"Deep Learning Frameworks for Natural Language Processing: BERT and GPT-2"
Research Computing Center Workshop — Spring 2020

Jeffrey Tharsen tharsen@uchicago.edu

This Guide and ipynb files :
https://github.com/rcc-uchicago/BERT-ELMo_tutorial_Spring2020

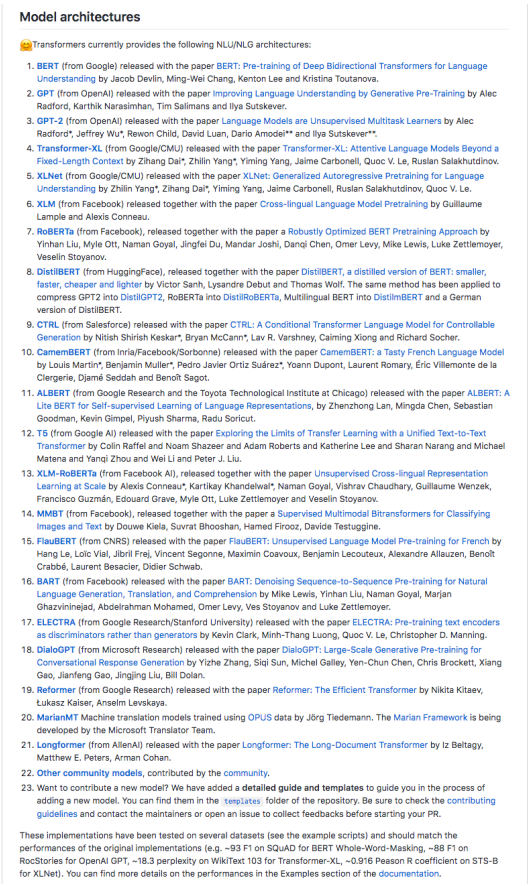
1. "This Grad Student Used a Neural Network to Write His Papers" (GPT-2)
   https://futurism.com/grad-student-neural-network-write-papers

2. **What are Transformers?**

	GPT	Bert	Transformer-XL	XLM	GPT-2	UniLM	XLNet	RoBERTa	Albert	CTRL	DistilBert	Bart	T5
Encoder stack	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decoder stack	✗	✗	≈	✗	≈	✗	≈	✗	✗	≈	✗	✓	✓

Not all models implement the Encoder-Decoder architecture; they are actually only becoming popular now. Transformer-XL, GPT2, XLNet and CTRL approximate a decoder stack during generation by using the hidden state of the previous state as the key & values of the attention module. Side note: all these 🤞 models are implemented in the transformers library or will be soon.

- **https://github.com/google-research/bert** (official Google repository for BERT)
- **https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2seq-af4d7bf14bb8**
- **https://github.com/huggingface/transformers**



Model architectures

🤗Transformers currently provides the following NLU/NLG architectures:

1. **BERT** (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
2. **GPT** (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
3. **GPT-2** (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford*, Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskever**.
4. **Transformer-XL** (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov.
5. **XLNet** (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le.
6. **XLM** (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.
7. **RoBERTa** (from Facebook), released together with the paper a Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov.
8. **DistilBERT** (from HuggingFace), released together with the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter by Victor Sanh, Lysandre Debut and Thomas Wolf. The same method has been applied to compress GPT2 into DistilGPT2, RoBERTa into DistilRoBERTa, Multilingual BERT into DistilmBERT and a German version of DistilBERT.
9. **CTRL** (from Salesforce) released with the paper CTRL: A Conditional Transformer Language Model for Controllable Generation by Nitish Shirish Keskar*, Bryan McCann*, Lav R. Varshney, Caiming Xiong and Richard Socher.
10. **CamemBERT** (from Inria/Facebook/Sorbonne) released with the paper CamemBERT: a Tasty French Language Model by Louis Martin*, Benjamin Muller*, Pedro Javier Ortiz Suárez*, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah and Benoît Sagot.
11. **ALBERT** (from Google Research and the Toyota Technological Institute at Chicago) released with the paper ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut.
12. **T5** (from Google AI) released with the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer by Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu.
13. **XLM-RoBERTa** (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau*, Kartikay Khandelwal*, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov.
14. **MMBT** (from Facebook), released together with the paper a Supervised Multimodal Bitransformers for Classifying Images and Text by Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, Davide Testuggine.
15. **FlauBERT** (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, Didier Schwab.
16. **BART** (from Facebook) released with the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov and Luke Zettlemoyer.
17. **ELECTRA** (from Google Research/Stanford University) released with the paper ELECTRA: Pre-training text encoders as discriminators rather than generators by Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning.
18. **DialoGPT** (from Microsoft Research) released with the paper DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation by Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, Bill Dolan.
19. **Reformer** (from Google Research) released with the paper Reformer: The Efficient Transformer by Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya.
20. **MarianMT** Machine translation models trained using OPUS data by Jörg Tiedemann. The Marian Framework is being developed by the Microsoft Translator Team.
21. **Longformer** (from AllenAI) released with the paper Longformer: The Long-Document Transformer by Iz Beltagy, Matthew E. Peters, Arman Cohan.
22. **Other community models**, contributed by the community.
23. Want to contribute a new model? We have added a **detailed guide and templates** to guide you in the process of adding a new model. You can find them in the `templates` folder of the repository. Be sure to check the contributing guidelines and contact the maintainers or open an issue to collect feedbacks before starting your PR.

These implementations have been tested on several datasets (see the example scripts) and should match the performances of the original implementations (e.g. ~93 F1 on SQuAD for BERT Whole-Word-Masking, ~88 F1 on RocStories for OpenAI GPT, ~18.3 perplexity on WikiText 103 for Transformer-XL, ~0.916 Peason R coefficient on STS-B for XLNet). You can find more details on the performances in the Examples section of the documentation.

- https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f

- http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/ (DistilBert)

3. **What Tasks can Transformers be trained for?**

- https://github.com/huggingface/transformers/blob/master/examples/README.md

> ## The Big Table of Tasks
>
> | Task | Example datasets | Trainer support | TFTrainer support | pytorch-lightning | Colab |
> |---|---|---|---|---|---|
> | language-modeling | Raw text | ✅ | - | - | Open in Colab |
> | text-classification | GLUE, XNLI | ✅ | ✅ | ✅ | Open in Colab |
> | token-classification | CoNLL NER | ✅ | ✅ | ✅ | - |
> | multiple-choice | SWAG, RACE, ARC | ✅ | ✅ | - | Open in Colab |
> | question-answering | SQuAD | - | ✅ | - | - |
> | text-generation | - | - | - | - | Open in Colab |
> | distillation | All | - | - | - | - |
> | summarization | CNN/Daily Mail | - | - | - | - |
> | translation | WMT | - | - | - | - |
> | bertology | - | - | - | - | - |
> | adversarial | HANS | - | - | - | - |

- https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03

- https://medium.com/@Moscow25/the-best-deep-natural-language-papers-you-should-read-bert-gpt-2-and-looking-forward-1647f4438797

- https://medium.com/@ranko.mosic/googles-bert-nlp-5b2bb1236d78

4. **Jupyter Notebook : Single-token prediction with Google's BERT (via Pytorch)**
   BERT_Predict_masked_token.ipynb

5. **Jupyter Notebook: Next sentence predictiom with OpenAI's GPT-2**
   BERT+GPT2_sentence_prediction.ipynb

   https://openai.com/blog/better-language-models/

   The code comes from this tutorial :
   https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/02_how_to_generate.ipynb

6. **Training BERT for various tasks (on Google Cloud TPU via Colab)**

   Google Cloud TPU Quickstart:
   https://cloud.google.com/tpu/docs/quickstart

   https://cloud.google.com/tpu/docs/setup-gcp-account

   In Cloud Console ("Dashboard")
   Create a cloud storage bucket; run the tutorial
   Enable billing – you must put in a credit card number
   "Start my free trial"
   Create bucket (I used "bert_tutorial_12345")

   Colab : Predicting Movie Review Sentiment with BERT on TF Hub ***(requires Tf 1.x, you may need to add* !pip install --upgrade --force-reinstall tensorflow-gpu==1.15.0 *at the beginning)***
   https://colab.research.google.com/github/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb

7. **Training BERT for various tasks on Midway using pre-built models**

   A. **Classification : MRPC (Sentence Equivalence) via GLUE**
   See Appendix I below

   B. **Question & Answer via SquAD (1.1 / 2.0)**
   https://github.com/google-research/bert#squad-11
   https://github.com/google-research/bert#squad-20

8. **How to build ("pre-train") a BERT model from a custom source corpus on Midway**

   A. **The basic procedure :**
   https://github.com/google-research/bert#pre-training-with-bert

   B. Formatting the source corpus : *one line per sentence, one blank line between documents*

   C. Creating vocab.txt (WordPiece tokenization)
   https://github.com/kwonmha/bert-vocab-builder

   D. Setting up bert_config.json (make sure the vocab length is correct)

   Tips and best practices: **https://ufal.mff.cuni.cz/pbml/110/art-popel-bojar.pdf**

9. **How to build a GPT-2 model from a custom corpus**

   https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f

   https://towardsdatascience.com/train-a-gpt-2-transformer-to-write-harry-potter-books-edf8b2e3f3db

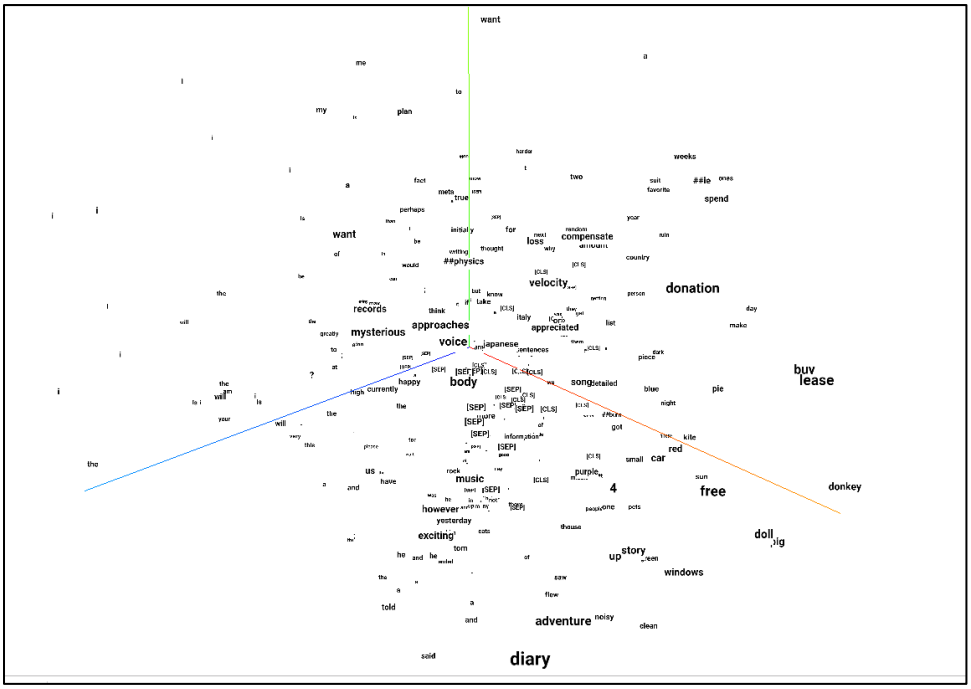
10. **Retrieving and Visualizing the vectors from a BERT model (t-SNE with PCA)**

    A. **Google's method :**
    https://github.com/google-research/bert#using-bert-to-extract-fixed-feature-vectors-like-elmo

    B. **The "bert-embedding" library on Github :**
    https://github.com/negedng/bert-embedding

    C. **Colab (for Tf 2.x) :**
    https://colab.research.google.com/drive/1hMLd5-r82FrnFnBub-B-fVW78Px4KPX1#scrollTo=_Iwew0KP8vRM

    D. **Use Tensorflow's Tensorboard Embeddings Projector (3 dimensions) :**
    https://towardsdatascience.com/bert-visualization-in-embedding-projector-dfe4c9e18ca9



Appendix I. Using Midway to train BERT for Classification : MRPC (Sentence Equivalence) via GLUE

I. Clone or download Google's main BERT repository:
git clone https://github.com/google-research/bert.git

II. Download one of the pre-trained Google BERT models:
https://github.com/google-research/bert#pre-trained-models

e.g. **BERT-Base, Uncased** (recommended)
12-layer, 768-hidden, 12-heads, 110M parameters
https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

Open Firefox, download the zip file, then move it to your working directory and unzip it:

mv ~/Downloads/uncased_L-12_H-768_A-12.zip .
unzip uncased_L-12_H-768_A-12.zip

III. The model zip file will contain:

a.	vocab.txt	BERT-Base, Uncased = 30,522
b.	bert_config.json	the all-important config file
c.	bert_model.ckpt.meta	binary metadata file
d.	bert_model.ckpt.index	binary index file
e.	bert_model.ckpt.data-00000-of-00001	the DATA, aka "the checkpoint file"

IV. To run the GLUE test
First download download_glue_data.py script from
https://gist.github.com/W4ngatang/

module load Anaconda3/2018.12

Run : python download_glue_data.py

cd bert

Create an sinteractive session for yourself on Midway :

sinteractive –mem-per-cpu=24G
(or : sinteractive –mem-per-cpu=24G –partition=gpu2 –gres=gpu:1)

Load the modules and Tensorflow 1.13.1 environment :

module load Anaconda3/2018.12
source activate tf-cpu-1.13.1 (or source activate tf-gpu-1.13.1)

Export these directories (makes it easier to run run_classifier.py) :

export BERT_BASE_DIR=/path/to/bert/uncased_L-12_H-768_A-12
export GLUE_DIR=/path/to/glue_data

To train the classifier, run this:

```
python run_classifier.py \
 --task_name=MRPC \
 --do_train=true \
 --do_eval=true \
 --data_dir=$GLUE_DIR/MRPC \
 --vocab_file=$BERT_BASE_DIR/vocab.txt \
 --bert_config_file=$BERT_BASE_DIR/bert_config.json \
 --init_checkpoint=$BERT_BASE_DIR/bert_model.ckpt \
 --max_seq_length=128 \
 --train_batch_size=32 \
 --learning_rate=2e-5 \
 --num_train_epochs=3.0 \
 --output_dir=/tmp/mrpc_output/
```

→ *change this to somewhere useful*

Then to run it, set TRAINED_CLASSIFIER to the output directory

```
export TRAINED_CLASSIFIER=/tmp/mrpc_output/
```

→ *what you used above*

Create the file test.tsv in the /bert directory (see below for a sample);
the process will create test_results.tsv in your output_dir.

When test.tsv is ready, run this to create test_results.tsv in the output_dir :

```
python run_classifier.py \
 --task_name=MRPC \
 --do_predict=true \
 --data_dir=$GLUE_DIR/MRPC \
 --vocab_file=$BERT_BASE_DIR/vocab.txt \
 --bert_config_file=$BERT_BASE_DIR/bert_config.json \
 --init_checkpoint=$TRAINED_CLASSIFIER \
 --max_seq_length=128 \
 --output_dir=/tmp/mrpc_output/
```

→ *change this also to somewhere useful,*
I use mrpc_test

Here are samples of both of these files (note that they are tab-delimited):

test.tsv :

guid	text_a	text_b
casd4	I hate pizza, as everone knows.	Pizza is the food I hate the most.
3ndf9	She is the greatest singer in the world.	As a singer, she's just the very best.
abcde	She loves me not.	Are penguins cold?

test_results.tsv will contain a list of vectors, to combine with test.tsv, do:

```
import pandas as pd #read the original test data for the text and id
df_test = pd.read_csv('bert/test.tsv', sep='\t')
#read the results data for the probabilities
df_result = pd.read_csv('mrpc_test/test_results.tsv', sep='\t', header=None)   #change to the output dir
#create a new dataframe
df_map_result = pd.DataFrame({'guid': df_test['guid'],
   'text_a': df_test['text_a'],
   'text_b': df_test['text_b'],
   'label': df_result.idxmax(axis=1)})
#view the first 3 rows of the newly created dataframe
print(df_map_result[0:3])
```

The final results should look something like this :

	guid	text_a	text_b	label
0	casd4	I hate pizza, as everone knows.	Pizza is the food I hate the most.	1
1	3ndf9	She is the greatest singer in the world.	As a singer, she's just the very best.	1
2	abcde	She loves me not.	Are penguins cold?	0