*“Deep Learning Frameworks for Natural Language Processing: BERT and GPT-2”*

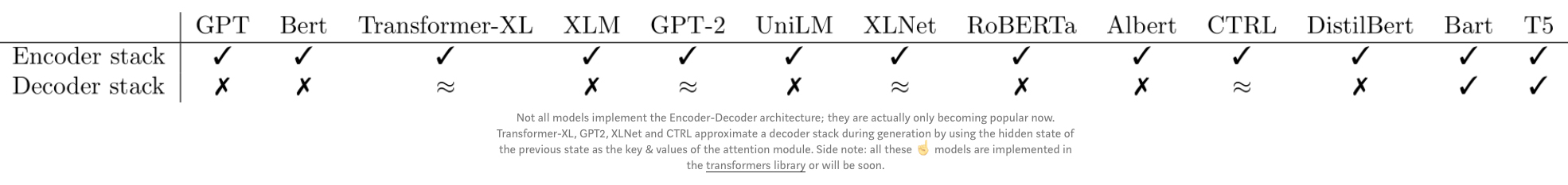
*Research Computing Center Workshop — Spring 2020*

*Jeffrey Tharsen tharsen@uchicago.edu*

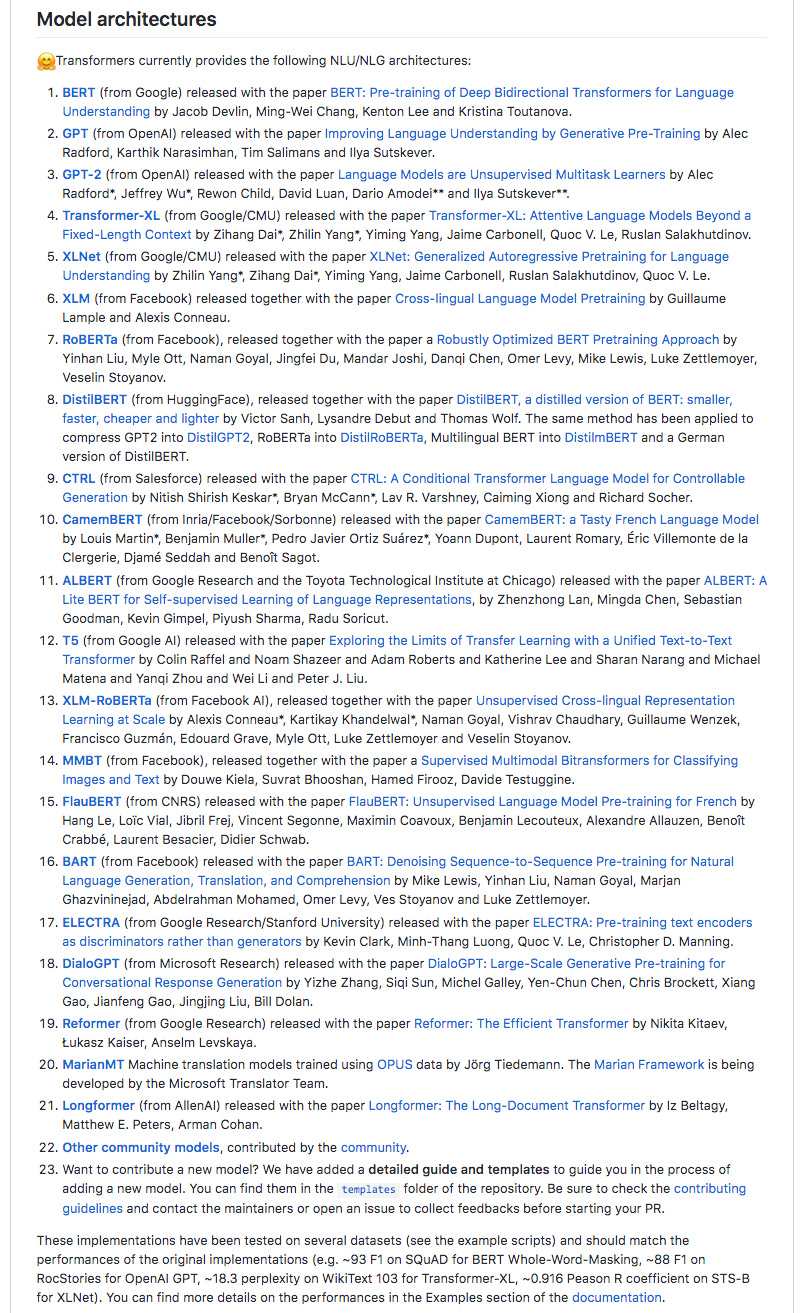
***This Guide and ipynb files :***

[**https://github.com/rcc-uchicago/BERT-ELMo\_tutorial\_Spring2020**](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?**

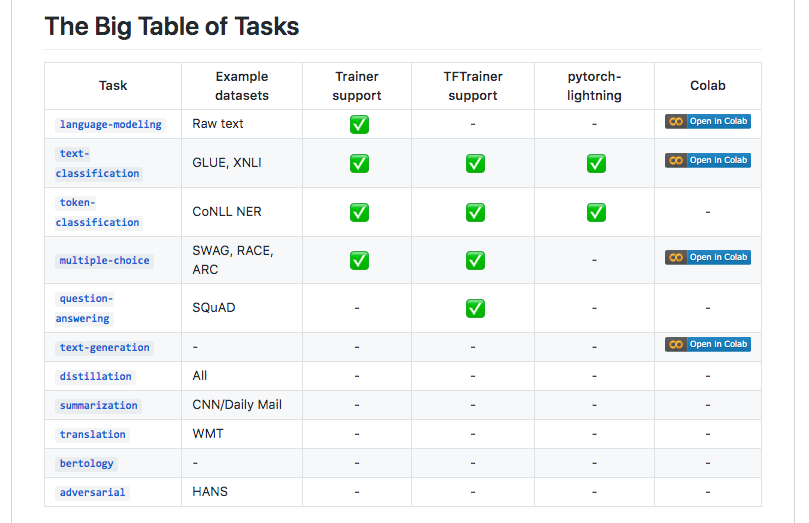


* [**https://github.com/google-research/bert**](https://github.com/google-research/bert) (official Google repository for BERT)
* [**https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2seq-af4d7bf14bb8**](https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2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



* <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)

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

* <https://github.com/huggingface/transformers/blob/master/examples/README.md>   
  
* <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>

1. **Jupyter Notebook : Single-token prediction with Google’s BERT (via Pytorch)**

**BERT\_Predict\_masked\_token.ipynb**

1. **Jupyter Notebook: Next sentence predictiom with OpenAI’s GPT-2**

**BERT+GPT2\_sentence\_prediction.ipynb**  
[**https://openai.com/blog/better-language-models/**](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>

1. **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/quickstart)

[***https://cloud.google.com/tpu/docs/setup-gcp-account***](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>

1. **Training BERT for various tasks on Midway using pre-built models**
2. **Classification : MRPC (Sentence Equivalence) via GLUE  
   See Appendix I below**
3. **Question & Answer via SquAD ( 1.1 / 2.0 )**

<https://github.com/google-research/bert#squad-11><https://github.com/google-research/bert#squad-20>

1. **How to build (“pre-train”) a BERT model from a custom source corpus on Midway**
2. **The basic procedure :**   
   [**https://github.com/google-research/bert#pre-training-with-bert**](https://github.com/google-research/bert#pre-training-with-bert)
3. Formatting the source corpus : *one line per sentence, one blank line between documents*
4. Creating vocab.txt ( WordPiece tokenization )  
   <https://github.com/kwonmha/bert-vocab-builder>
5. 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**](https://ufal.mff.cuni.cz/pbml/110/art-popel-bojar.pdf)

1. **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>

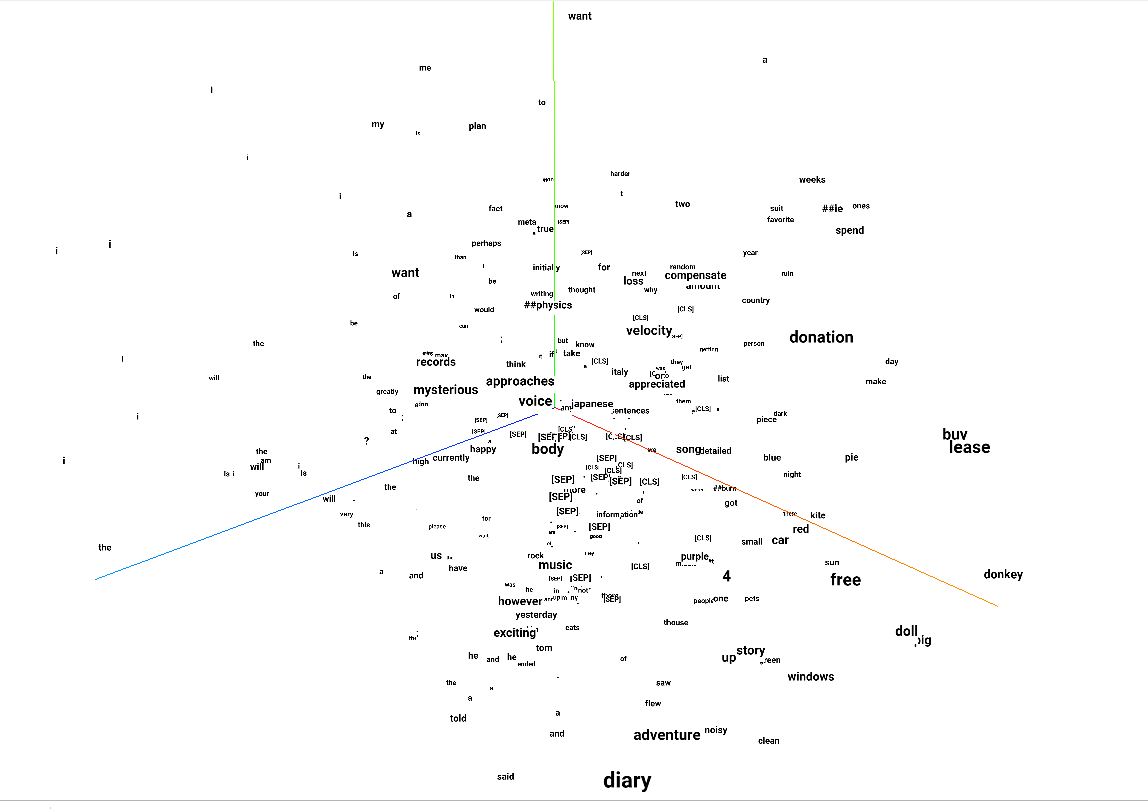
<https://towardsdatascience.com/train-a-gpt-2-transformer-to-write-harry-potter-books-edf8b2e3f3db>

1. **Retrieving and Visualizing the vectors from a BERT model (t-SNE with PCA)**
2. **Google’s method :**<https://github.com/google-research/bert#using-bert-to-extract-fixed-feature-vectors-like-elmo>
3. **The “bert-embedding” library on Github :**<https://github.com/negedng/bert-embedding>
4. **Colab (for Tf 2.x) :**

<https://colab.research.google.com/drive/1hMLd5-r82FrnFnBub-B-fVW78Px4KPX1#scrollTo=_Iwew0KP8vRM>

1. **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**

1. Clone or download Google’s main BERT repository:  
   git clone <https://github.com/google-research/bert.git>
2. 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

1. **The model zip file will contain:**
   1. vocab.txt BERT-Base, Uncased = 30,522
   2. bert\_config.json the all-important config file
   3. bert\_model.ckpt.meta binary metadata file
   4. bert\_model.ckpt.index binary index file
   5. bert\_model.ckpt.data-00000-of-00001 the DATA, aka “the checkpoint file”
2. **To run the GLUE test**

First download download\_glue\_data.py script from

[*https://gist.github.com/W4ngatang/*](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