## Assumptions:

You are running this code on Windows 10 machine that has ample RAM and a CUDA capable gpu.

## What to install:

* Bazel (required for BERT local)
  + Instructions: <https://docs.bazel.build/versions/master/install-windows.html>
* Python
  + Instructions: <https://docs.conda.io/projects/conda/en/latest/user-guide/install/>
* Docker (optional)
  + Instructions: <https://docs.docker.com/docker-for-windows/wsl/>
  + Instructions require installation of WSL2
* Latest Nvidia Drivers (recommended)
  + Instructions: <https://docs.nvidia.com/cuda/wsl-user-guide/index.html>
  + Instructions require installation of other software and packages you must follow all
* BERT Model (optional)
  + Google BERT Models: <https://github.com/google-research/bert>
  + SciBERT Model: <https://github.com/allenai/scibert/>

## Prepare Data:

### Generate json info for each dataset

Script: “./competition/create\_bert\_data.py”

Purpose: combine and convert the usable metadata and file contents into a single file for consumption by other scripts

Command line:

set BASE=%CD%

python create\_bert\_data.py ^

--vocab\_file %BASE%\datasets\scibert\_vocab.txt ^

--run\_type "train;test" ^

--input\_dir %BASE%\datasets ^

--output\_dir %BASE%\json\_data ^

--tokenize

### Convert BERT model checkpoints to required format

Script: “./competition/checkpoint\_converter.py”

Purpose: convert a BERT model that is saved in an order checkpoint format into the TF2.0+ format

Source: adapted from tensorflow model garden

Command line:

set BASE=%CD%

set BERT\_DIR=%BASE%\bert\checkpoints\uncased\_L-4\_H-256\_A-4\_TF2

python checkpoint\_converter.py ^

--bert\_config\_file=%BERT\_DIR%\bert\_config.json ^

--checkpoint\_to\_convert=%BERT\_DIR%\bert\_model.ckpt ^

--converted\_checkpoint\_path=%BERT\_DIR%\converted\_model.ckpt ^

--checkpoint\_model\_name=tfrbert ^

--converted\_model=encoder

## Run Cranfield Data (recommended for Reviewers):

Different combinations of arguments can produce a large variety of results. I have added the arguments required to produce some of the better results that I have come across during my testing

### Prep Cranfield Data

Script: “./competition/cranfield\_metapy/create\_cranfield.py”

Purpose: to create the cranfield datasets for use with metapy

Command Line:

set BASE=%CD%

python create\_cranfield.py ^

--run\_type "train;test" ^

--query\_keys "query" ^

--doc\_keys "title;abstract:intro;text" ^

--cranfield\_dir "%BASE%" ^

--input\_dir "%BASE%\..\json\_data"

### Run Ranker

Script: “./competition/cranfield\_metapy/search\_eval.py”

Purpose: to rank the documents using the specified ranker

Command line:

set BASE=%CD%

python search\_eval.py ^

--run\_type "test" ^

--ranker "bm25+" ^

--params "1.0;0.38;1.0" ^

--dat\_keys "title;abstract;text" ^

--doc\_weights "0.8;0.5;0.0" ^

--cranfield\_dir "%BASE%" ^

--predict\_dir "%BASE%\..\.." ^

--remove\_idx

### Simple BM25

Run create\_cranfield.py with modified arguments:

--run\_type “test”

--doc\_keys “title:abstract:intro”

Run search\_eval.py with modified arguments:

--ranker “bm25”

--params “0.25;0.35;0”

--doc\_weights “1”

--dat\_keys “title”

### Bm25+

Run create\_cranfield.py with modified arguments:

--run\_type “test”

--doc\_keys “title:abstract:intro”

Run search\_eval.py with modified arguments:

--ranker “bm25+”

--params “1.0;0.38;1.0”

--doc\_weights “1”

--dat\_keys “title”

### BM25+ with Rocchio Feedback

Run create\_cranfield.py with modified arguments:

--run\_type “test”

--doc\_keys “title:abstract:intro”

Run search\_eval.py with modified arguments:

--ranker “rocchio”

--params “1.0;0.38;1.0”

--doc\_weights “1”

--dat\_keys “title”

## Run BERT Model:

### Choose BERT Model (Google’s BERT vs SciBERT)

Note: Due the time and resources required to run these models I do not suggest that reviewers take this path.

The links to install the BERT models are located in the “What to Install” section.

### Generate ELWC files for train data

### Train model

### Generate Predictions

### Run directly

### Run on Docker with Windows Subsystem for Linux (not recommended solution for project)

Convert BERT Models:

pip install tf-models-nightly

python checkpoint\_converter.py --bert\_config\_file=%MODEL\_DIR%/bert\_config.json --checkpoint\_to\_convert=%MODEL\_DIR%/bert\_model.ckpt --converted\_checkpoint\_path=%MODEL\_DIR%/bert\_modelv2.ckpt --checkpoint\_model\_name=scibert --converted\_model=encoder

Attempts

* Cranfield
  + Combine query, question, and narrative into and run as one query on all docs
  + Combine title, abstract, and body text and query over all
  + Combine title, abstract and query over all
  + Separate title, abstract, and body text. Rank over each individual component and using weighting to combine rankings
  + Ranking using:
    - DirichletPrior
    - JelinekMercer
    - OkapiBM25
    - BM25+
    - BM25+ and Rocchio (pseudo) Feedback
* BERT
  + Using narrative
  + Using query
  + Using all docs
  + Using ranking results from BM25
  + Splitting docs into numerous chunks due to size limitations
    - Tested using average of all chunk ranks, geometric mean of chunks, max chunk ranking, and first chunk ranking

## Resource Links:

<https://docs.vespa.ai/documentation/reference/bm25.html>

<https://github.com/vespa-engine/cord-19/blob/master/cord-19-queries.md>

<http://cognitiveai.org/2020/09/08/using-tensorflow-ranking-bert-tfr-bert-an-end-to-end-example/>

<https://github.com/cognitiveailab/ranking>

<https://github.com/tensorflow/models/tree/master/official/nlp/bert>

<https://ai.googleblog.com/2018/12/tf-ranking-scalable-tensorflow-library.html>

<https://arxiv.org/pdf/1812.00073.pdf>

<https://github.com/tensorflow/ranking>