

Hints on How to Implement Pseudo Feedback using Rocchio

[Subscribe for email updates.](#) PINNED  APPROVED No tags yet. [+ Add Tag](#)Sort replies by: [Oldest first](#) [Newest first](#) [Most popular](#)[Hussein Hazimeh](#) STAFF · 5 days ago  PINNED  APPROVED

Implementing pseudo feedback using Rocchio requires extracting the terms in the top k documents which are blindly assumed to be relevant. To extract the terms in each of these documents efficiently, you should use a forward index. A forward index is a data structure that allows accessing the postings list of a document using its ID, i.e. the forward index has document IDs as keys and postings lists as values. Below we provide some instructions and sample codes that should be useful for you to implement a forward index and update the weight vector of a query.

First include the following two header files at the top of **competition.cpp**:

```
#include "index/forward_index.h"
#include "index/postings_data.h"
```

To create the forward index, add the following to the top of the main function (for example directly below the line that defines idx):

```
auto fidx = meta::index::make_index<index::memory_forward_index>(argv[1]); // fidx is a pointer to the forward index
```

See MeTA's [forward index reference page](#) for more information about the built-in functions of the forward index.

To access the postings list of the i th top document in the vector **ranking** (which is defined in **competition.cpp**), you can use:

```
auto postings_list = fidx->index::forward_index::search_primary(ranking[i].first); // postings_list is a pointer to the postings list of the document
```

Visit MeTA's [postings data reference page](#) for more information on this data structure. **postings_list->counts()** will return a vector of pairs where each pair has the form $(term, frequency)$. This will allow you to extract the terms and their corresponding frequencies from each document.

Since queries are instantiated from the document class, you can use the built-in functions in the document class to modify a query. Check the [document class reference page](#). One important member is the **increment** function which allows you to increase the weight of a certain term in the query.

The code below retrieves the top 10 documents based on the original query, and then creates a new query which is the sum of the old query vector and the highest ranked document. Finally, it calls the ranking function again on the new query. Note that what this code is doing may not be ideal since it is considering only 1 feedback document and is assigning equal weights to the original query and the feedback document. When you implement Rocchio, you should experiment with the weights of the original query and the positive documents set, in addition to the number of top documents to consider (5, 10, 20 ? The MAP on the training queries should give you a good answer).

```
/* The code assumes that the inverted and forward indices were instantiated before in the main
with pointers named idx and fidx, respectively. It also assumes that the string content contain
s the text of the current query */

corpus::document query; // Declare the original query

query.content(content); // Set the content of the original query

idx->tokenize(query); // Tokenize the original query - Make sure to add this!

auto ranking = ranker->score(*idx, query, 10);

auto postings_list = fidx->index::forward_index::search_primary(ranking[0].first); // Return t
he postings list of the top document in ranking

auto term_freq = postings_list->counts(); // term_freq contains the pairs of the form <term_i
d, term_freq>

corpus::document new_query; // Declare the new query

// Add the terms of the original query to the new query
for (auto& qpair : query.counts()) // Loop over the original query's pairs
    new_query.increment(qpair.first,qpair.second); // qpair.first is the term name and qpair.s
econd is the term frequency

// Add the term of the top ranked document to the new query
for (auto& tfpair : term_freq) // Loop over the document's pairs
    new_query.increment(idx->term_text(tfpair.first),tfpair.second); // idx->term_text convert
s a term id to the corresponding term name

ranking = ranker->score(*idx, new_query, 50); // Call the ranker again on the new modified que
ry
```

↑ 6 ↓ · flag



Hristina Galabova Signature Track · 5 days ago 🔗

Thank you, this is helpful. I have follow-up questions related to manipulating the query:

- Is there a way to modify the term frequency of a term that is already stored in the query document? For example, as it stands, the training queries are created as documents and getting counts() of a query document produces an unordered map of the terms and their respective counts, which are 1 for all training queries. How can I manipulate this count - I would like to apply a parameter to modify the value.
- I am able to retrieve all terms within a relevant document and get their counts. How can I create a document object out of this information? I see that a document can be created from a text file but in this case I already have the data broken down by term & count. How can I place this in a document so I can further process it with functions that require a query? I do not see a constructor that takes unordered map?

Thank you!

↑ 0 ↓ · flag

[+ Comment](#)



Hristina Galabova Signature Track · 5 days ago 🔗

I answered my own question on the first bullet point - I can see how increment could be used to either increase or decrease the value.

The second question remains, though - is there an elegant way to create a document object from a map containing terms and their counts? I suppose I could loop through the term IDs, get their text, string them all together, create a doc, and then go back through the doc and assign the right weights to each term. But I am hoping there is a better way to do this...

↑ 0 ↓ · flag

Hussein Hazimeh STAFF · 5 days ago 🔗

There is no built-in function in the document class that can directly do that. One way is to loop over the <term_id, term_freq> pairs in the map and use the increment function in the newly created document to assign each term_id its corresponding term_freq. This is very similar to what is being done in the last for loop in the code shown above. So you can write something like:

```
// Assuming the variable term_freq contains a map of the form <term_id,term_freq>
corpus::document new_document;
for (auto& tfpair : term_freq)
    new_document.increment(idx->term_text(tfpair.first),tfpair.second);
```

↑ 0 ↓ · flag

Hussein Hazimeh STAFF · 5 days ago 🔗

Also, if you're planning to use the counts() function to read the terms and frequencies from the query, make sure to execute "idx->tokenize(query);" directly after setting the content of the query. Without tokenization, the counts() function will return an empty list.

↑ 0 ↓ · flag



Hristina Galabova Signature Track · 4 days ago 🔗

Hi Hussein,

I am not able to use query.increment to add terms that are not already in the query.

The method successfully increments the count for a term already in the query.

However, when I try to use it to add more terms to the query, the query is not adjusted.

I have tried the following:

- query.increment("test", 1); -- this did not work and I thought I would need to use a term that was already indexed so I tried
- query.increment("general", 1); I know general is one of the index terms after tokenization has been performed. This did not work either
- query.increment(idx->term_text(12604), 1); where 12604 is the ID of an existing term in the index

In all of the above cases I get no errors but when I try to print out the query by using query.content(); I still get the original query only.

What am I doing wrong?

Thank you!

↑ 0 ↓ · flag

[+ Comment](#)

Renaud Dufour · 5 days ago 🔗

Thanks a lot Hussein, implementating the feedback seems more doable now, will give it a try!

↑ 1 ↓ · flag

[+ Comment](#)

Renaud Dufour · 5 days ago 🔗

Just a quick question, I am able to display the query content using query.content(), however this does not work for the new_query when I try new_query.content(), in which case i get an error telling the document is empty.

Also I observed that when going through query.counts() in the first loop (add terms from original query

to new query), we add several terms like `</s>` and `<s>`, for example when I print out the terms added to the new query I have something like :

Is is normal / related to the fact that we cannot print `new_query.content()` ?

```
Ranking query 100: m1
Adding new term from original query : </s> with count 1
Adding new term from original query : m1 with count 1
Adding new term from original query : <s> with count 1
Adding new term from top doc : </s> with count 38
Adding new term from top doc : <s> with count 38
Adding new term from top doc : abid with count 1
Adding new term from top doc : abu with count 5
Adding new term from top doc : access with count 1
Adding new term from top doc : achiev with count 1
```

↑ 0 ↓ · flag

[+ Comment](#)



Lingyue Pan Signature Track · 4 days ago 🔒

Thank you for your post so that I could implement the Rocchio, but I got a MAP lower than the original MAP which I didnt use Rocchio. I have change the weight and the number of top documents, it seems that the datas not suit for using Rocchio.Is it normal or did I do something wrong?

↑ 0 ↓ · flag

Renaud Dufour · 4 days ago 🔒

I also implemented the feedback and tuned the weights ($\alpha = 1$, changing β) and didn't get any improvement. The MAP keeps increasing as $\beta \rightarrow 0$ and the best value corresponds to the ranking without feedback.

As you says the data may not suit for Rocchio, I observed that in many cases the words added to the query from the top documents are words like "learn" or "cours", which is not surprising considering this is a mooc dataset. So I would be curious to know if anyone succeeded in improving significantly its score with Rocchio...

↑ 0 ↓ · flag

Gayathri Sankaran Signature Track · 3 days ago 🔒

I got a lower MAP score with Rocchio as well. Is there a factor by which the term weights can be modified?

↑ 0 ↓ · flag

Hussein Hazimeh STAFF · 15 hours ago 🔒

Good observation Renaud! One way to deal with this issue is to add the frequent words that

are appearing in all the MOOCs to the stopwords list and index dataset again. Also, more aggressive IDF weighting should help in case common terms are still dominating the modified query.

↑ 0 ↓ · flag

[+ Comment](#)



James McCullough Signature Track · 4 days ago 🔗

This is awesome, thank you Hussein for providing this, and for providing the basis of the code we're using in competition.cpp. It is greatly appreciated!

↑ 0 ↓ · flag

[+ Comment](#)

Anonymous · 3 days ago 🔗

I am completely lost with this programming assignment. I am new to C++. For ex. I am unsure about the type of the ranked documents to be passed to the Rocchio function, when I passed "auto" it gave me a compilation error. I tried the code posted above and got top 10 documents for 50...Don't know what to do with it. Been reading the same thing again and again, but totally lost.

↑ 0 ↓ · flag

[+ Comment](#)

Renaud Dufour · 3 days ago 🔗

Hi, I understand that the assignment is tough when you have to learn C++ along the way. It is the same for me and I already spent >10h-15h to be able to implement different rankers, tune them and implement 'kind of' Rocchio feedback. It is the first time the course run so I understand it is not easy for the instructors to find the appropriate level of difficulty and a good tradeoff between giving too many advises and not enough.

regarding the types, the query is of type document, so you have to pass it using

`meta::corpus::document & myquery` (more info [here](#))

for the ranking, it is a vector of pair and you should use `std::vector<std::pair<long unsigned int,double>> & ranking` (this is what is returned by the `ranker::score()` function, see [here](#))

Also, note that from the posts above, it seems Rocchio feedback is not doing a wonderfull job (at least for me) so you can also focus on simpler tasks like tuning different rankers.

↑ 0 ↓ · flag

Cristi Dumitrescu · 2 days ago 🔗

Same here, Rocchio pseudo feedback turned out to be a complete waste of time. Spent >10h, tried several thousand combinations of number of terms taken, number of documents

considered, alpha and beta - none of them resulted in an improvement over what I already had. Also tried different ranking algos with it, also tried Kullback-Leibler in several forms, same results. My guess is that the dataset is too topic-focused on learning to get good results with pseudo feedback.

I'm really curious if anyone got an improvement using Rocchio pseudo feedback.

↑ 1 ↓ · flag

Alberto Carrasco Signature Track · a day ago

I am wondering the same, Cristi. When I apply some weight to the training documents within my Rocchio function, the MAP lowers dramatically (my best MAP after Rocchio is 0.2 less than the MAP without smoothing). Not sure if there is something I am doing wrong.

As you suggest below, it would be of help if someone who got a improvement using Rocchio could share their experience.

↑ 0 ↓ · flag

Cristi Dumitrescu · a day ago

0.2 is huge, I'm guessing you're doing something wrong. I sorted the terms list, limited the number of terms considered, made meta work with floating point query weights so I can use proper alpha and beta coefficients and the resulting MAP was lower than the non-Rocchio by somewhere between 0.001-0.03, depending on the number of top docs considered, number of top terms considered and alpha/beta settings. Like I said, I tried lots of combinations and none of them resulted in an improvement so Rocchio seems like a wild goose chase to me. Actually, I even got several results that were up to 0.01 higher than my original MAP but submitting them didn't improve my score in the score table so I suspect they were just overfitting the given query results dataset.

Speaking of the dataset, it does contain some good queries and evaluations, but it also contains a lot of poor choices and it looks like some people didn't even bother selecting relevant results. For instance, if you check the first few queries you will notice that query #1 "economics statistics" only has a single result marked as relevant, same as query #3 "chinese"... Both these queries have plenty of relevant documents in the database.

Finally, in my experience there's a large difference between the MAP on the test set and the MAP on the validation set. My test MAP of 0.66xx ended up below 0.63 when the result was uploaded. I can't help but wonder if there's maybe a setting that produces a worse MAP figure on the test set but does better on the validation set. I gotta say it feels pretty pointless to optimize on the test set when the difference is so large and many of the query/feedback pairs are so poor. Wish we had some cross-checking of each query available, at least on the test set.

↑ 1 ↓ · flag

 Hristina Galabova Signature Track · a day ago

Here is my experience so far. I decided to determine what (if any) improvement I would get on the testing data if I used the information we have on relevant documents instead of

pseudo relevancy. My thought was that this would provide me with the upper limit of the improvement I could get by using Rocchio with specific values of alpha and beta. By doing that I was able to go from a MAP of 0.6... to 0.7... on the testing set, which I think is quite an improvement considering the range of MAPs I was getting when I was playing with ranking parameters for BM25.

My next step is to see what results I would get with the same algorithm and the same alpha and beta values but this time using pseudo relevancy.

I am hoping to have results tonight or tomorrow and will share my experience.

↑ 0 ↓ · flag

Cristi Dumitrescu · a day ago

Thanks for sharing your experience, Hristina. However, as I'm sure you're aware, what you did is extreme overfitting - you're basically making the search engine very biased toward the results that are known to be good. I'm really curious what you'll get using pseudo feedback, though.

↑ 2 ↓ · flag

Renaud Dufour · a day ago

I have a similar difference between train/test MAP. This is not really surprising to me considering the datasets are not very big, we are just tuning our search engine considering 100 queries which, as you says, are not all very good. So we can easily end up overfitting the training set. I stopped working on the Rocchio feedback and focus more on tuning different rankers and see how robust they are, which I think is also an important aspect. For example for okapi_bm25, exporting and plotting the curves of MAP versus parameters (k1,b,k3), we see that a good performance can be achieved but only for a narrow range of values, i.e. there is a sharp maximum meaning that efficiency can quickly decrease if the training set is not representative of the test set. Some other ranker are more robust, i.e. changing a bit the parameters results in a MAP on the test set in the same range (MAP curves exhibit kind of a plateau).

We also have to be careful about over-fitting of the test set (since the number of submissions is not limited) and do not forget the final ranking will be done by adding the relevance judgement of assignment2/part1.

↑ 1 ↓ · flag



Hristina Galabova · 4 minutes ago

Signature Track

Well, after trying many variations of the number of docs to consider for pseudo relevancy, what words from those docs to pick, and various values of alpha and beta, I was also not able to achieve an improvement with Rocchio.

Playing with all options and doing the coding helped me better understand how Rocchio is applied and also got me to brush up on my C++ skills. So even though I spent way too much time on this, I consider it well worth it.

On to exploring other options.

↑ 0 ↓ · flag

[+ Comment](#)

Cristi Dumitrescu · 2 days ago 

Hussein, many thanks for the tips but I am really curious about something: given the fact that the exercise notes specifically suggest Rocchio and you actually posted these hints here, did the course staff try to implement Rocchio pseudo feedback in the context of this exercise? If so, was a better MAP result obtained when pseudo feedback was enabled?

↑ 1 ↓ · flag

Hussein Hazimeh STAFF · 15 hours ago 

We were able to get a small improvement in MAP after implementing Rocchio with some additional modifications to the stopword list and the IDF weighting of the ranking function.

↑ 0 ↓ · flag

[+ Comment](#)

Cristi Dumitrescu · a day ago 

Actually, I was able to achieve a small improvement after severely butchering Rocchio into something else of my own creation, based on the same idea. The big problem is that improvements on the test set that is provided are not always equivalent to improvements on the set tested leaderboard.

↑ 0 ↓ · flag

Renaud Dufour · a day ago 

well, considering your jump on the leaderboard it's not bad at all :) efforts are finally paying! I still have the best precision at 5 >_< (although I don't know how to interpret that...)

↑ 0 ↓ · flag

Cristi Dumitrescu · a day ago 

Actually the leaderboard jump was due to something else that I came up with. My sort-of-Rocchio on top of that only contributed some 0.003 or so.

↑ 0 ↓ · flag

John Roth · 15 hours ago 

Can you tell us a bit more about your method, Cristi D?

I could use some inspiration, because I am not getting it from the .02 increase in MAP over the defaults that all your good work has amounted to. Work is its own reward of course.

I wonder what our fine TA's could accomplish with this data set.

↑ 0 ↓ · flag

[+ Comment](#)

Hema Tanikella Signature Track · a day ago 

I am completely lost in this assignment. I feel like the level of pre-instruction is low on this assignment compared to the first especially since I am not very well-versed with C++. I am still going to try and give it a shot but I doubt I can complete it.

↑ 0 ↓ · flag

Cristi Dumitrescu · a day ago 

I completely agree, in my opinion C++ is a rather poor choice for the programming assignments given the fact that there are very few people that are proficient with it nowadays and the speed boost that it provides over other languages is completely unnecessary in the context of this exercise.

As for completing the assignment, that's very easy. You can simply configure and run any ranker in the config.toml file then submit your results. You definitely won't score in the top 10 but you will get the point for the submission. Basically all you have to do is compile it and run it.

↑ 0 ↓ · flag

Hema Tanikella Signature Track · 20 hours ago 





@Cristi- Thanks for your post. I was able to complete the scoring function in competition.cpp based on last week's assignment. I guess now that is up and running, I am going to try and tune the parameters at the very least and see if I get better results.

↑ 0 ↓ · flag

[+ Comment](#)

New post

To ensure a positive and productive discussion, please read our [forum posting policies](#) before posting.

B	<i>I</i>			 Link	<code><code></code>	 Pic	Math		Edit: Rich ▼	Preview
<div></div>										

☐ Make this post anonymous to other students

☒ Subscribe to this thread at the same time

Add post