect Knowledge Base (Liu et al., 2016)	55.0 %
USSM + Cause-Effect + WordNet (Miller, 1995) + ConceptNet (Liu and Singh, 2004) KB	56.7 %
Subword-level Transformer LM (Vaswani et al., 2017)	58.3 %
Single LM (partial) (Trinh and Le, 2018)	53.3 %
Single LM (full) (Trinh and Le, 2018)	60.0 %
Patric Dhondt (WS Challenge 2016)	45.0 %
Nicos Issak (WS Challenge 2016)	48.3 %
Quan Liu (WS Challenge 2016 - **winner**)	58.3 %
USSM + Supervised DeepNet	53.3 %
USSM + Supervised DeepNet + 3 KBs	66.7 %
Our Proposed Method	**68.3 %**

Figure 2: evaluation of BERT as a technique to solve PDP in Attention Is (Not) All You Need for Commonsense Reasoning.

Further, this project will be looking at implementing domain pre-training, which was made available to BERT users. This pre-training function uses a checkpoint file to indicate from where the train is to begin. If no file is given the training will start from scratch, this is said to take a well optimized TPU based system multiple days to run. This project will look at taking multiple different models and further training each of them on data obtained from Google Kaggle competition, Gendered Pronoun resolution. We will take the source code from the "Attention Is (Not) All You Need for Commonsense Reasoning." github[2].

The original directory contained mainly two files, first, commonsense.py. This source code is the main file, it takes arguments for later use, it calls other functions to process the data and runs the BERT function. The arguments are the model to be run, the directory where to find the model, a flag indicating if an uncased model was picked, warning is printed if a mismatch occurs, and a debug flag, which prints each sentence as they are fed into BERT. The function analyzeAttentionSingleTupleDecoy which takes the output from the data_processors.py, returns an attention matrix to be used as BERT's input. An Activity is returned but never used. The Maximum Attention Score(MAS) is used to measure the success of attention on the sentences. The sentences, which contain either pronoun, with the highest score get the highest score. And this is displayed as the accuracy for each individual sentence configuration. Data_processor.py takes as input a XML file which it converts into two objects. The first is an object of sentences divided at the pronoun, called text_a and text_b. And second is the noun that BERT has to choose between to solve the PDP problem. Both configurations return, with the nouns inter-changing the role of ground truth or decoy.

The implementation of the source code must have been done before some update to libraries as some modernization had to be done to make the program runnable. This is the first step, and getting the same result obtained by re-running the experiment from the article

[2] https://github.com/SAP-samples/acl2019-commonsense-reasoning

would help indicate well functionality of the program. First, the program contains two programs one being the PDP solver and the other being the WSC challenge solver. This later was removed from commonsense.py and data_processors.py to save storage space in the directory. This also removed an argument of the program that indicated which program was to run. In data_processors.py, several things were incorrectly operating. First, the XML conversion was not returning the correct element from the tree, therefore falsifying the BERT result later. Second, the function that created the inter-changing roles object, previously mentioned, was not accepting options with more than two choices, therefore, functionality was added to run 2 or more choices. The PDP-60 dataset used several multiple choice sentences, making this modernization necessary. The final change in this file was to fix the guid_dict variable encounter, which keeps track of which sentences correspond to each example. This indicated wrongly sentences resulting in false negatives in BERT. Finally, a new document was added to pre-train BERT. This new file, called "domain_training.py" was called from commonsense.py and listed a new flag "--domain_training" which requests a directory to look for BERT_model and save the new BERT_models once created. The implementation of these features demanded to download the BERT library, which is present in the domain_training directory. As well as, checkpoint documents, also added to domain_traing directory.

Dataset

PDP was presented in 2016 as part of the IJCAI-2016 conference, a dataset of 60 examples were given to the participants of the conference. The PDP-60 dataset is a
A new version of the dataset was created named PDP-62, which is said to be less biased, however, the dataset is not available to be downloaded. For this reason the PDP-60 dataset will be used. Additionally, it offers a better opportunity to be compared to the results found in the article[T. Klein, 2019]. The dataset was directly taken unmodified from the conference's website. The dataset is a collection of sentences manually taken from books (A. Sewell, 1877)(Brunhoff, 1933)(C. Keene, 1942). The sentences were modified to summarize context and backstory into a single sentence, to clarify or simplify or to change gender or number of nouns and pronouns in order to introduce ambiguity. For challenge problems, character names were also changed. A single passage may have been used to create multiple Pronoun Disambiguation Problems. The dataset ws obtain from the NYU statistics department page[3] which converted the plain text into XML trees. The trees represented the root by a collection tag, each of the 60 examples were delimited by a schema tag. The schema tag had six children, those being the text, quote, answer, correctAnswer, humanSubject, source. The text tag contains three sections to represent the text before and after the pronoun and the pronoun itself. The quote section has a quote used as reference to the pronoun referred by quote1 or quote2 tags which differentiated the placement, either before(represented by 1) or after(2) and the pronoun again. The answer section contains two or more possible answers of words that could be used. The correct answer simply includes a letter telling the correct answer, representing the answer in alphabetical order. The final two

[3] https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/PDPChallenge2016.xml

sections contained the human ratio of correct to incorrect answer, and the source of the quote. The conference asked 19 people to answer the question. This was used to analyze the accuracy of the algorithms in the conference.

Experiment

Systems Configuration

The systems available for the experiment were a 2019 MacBook Pro, with 16Gb of RAM on a 2.4GHz Quad-Core Intel i5 CPU with integrated GPU. Additionally, the Compute Canada Compute Server named Graham was made available to this project. The server allows for different system configuration to be used, however, available to me was a 2 CPU, 32 Gb of RAM, one P100 Nvidia GPU on the Pascal architecture. This system was vastly capable of running this type of training procedure. The Canada Compuate Graham server was used to run the training of the domain focus model. Where the Mac was sufficient to simply run the pre-trained model on the data.

BERT Models

This experiment ran four models of BERT on the data provided by PDP-60.

	Base	Large
cased	0.63333	0.4666
uncased	0.68333	0.4666

Figure 3: The results of the BERT's models to solve the PDP problems

Evaluation

BERT Models

The article[T., Klein, 2019] ran a based uncased model on the data and compared to other models. This project continued this experiment by running the same data on other models such as base-uncased, base-cased, large-uncased and large-cased. The models that are offered that are smaller, are offered for less resourceful systems. The bigger were not included as BERT takes a performance hit when run on GPU as it is run on a less than optimal max_batch capacity. To remedy this TPU, such as one available on Colab, should be used coming with a price for processing time. This drop in performance is shown when running the large model, both uncase and cased, takes an accuracy hit compared to base. This, also, surprisingly affected the large model as well, which would have with no doubt been helped by running on a more resourceful system.

Domain-training

The article[T., Klein, 2019] describes a single experiment being done, this experiment measures the accuracy of the BERT model against the answers from the PDP-60 dataset. This is running a *bert-base-uncased* model and given results of 68.3%, which when compared to other supervised and unsupervised models give performance above all. Interestingly, this is the only experiment done to measure the performance of the BERT model, therefore the first experiment to be done is selecting multip[le BERT models to compare to this base. BERT has feature that enables it to be further trained (ie. domain trained) this is said on the BERT github[4] to greatly improve performance even when provided short corpus. This experiment was to look at using different BERT models as a checkpoint to train the model. In the domain_training/BERT_model directory two folders are found with the base and large uncased model as the previous experiment showed superior result for uncased compared to cased models. The corpus provided by the GAP-conference[5] on Kaggle by Google is pre-divided into training, test and validation sets. The training and test is 4454 snippets of wikipedia articles. It is constructed as a gendered unbiased corpus that gives equally divided Male and Female pronouns. Unlike PDP-60 this corpus gives a third option of neither choice being the correct choices to replace the pronoun. PDP-60, however, included non-gendered pronouns, such as they, and them. Google provides gap_scorer.py to evaluate the F1 Scores of the corpus.

Task Setting	**M**	**F**	**B**	**O**
snippet-context	69.4	64.4	*0.93*	66.9
page-context	72.3	68.8	*0.95*	70.6

The metrics are **M**asculine and **F**eminine examples, **O**verall, and a **B**ias factor calculated as F / M.

The program evaluates the corpus against the syntactic parallelism baseline of the test set. The documents are in a TSV file format that has the same information as PDP-60, plus further processing. The TSV files contain the offset of the pronouns and choices in the text that makes it easier to locate the words in the corpus. For our use, only the text is extrapolated and each text section concatenated with a newline (in unix: “\n”), as per the specification of input.txt in the BERT Github under the Pre-Training Bert section. This is run in create_pretraining_data.py outputting a .tfrecord file, which is used in the

[4] https://github.com/google-research/bert
[5] https://github.com/google-research-datasets/gap-coreference

run_pretraining.py, which in turn outputs a BERT_model, in five files which includes the vocab.txt, bert_model.ckpt.meta, .index and .data-00000-of-00001, and a .JSON config file.

This experiment was not produced as problems were encountered when trying to run the experiment on the graham.computecanada.ca server. The server did not allow for download in Python. In data_processors.py, which is used to process the data before being used in BERT, uses nltk.download('punkt') and nltk.download('averaged_perceptron_tagger'). These downloads were timing out and no reply was given by the support team. This being said the expectation of such a process is mixed. On one hand, in Google's github README.md on BERT, it was mentioned that even a small corpus would generally improve the performance of the model. However, it would also say that when running any of the processes of BERT on other hardware than a TPU, this includes CPU and GPU, performance would decrease. This experiment would have drawn the line in the sand as per which of these characteristics of BERT would win over the other.

Conclusion

BERT is a young and powerful model. It has gained large attention from the NLP community and has a string model and is now being explored by many to solve more complex problems, and to create models based upon its architecture. This state-of-the-art model surpasses the previous heavy weight, such as recurrent models previously used for classification. The models offered on the github are highly dependent on the hardware used to evaluate the sentences. In the PDP challenge, the base-uncased model did best when compared to the uncased and large model. This shows that the more trained model(large) even though a better model is affected by the batch size offered by the GPU. Had the pre-training experiment worked on the Compute Canada server, a different evaluation would have been seen. Considering the resources made available by this server, this shows a need for further advancement in algorithms for moderately resourceful machines, such as laptops to run the more complex model. Kaggle participants have been using BERT in conjunction with other techniques to better fine tune it. This offers further performance to the model that a simple run is not capable. Keep an eye on the github as Google often updates the content and models available, offering multilingual and portable versions of the model. Advancement in the sector of attention-based models are in the works, interesting mention of different types of output such as picture, audio and video. BERT is sure to be found to solve the next NLP problems.

Future Work

- Run a glue and\or squad evaluation. Glue is run in IJCAI2016 and could greatly help to compare this novel technique to ones presented to the conference in 2016.
- Implementing MLP trained on ELMo is a powerful tuning tool for BERT to fine tune would be best as shown in Kaggle competition result discussed above.
- Running the Experiment with domain training, as problems were encountered when using the graham.computecanada.ca server. Solving the problems would implicate help from the support team at Compute Canada to install some NLTK data. Additionally, the server is currently over used due to the Global situation (COVID-19).
- In terms of the report a look at previously used techniques would broaden the understanding of the decision made when implementing this algorithm, These may include the previous word embeddings models such as Word2Vec and GloVe. These models make use of single “word embedding” which have the same representation for homonyms. Unidirectional pre-trained models: OpenAI GPT. These models used contextualizing words to its left (or right). For Example, In the sentence “I made a bank account” , the context will only be based on “I made a …” or “account”. Further closer looking at the BERT techniques that Shallowly bidirectional pre-trained model: ELMo used weighted sum of forward (left to right) and backward pass. These techniques bring insight to the work done by the Google Team of researchers to obtain a state-of-the-art model presented.

Build instruction

The project demands for numerous libraries to be downloaded, which can be found in the directory. This has dramatically increased the size of the directory to 1.93Gb. In the acl2019-commonsense-reasoning directory are the folders containing bertviz a visualization library used by the article[T., Klein, 2019]. This is lacking in some of the features of BERT as it was an earlier implementation of the model. Following this the data folder containing the data for the PDP-60 and the Google generated corpus divided in three file train, test and validation. The last directory is the domain_training folder that stores the BERT library, as Bertviz does not implement some of the new functions. Three BERT models were downloaded and stored in the BERT_model, this is done as the domain training runs from checkpoints instead of from scratch. Finally, the new_BERT_folder is used to store the output of the domain trained model that will be used in the commonsense.py file instead of a standard BERT model. The files

The code can be run with or without the pre-training. If no GPU or TPU is available it is recommended not to run the pre-training as it is dramastically slowed down by the CPU. TPU usage is optimal.

To run the code:

1. cd acl2019-commonsense-reasoning #make sure to be in the correct directory.
2. *pip install -r requirements.txt* #command to install the required libraries
3. *python3 commonsense.py --data_dir=./data/ --bert_mode=bert-base-uncased --do_lower_case --debug* #To run bert without pre training.
    a. --bert_model can be:
        i. bert-base-uncased
        ii. bert-base-cased #make sure to set --do_lower_case to False
        iii. bert-large-uncased
        iv. bert-large-cased #make sure to set --do_lower_case to False
    b. Add --domain_training to use the pre-training features that we implemented, it must be given the path to the domain training data by adding:

        --domain_training=./data/gap-conference

To convert TSV into XML: python3 gap_data_to_xml --filename={filename.tsv}

References

Anna Sewell: Black Beauty, J. M. Lupton, 1877

Brunhoff, Jean de. Story of Babar. Random House, 1933.

Devlin, Jacob, et al. BERT: Pre-Training of Deep Bidirectional Transformers for ... nlp.stanford.edu/seminar/details/jdevlin.pdf.

Keene, Carolyn. The Quest of the Missing Map. Grosset & Dunlap, 1942.

Klein, Tassilo, and Moin Nabi. “Attention Is (Not) All You Need for Commonsense Reasoning.” Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, doi:10.18653/v1/p19-1477.

Peters, Matthew, et al. “Dissecting Contextual Word Embeddings: Architecture and Representation.” Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, doi:10.18653/v1/d18-1179.

Vaswani, Ashish, et al. Attention Is All You Need - ArXiv. arxiv.org/pdf/1706.03762.pdf.