CS458 - Assignment 4
See deadlines below

For our final assignment you will be implementing a few evaluation metrics and then examine the performance of our IR system and the impact of different parameter settings. The evaluation process is critical to any research project and this assignment will be good practice in preparation for your final projects.

As always, read through this **whole** document before starting.

Deadlines

This assignment is broken into two parts:

- implementing the evaluation metrics
- using those evaluation metrics to examine the system performance and writing up the results

To make sure that you make steady progress on the assignment (and save time to think about your final project write-up :), there will be two separate deadlines for this assignment:

> *Tuesday, November 6, by the beginning of class:* Submit your working code. Submit this as assignment 4a.
>
> *Friday, November 9, by 6pm:* Submit your writeup. Submit this as assignment 4b.

Even though I have broken the submission into two parts, I strongly encourage you to get the evaluation metrics implemented before Tuesday and start working on your experimentation and writeup.

Data

For ease of experimentation and availability, we're using the Cranfield test collection. The data set is small (1400 documents and 225 queries), but does have a full enumeration of all relevant documents for each query. For our current purposes, this will be sufficient, but I will look into obtaining some larger data sets for use with your final projects.

The data can be found at:

```
/home/dkauchak/PUBLIC/cs458/assign4/data/
```

In there you will find three files:

- `cranfield.document`: The list of documents for our corpus (see `search.data.CranfieldReader` below for reading the data).
- `cranfield.queries`: The list of queries we'll be evaluating our system with (see `search.evaluation.BasicQueryReader` below for reading the queries).
- `cranfield.query.relevance`: The relevance judgements for the queries. Each entry in the file contains three integers. The first number is the line number of the query. The second number is the document ID. The third number is a relevance score from 1 to 4, with 1 being the most relevant and 4 being the least. For our purposes we will only consider documents to be relevant or not relevant. *A relevant document is any document with a score of 1 or 2.* I have left the other data in there if you want to experiment further.

  For example, for the first query, the relevant documents are `184, 29, 31, 57, 378, 859, 875`.

Changes to the code base

I made a few changes to the code base to get things in better shape and also added some I/O functionality to get you started on this assignment. As before, you are welcome to use your existing code, but you should make sure there are not any existing bugs. In addition, you will also need to copy over a few of your existing files (though mostly just for I/O) to stay compliant with the current code base.

- Created a new subpackage `search.evaluation` for use in this assignment.
- Added functionality for reading the Cranfield data.
  - `search.evaluation.BasicQueryReader` can be used to read the provided query file.
  - `search.data.CranfieldReader` can be used to read the cranfield data set documents.
- Created a new subpackage `search.data` which contains all of the document reading classes.
  - Added the class `BasicDcoumentReader` which makes adding new readers easier. To create a new reader you just need to extend `BasicDocumentReader` and then override the `readNextDocText` method (see `search.data.TDTReader` for an example)

  - Added a `reset()` method to the `data.DocumentReader` interface. Calling reset should cause the reader to start over again at the beginning of the data set.

- Added functionality to support query processing using tokenizers and token normalization

  - Added a `Query` class which is used to represent a user query.
  - `Index.rankedQuery` now takes a query.
  - `Search` now takes a `Tokenizer` and `TokenProcessor`. The same tokenizer and token processor used with the document reader should be used with the search engine.

What to implement

The skeleton code for this project can be found at:

```
/home/dkauchak/PUBLIC/cs458/assign4/code/
```

You will be implementing four different evaluation metrics to play with on our data. The method skeletons are found in `search.evaluation.Evaluator`. For programming, this is technically all you'll need to handin. However, you will also need to run some experiments. Below are a suggested set of steps to implement.

1. Using `search.data.CranfieldReader`, figure out how to create a new index over the cranfield corpus.
2. Using `search.evaluator.BasicQueryReader`, figure out how to read in all of the queries.
3. You should now be able to issue all of the test queries to the corpus and make sure things are working right.
4. Look at the relevance file. Take a look at `search.evaluation.Evaluator` and see what is expected to be passed to the evaluation functions. Write a class or method to read in the relevance data.
5. Write evaluation methods.

Experimentation and Writeup

Now that we have some methods for evaluating our system, we can see how well it's doing and we can see what effect our different system variations have on performance. As I mentioned in the introduction, this is a common problem to encounter when doing research.

For your write-up, investigate the performance of your system. Your write-up should be one to two pages of *well organized* text that describes overall how the system performs and examines a variety of parameter settings. Include a table or two showing some of your results. Since the actual programming part of this assignment should be relatively light, I expect you to spend some quality time playing with the system and analyzing the results.

You should discuss some, but not necessarily all of the following: how does tokenization affect performance? How do the token normalization affect performance? How do the tf-idf parameters affect performance? What are the optimal settings for the parameters? How do the different evaluation metrics differ/perform? What types of parameters are the most important to examine? Are any results counter-intuitive?

Note, it's likely you'll have to write some extra code to automate the experimentation process. I strongly recommend doing this since you'll be trying many variations of different parameters.

All of the evaluation methods you implemented are query-level metrics. To get an overall score for a system, the easiest thing to do is just to average the scores over all of the queries. This isn't perfect, but will suffice for now.

As an aside, the data set we're playing with is already lowercased, so that feature will not have an impact and will not be explored.

When grading the write-up I will look for the following:

- Is it well organized?
- Do you provide sufficient data?
- Do you analyze the data provided?
- Did you tackle a number of the questions above?
- Is the writing clear, grammatically correct, etc.?
- Are your arguments compelling?
- Does it appear that you spent some time and effort playing with the data and thinking about the results?

Hints/Comments

- To debug your evaluation metrics, you can simply pass in data and make sure it's doing the right thing.
- Both precision and recall share a common functionality.
- The definition of rPrecision can be found in the book.

- Both rPrecision and MAP leverage the precision function.
- Plan on spending some time on the writeup since it does involve running a number of experiments.
- I'll say it again, automate your experimentation process!

What to turn in and how to turn it in

- By the first deadline, submit a "jar" file of your code, which should contain all classes required to get your code working, including the original files I provided, as well as any supporting classes you need. See the assignment 1 writeup for details on creating a jar file. Submit this as assignment number 4a.
- By the second deadline, submit your writeup with your experimentation results and analysis. Make sure that all group member's names are at the top of this file. Submit this as assignment number 4b.