CS 221 Information Retrieval

Winter 2012 Projects

Project 1 -- Search Engine over Wikipedia

Goal: Retrieve Wikipedia pages related to the query and order them according to relevance

UI: regular web-browser search engine interface (e.g Google, minus the ads)

Topic coverage: web crawling (a little), text processing, link analysis, feature extraction, indexing, ranking and scoring, evaluation

Additional topics: map-reduce

Main components: lucene and/or katta and Hadoop

Baseline: google over site:wikipedia.org

Milestones

Milestone	Expected outcomes	Evaluation
M1 2/2 Lucene Basics 25%	- Get the Lucene demo working for 10 or so random articles of Wikipedia. Required steps: - Download the latest English-Wikipedia dump: http://en.wikipedia.org/wiki/Wikipedia:Database_download - Download the latest Lucene: http://lucene.apache.org/java/docs/index.html Notes: - The Lucene demo works for plain text files. You need to change it so that it works for HTML files. - If you prefer to do your project in a language other than Java, find an implementation of Lucene for your favorite programming language.	Does it work as intended? Do you understand what you did? You will show your demo to the Reader in a f2f meeting.
M2 2/16 Evaluation 25%	- Develop a test set of ~900 articles (see Additional Notes) and compute the NDCG@5 and/or MAP for a naïve indexing of those articles. By naïve I mean, an indexing that simply does what you did for the M1 demo. - Estimate the time for indexing the entire 4.8M articles of Wikipedia in your machine	Do you have the test set? Do you have the first results of NDCG@5 and/or MAP for that set? (both are better than just one) Do you have a believable estimate of the time it will take to index the entire Wikipedia?

		You will show your demo to the Reader in a f2f meeting.
M3 3/1 Lucene Advanced + Ranking 25%	- Improve the performance of your search engine by adding better features to the index (e.g. more structural information)	Were you able to improve the NDCG@5 and/or MAP metrics? You will show your demo to the Reader in a f2f meeting.
M4 3/15 The whole package 25%	Develop the complete search engine, including the web UI, and index at least 48,000 articles (1% of Wikipedia), but more is better. Report your engine's best NDCG@5 and MAP for the test set.	Does it work as expected? How many articles were you able to index? (more is better) What improvement did you get on NDCG@5 and/or MAP from M2 until final delivery? (more improvement is better) You will show your demo to the Reader and, maybe, the entire class.

Additional Notes:

For Milestone 2 you are asked to develop a test set of about 900 articles and compute performance metrics. Here are instructions for how to do that.

Step 1: Collect the dataset

Issue queries to Google (recommended you do this programmatically), and collect the 30 top results for each query (i.e. their URLs). There are 30 queries (see list below). As such, you will end up with roughly 30x30=900 Wikipedia articles. This will be your test set; it will be used to compare how your search engine performs wrt Google. If you happen to get a Wikipedia page that is not in the dump you downloaded (e.g. Category or User pages that start with Category:/User:), you should discard it from the results.

List of queries -- note: you should add "site:en.wikipedia.org" to all of these queries

DNA
Apple
Epigenetics
Hollywood
Maya
Microsoft
Precision
Tuscany
99 balloons
Computer Programming
Financial meltdown
Justin Timberlake
Least Squares
Mars robots
Page six
Roman Empire
Solar energy
Statistical Significance
Steve Jobs
The Maya
Triple Cross
US Constitution
Eye of Horus
Madam I'm Adam
Mean Average Precision
Physics Nobel Prizes
Read the manual
Spanish Civil War
Do geese see god
Much ado about nothing

Step 2: Calculating performance of your search engine

MAP

From the universe of 900 articles in the test set, and for each query, we will consider the top 10 results from Google as "relevant" and all the others "irrelevant." The average precision should be computed over the top 10 positions.

NDCG

For computing the NDGC we are going to use Google's results as the ground truth. There will be 6 levels of relevance: Google's first result has rel=6, and the 5^{th} result has rel=2; all results between the 6^{th} and the 10^{th} are rel=1; all other results are considered irrelevant (rel=0).

Extra Credit (15%)

Can be done in milestones 3 or 4. Its value is not just the extra 15%; you may want to put Hadoop in your resume, but this is a lot of extra work! I will be impressed!

- Extract the graph of the articles using Hadoop
- Compute PageRank for every article in Wikipedia. Well, ok, that is a very hard problem to do in map-reduce style, so maybe you should just compute the fan-in (and fan-out, since you're at it).
- Once you master the Hadoop emulator, you can have access to a live Hadoop cluster on Amazon and run the PageRank computation there.

The goal here is to be able to use network metrics as additional piece of information associated with the documents, so you can use it to boost them accordingly during indexing.

Required steps for the extra credit:

- Install the Hadoop emulator
- Use the following tutorial or frameworks for your reference
  (i) http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
  (ii) http://hadoop.apache.org/common/docs/r0.17.1/mapred_tutorial.html
  (iii) http://www.ics.uci.edu/~yganjisa/files/2011/hadoop-presentation/WikiNgrams-Skeleton.zip

Once you master the emulator, please contact Prof. Lopes for a live AWS cluster

Project 2 -- Sentiment Analysis for Twitter

Goal: find out how the Twitterverse is feeling about certain keywords

UI: http://twittersentiment.appspot.com/ or something similar

Topic coverage: text processing, feature extraction, classification, indexing, scoring, evaluation

Main components: weka, mallet, etc.

Evaluation: Your system will be tested twice on unseen test data during phase M2 and M4. You will be notified of your classifier's performance on M2, so that you can use the feedback to improve the performance.

Milestone	Expected outcomes	Evaluation
M1 2/2 Basics + Evaluation 25%	- Get the demo working and measure its classification performance Required steps: - Download the sentiment analysis demo from: http://www.sananalytics.com/lab/twitter-sentiment/ - Download and install all the necessary Python libraries that it requires - Download 5500(-ish) hand-classified tweets mentioned in the above tool from http://www.ics.uci.edu/~lopes/teaching/cs221W12/projects/twitter-sanders-data.zip. This can be used as training data for your classifier/model Notes: - Evaluation will be done using standard Precision, Recall and F-Measure metric values. Use cross-validation technique (http://en.wikipedia.org/wiki/Cross-validation_(statistics)) on the above data set to separate training and test set.	Does it work as intended? Were you able to compute the performance? Do you understand what you did? You will show your demo to the Reader in an f2f meeting.
M2 2/16 Classification Advanced 25%	- Improve the performance of classification by experimenting with different classifiers and/or adding better features - You can also try improving training of classifier by (i) creating more training data manually classifying new tweets (ii) using movie reviews data set (IMDB and rotten tomatoes) available at (http://alias-i.com/lingpipe/demos/tutorial/sentiment/read-me.html) - Your classifier will be evaluated on part of unseen test data (part of complete test set). You will be notified of its accuracy on the data. This will give you an indication of your classifier's performance. Notes: - Features are important part of classification, use contextual features (tweet location, time, etc.) along with textual features to improve the performance. More on contextual features:	Do you have a good feature set? Do you have a good training and test data set? What classifiers are you using? Were you able to improve performance? You will show your results to the Reader in an f2f meeting.

	(http://www.ics.uci.edu/~hsajnani/Publications/AAAI2011.pdf) - If you really don't like Python, you are welcome to re-implement everything in your favorite programming language, assuming you can find all the right libraries in that language.	
M3 3/1 More features + Twitter 25%	- Use WordNet (http://wordnet.princeton.edu/) and/or sentiment dictionary (http://www.ics.uci.edu/~lopes/teaching/cs221W12/projects/sentiment-dictionary.zip) to improve classification - Talk to Twitter: get a live feed from Twitter using the Twitter API Notes: -Sentiment dictionary is a labeled dictionary in which words are labeled between 5 to -5 expressing their positive or negative sentiment -The goal with WordNet and Sentiment dictionary is to be able to broaden the vocabulary beyond the words present in the training set so that classification will be further improved. -It is important to keep in mind that dictionary approach may not always boost performance, as lot of it depends on your training and testing data. Usually hybrid approach outperforms the rest. - http://mytwitmood.tutafuta.com/ is an example web app to get overall mood from last 20 tweets of a particular user. Its algorithm is not only based on single word adjectives but also based on combination of words	Were you able to use WordNet and/or sentiment dictionary successfully? Did improve classification? If so, why? If not, why not? Were you able to get a live feed from Twitter using the API? You will show your results to the Reader in a f2f meeting.
M4 3/15 The whole package 25%	Develop the complete tool, including the web UI, and report its best performance for the test set. Notes: -Your classifier will be evaluated on the complete unseen test data (which includes test set of M2). -http://smm.streamcrab.com/ has nice analysis and visualization of output	Does it work as expected? What improvement did you get from M2 until final delivery? (more improvement is better) You will show your demo to the Reader and, maybe, the entire class.

Extra Credit (18%)

Can be done in milestones 3 or 4. Its value is not just the extra 18%; you may want to put Hadoop in your resume, but this is a lot of extra work! I will be impressed! – even more so than with Project 1, because

for this project I am not even going to suggest anything for using Hadoop. If you think you need large-scale parallel processing to improve classification, talk to me. Hence the extra-extra 3%.

Project 3 -- Chemistry Community Analysis

Goal: some people believe that Chemistry is a much more closed field than all other scientific fields. Find out whether this is true or false. "Closeness" can be measured by a number of metrics, for example: (1) average number of authors per paper; (2) average number of affiliations per paper; (3) ratio of Institutions/Authors of the entire field; (4) citations to other papers in other fields. Additional metrics can/should be devised as the project develops.

Warning: don't be fooled by the apparent simplicity of the Milestones of this project, when compared to the other 2 projects. The absence of detailed instructions is a symptom that this is a much more open-ended project, it hasn't been done before, and, as such, it is much harder to do [well] than the others. I expect *you* to drive the project. If you usually don't do well with severely underspecified projects, don't choose this one, do one of the other two. If you usually do well figuring things out on your own, then it's ok to choose this project. But you have been warned – don't expect me to tell you what to do in detail; if you need that kind of guidance, do one of the other projects.

UI: none, not interactive

Topic coverage: web crawling, text processing, feature extraction, noise reduction, network analysis

Main components: up to you, but talk to me

Prize: if your project is well done, and the conjecture turns out to be true, I will work on a paper with you.

Milestone	Expected outcomes	Evaluation
M1 2/2 Find the sources 25%	- Identify one or more sites from where you can get scientific papers that contain authors, their affiliations and whether they are Chemistry or not. Possibilities: PubMed, PNAS. - Study the available data in-depth and make a plan for crawling it	Do you have a good grasp of the data you are about to crawl? You will have a meeting with Prof. Lopes.
M2 2/16 Crawling 25%	Get the data Required steps: write crawlers for those sites and collect the data locally in a way that is easy to parse/analyze	Were you able to get a very large collection of data? Is the local data easy to analyze? You will have a meeting with Prof. Lopes.
M3 3/1 Parse the	Extract authors, affiliations and, ideally, citations; produce an Excel spreadsheet (CSV) or any other similar artifact that gives a global picture of the data we want to study and that can serve to make all sorts of data analyses.	How good is your artifact with raw data?

data 25%		You will have a meeting with Prof. Lopes.
M4 3/15 The whole package 25%	Prove or disproof the conjecture beyond a shadow of a doubt.	Does your work provide convincing evidence either way? You will have a meeting with Prof. Lopes and, maybe, show your project to the entire class.