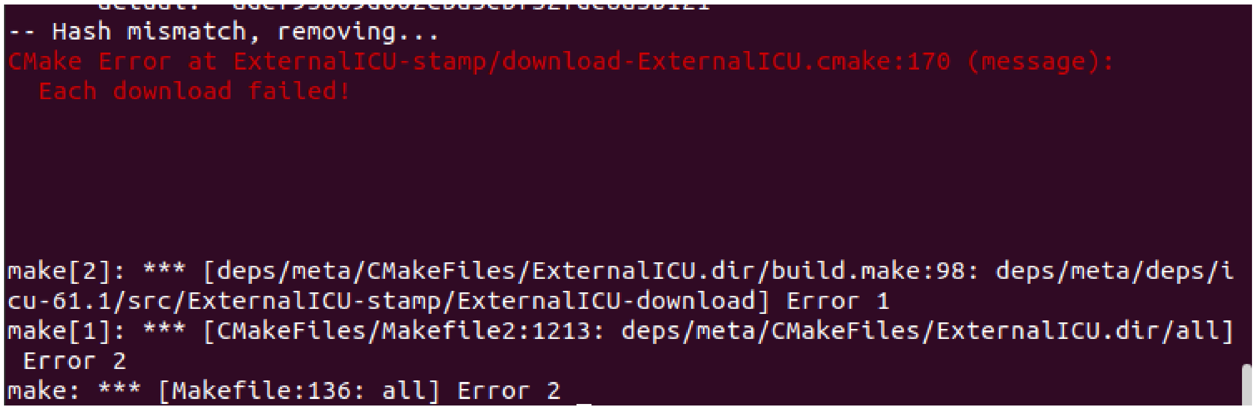
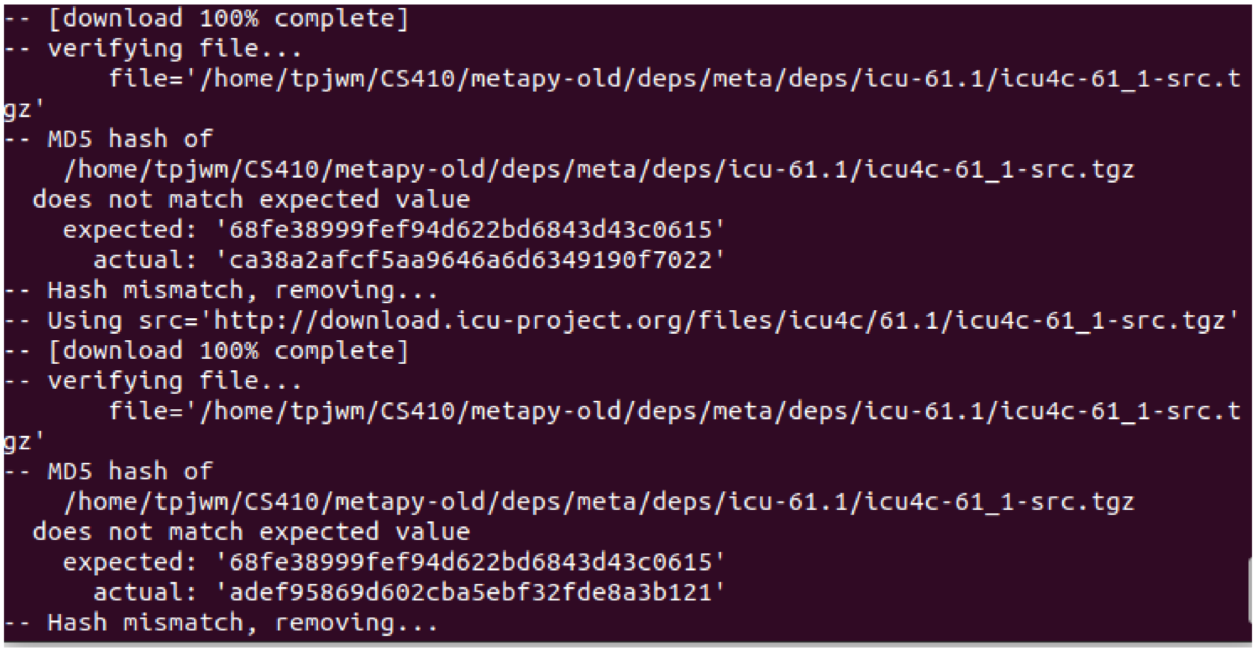
Completed so far for Task 1) Making metapy compatible with recent versions of Python.



```
-- Hash mismatch, removing...
CMake Error at ExternalICU-stamp/download-ExternalICU.cmake:170 (message):
  Each download failed!




make[2]: *** [deps/meta/CMakeFiles/ExternalICU.dir/build.make:98: deps/meta/deps/i
cu-61.1/src/ExternalICU-stamp/ExternalICU-download] Error 1
make[1]: *** [CMakeFiles/Makefile2:1213: deps/meta/CMakeFiles/ExternalICU.dir/all]
 Error 2
make: *** [Makefile:136: all] Error 2
```

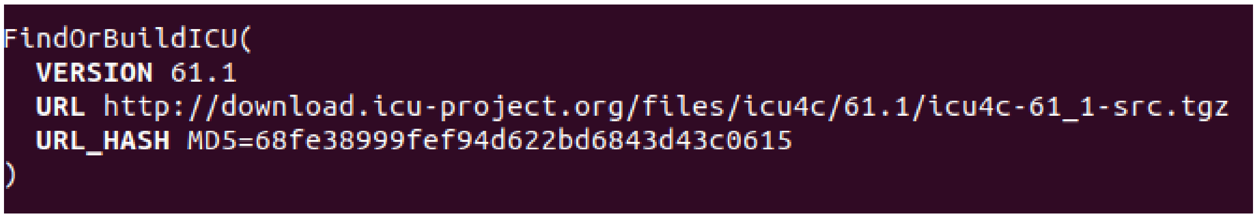
This is the error you get when you try to build metapy with a recent version of python (in this case 3.8.10).

The makefile is attempting to download the ICU file specified from "http://download.icu-project.org/files/icu4c/61.1/icu4c-61_1-src.tgz", but it fails multiple times with a different MD5 hash each time:



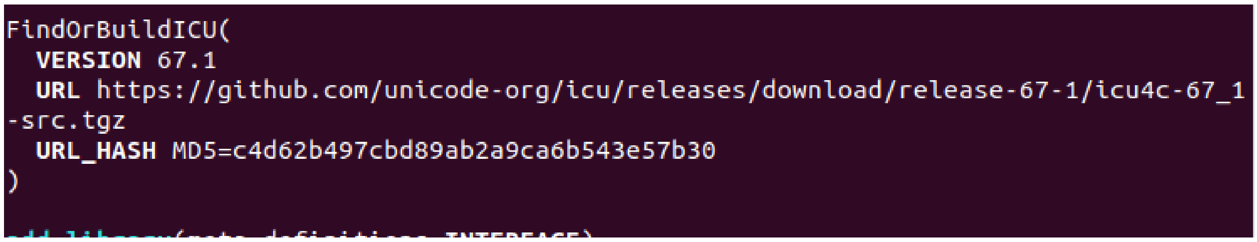
```
-- [download 100% complete]
-- verifying file...
       file='/home/tpjwm/CS410/metapy-old/deps/meta/deps/icu-61.1/icu4c-61_1-src.t
gz'
-- MD5 hash of
    /home/tpjwm/CS410/metapy-old/deps/meta/deps/icu-61.1/icu4c-61_1-src.tgz
  does not match expected value
    expected: '68fe38999fef94d622bd6843d43c0615'
      actual: 'ca38a2afcf5aa9646a6d6349190f7022'
-- Hash mismatch, removing...
-- Using src='http://download.icu-project.org/files/icu4c/61.1/icu4c-61_1-src.tgz'
-- [download 100% complete]
-- verifying file...
       file='/home/tpjwm/CS410/metapy-old/deps/meta/deps/icu-61.1/icu4c-61_1-src.t
gz'
-- MD5 hash of
    /home/tpjwm/CS410/metapy-old/deps/meta/deps/icu-61.1/icu4c-61_1-src.tgz
  does not match expected value
    expected: '68fe38999fef94d622bd6843d43c0615'
      actual: 'adef95869d602cba5ebf32fde8a3b121'
-- Hash mismatch, removing...
```

The offending makefile dependency is found in metapy/deps/meta/CMakeLists.txt



```
FindOrBuildICU(
  VERSION 61.1
  URL http://download.icu-project.org/files/icu4c/61.1/icu4c-61_1-src.tgz
  URL_HASH MD5=68fe38999fef94d622bd6843d43c0615
)
```

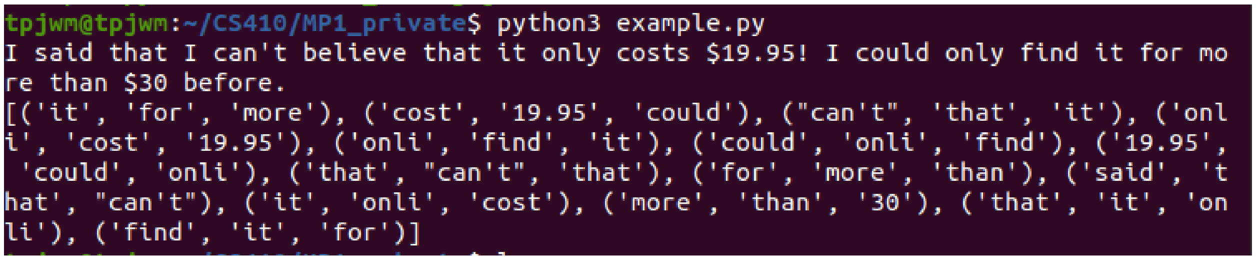
I followed this link and it redirected to the download page for all versions of ICU. I went to the github page instead and found a more recent release with the same format and used that instead. So the updated makefile with the link and new MD5 hash which looks like this:



```
FindOrBuildICU(
  VERSION 67.1
  URL https://github.com/unicode-org/icu/releases/download/release-67-1/icu4c-67_1
-src.tgz
  URL_HASH MD5=c4d62b497cbd89ab2a9ca6b543e57b30
)
```

Now when making the build with python 3.8+, the build succeeds.

Testing it on MP1 example shows that metapy works on the newest version now:



```
tpjwm@tpjwm:~/CS410/MP1_private$ python3 example.py
I said that I can't believe that it only costs $19.95! I could only find it for mo
re than $30 before.
[('it', 'for', 'more'), ('cost', '19.95', 'could'), ("can't", 'that', 'it'), ('onl
i', 'cost', '19.95'), ('onli', 'find', 'it'), ('could', 'onli', 'find'), ('19.95',
 'could', 'onli'), ('that', "can't", 'that'), ('for', 'more', 'than'), ('said', 't
hat', "can't"), ('it', 'onli', 'cost'), ('more', 'than', '30'), ('that', 'it', 'on
li'), ('find', 'it', 'for')]
```

ISSUES: I tried to fork meta and metapy and have a working updated repository, but ran into a lot of trouble with git submodules. Locally when I make these changes meta works, but I kept running into an error with a missing function when uploading these changes to github forks of meta and metapy and connecting them. I believe that this a problem with metapy using a developmental branch of meta (one that is more recently updated) so when I try to link my meta fork to metapy through git submodules, it causes additional errors. My solution to this was to upload a zip file of my local version of metapy to a github repository and link it as a submodule of our CourseProject Fork.

Pending Tasks:

- Currently, this fix only works with Ubuntu Linux 20.04 LTS. Pending time constraints we can also try to figure out a way to make metapy compatible with newer versions of python on the macOS and Windows operating systems.
- Further testing with other MPs and with other versions of python.

Completed so far for Task 2) Addition of other ranker functions using Metapy

This task was more research oriented, and one of the original goals of our project was to investigate different retrieval functions and implement another ranker function that is not included in the current version of metapy. Currently, the existing ranking functions in meta are the following: Absolute Discount, Dirichlet Prior, Jelinek-Mercer, KL-Divergence, LM Ranker, Okapi-BM25, and Pivoted Length.

We were curious about how different the outputs look like based on the ranker function, as each ranker function has its trade-offs. The goal of implementing a new ranker in metapy was to give more options to metapy users to rank text documents, and to find out which ranker function is overall the best one.

After doing some research, we discovered several different variants of the already existing Okapi-BM25 function. (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7148026/)

For reference, the standard Okapi-BM25 function is

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

One of these functions is BM25+, which introduces a new free parameter.

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \left[\frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} + \delta\right]$$

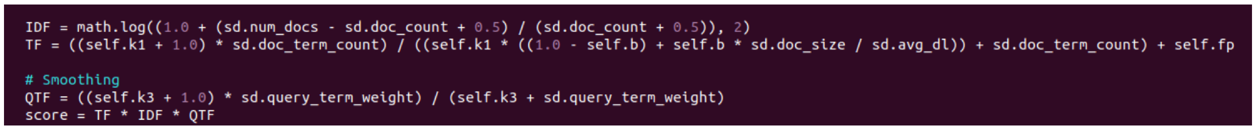
BM25+ was originally developed as an extension to BM25. One of the disadvantages of BM25 is that the component of TF normalization by doc length is not properly lower-bounded, which results in long documents (which may not contain the query term) being scored unfairly as having the same relevancy as shorter docs that do not contain the query term at all.

The free parameter (denoted as δ) is usually set to 1.0 in the absence of training data.

Theoretically, BM25+ should score documents with slightly more fairness as a result of the parameter.

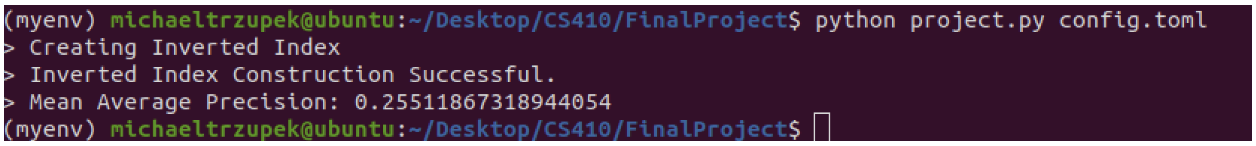
We implemented BM25+ using python. In order to do simple testing of BM25+, we used the skeleton of writing our own ranker and cranfield datasets provided from MP2.2. For now, we decided to keep it simple just to make sure if our implementation would work.

The BM25+ code consists of the following:



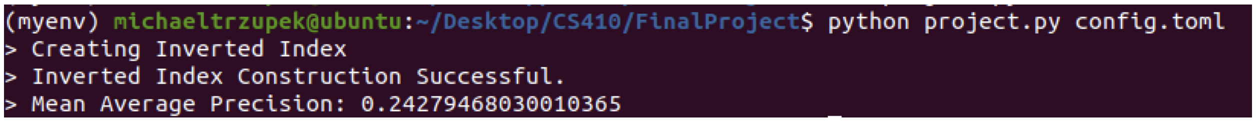
```
IDF = math.log((1.0 + (sd.num_docs - sd.doc_count + 0.5) / (sd.doc_count + 0.5)), 2)
TF = ((self.k1 + 1.0) * sd.doc_term_count) / ((self.k1 * ((1.0 - self.b) + self.b * sd.doc_size / sd.avg_dl)) + sd.doc_term_count) + self.fp

# Smoothing
QTF = ((self.k3 + 1.0) * sd.query_term_weight) / (self.k3 + sd.query_term_weight)
score = TF * IDF * QTF
```

Running the standard version of Okapi-BM25 yields the following:
Note: BM25 parameters were passed in with default values (k1=1.2,b=0.75,k3 = 500)
MAP = ~0.2551186



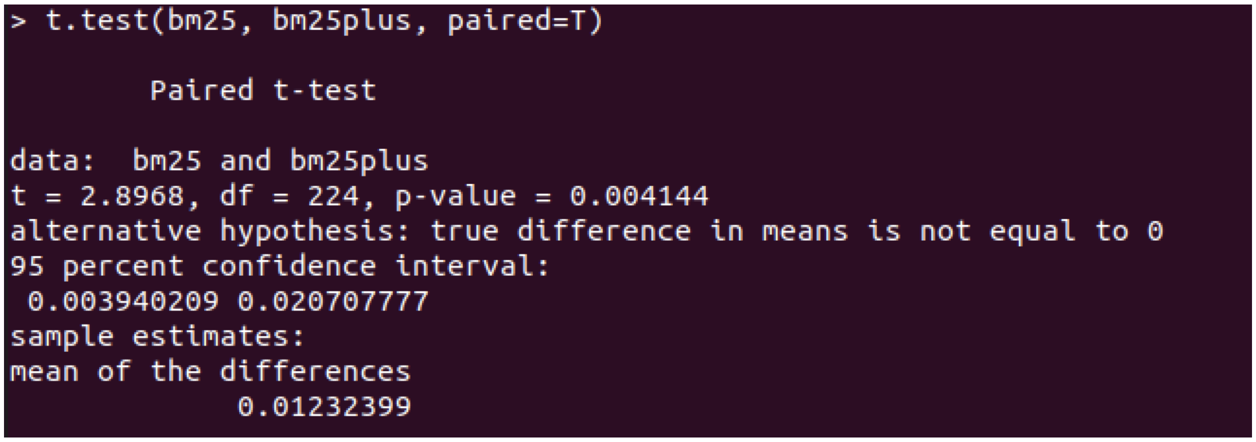
```
(myenv) michaeltrzupek@ubuntu:~/Desktop/CS410/FinalProject$ python project.py config.toml
> Creating Inverted Index
> Inverted Index Construction Successful.
> Mean Average Precision: 0.25511867318944054
(myenv) michaeltrzupek@ubuntu:~/Desktop/CS410/FinalProject$
```

Resultant output from BM25+ ranker. In our code, we set the free parameter to 1.0.
MAP = ~0.24279



```
(myenv) michaeltrzupek@ubuntu:~/Desktop/CS410/FinalProject$ python project.py config.toml
> Creating Inverted Index
> Inverted Index Construction Successful.
> Mean Average Precision: 0.24279468030010365
```

Using R, we used the significance test to find the p-value.



```
> t.test(bm25, bm25plus, paired=T)

        Paired t-test

data:  bm25 and bm25plus
t = 2.8968, df = 224, p-value = 0.004144
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.003940209 0.020707777
sample estimates:
mean of the differences
             0.01232399
```

Since the p-value is < 0.05, there is a significant difference between the average precisions of the two functions, illustrating just how significant a seemingly small change in the function is and how influential the free parameter of BM25+ is.

ISSUES:

- I ran into issues uploading the code to our final project repository, thus, you may be able not see the actual code just yet, although a fix is in progress.

Pending Tasks:

- Introducing more BM25 variants (such as BM25L)/introducing other ranker functions and comparing the different results to find if there is a better ranker for the current dataset.
- Testing the functions using custom query datasets instead of cranfield to check different outputs using documents of various lengths.
- If time permits, we could implement BM25+ within MeTa itself using C++ instead of python, thus enabling future users to simply call BM25+ like so: `metapy.index.BM25plus(k1, b, k3, free_parameter)`

Resources:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7148026/