Assignment 5

Fall 2016
CS834 Introduction to Information Retrieval
Dr. Michael Nelson

Mathew Chaney

December 16, 2016

Contents

List of Figures

List of Tables

1 Question 10.5

1.1 Question

Find a community-based question answering site on the Web and ask two questions, one that is low-quality and one that is high-quality. Describe the answer quality of each question.

1.2 Results and Discussion

Yahoo! Answers is the question and answering website I chose to use to complete this exercise. First, the high quality question I asked was:

"Were there treaties in place at that time of Russia's annexation of Crimea which were, without doubt, violated by this act?"

I deem this question to be of high quality based on grammaticality, spelling and punctuation, as well as the use of concise terminology that may be slightly obscure to an uneducated reader.

It is also a focused question that has a definite set of possible answers. Either treaties exist that were broken or there weren't, albeit with some amount of flexibility due to legal interpretation.

An example of a good response I received within minutes of posting the question is this:

> Absolutely, the Treaty of the Separation of States between Ukraine and Russia (c.1994) gave all Crimea to Ukraine without doubt. By said treaty Russia got to keep it's historic naval base in Crimea... But not an inch more. However, even in the mid 90's Crimea's population was 98 percent Russian, and Russia did have historic ties there from 1700 on.

This answer is well thought out and readable. It references an item that directly answers the question: a treaty that between the Russian Federation, the United States of America, and the United Kingdom concerning the territorial integrity and political independence of Ukraine (among other nations) in exchange for their nuclear disarmament. The answer also provides some reasoning as to how it could be a justified act in the eyes of the Russian Federation, which adds to the quality of the answer by providing insight from multiple perspectives.

Now, for the poor quality question, I asked:

"Trump is not russia, y u no belief he got hakerz to do it to us?"

Firstly, it begins with a premise that doesn't make semantic sense; I doubt there are many people who would suspect that Trump *is* Russia. It also lacks focus as to what is being asked for and has no real context and is full of misspellings and terrible grammar. For these reasons this is a bad question.

Correspondingly, the answers were lacking in quality based on content, poor grammar and spelling errors. Some could even be considered offensive. Refer to Table 1 for some sample answers.

Trump is Putin's *****
Ye
we got hakerzed in the ***
no
Penis

Table 1: Responses to "Trump is not russia, y u no belief he got hakerz to do it to us?"

So there does seem to be a correlation between the quality of the question and the quality of the answers one can expect to receive. Higher quality questions tend to receive higher quality answers.

2 Question 10.6

2.1 Question

Find two examples of document filtering systems on the Web. How do they build a profile for your information need? Is the system static or adaptive?

2.2 Answer

My first example is Amazon (www.amazon.com). This is an Internet marketplace where nearly any type of product can be purchased online and delivered to one's home. The filtering the system performs is done as a recommendation of items it believes a user would be interested in purchasing. It presents accurate recommendations by using historic data of items purchased together and calculating a correlation factor for pairs of items. Using this correlation factor it can recommend items that are often purchased together when a user adds one of the pair to their online shopping cart. For example, if one is shopping for mechanical pencils, it is likely that the user will also be interested in purchasing graphite refills or replacement erasers, because these items are often purchased together. This system must be dynamic because it has to build correlations on the fly as item availability changes day to day and new correlations need to be created and updated to suit a dynamic market.

My second example is Netflix (www.netflix.com). This is an online video streaming service that famously posted an open challenge for anyone that could provide a better recommendation system than their existing system. The data used for the challenge was over 100 million ratings from 400 thousand users regarding 17 thousand movies. Each data element was a quadruplet matching the form:

```
<user, movie, date of grade, grade>
```

The user and movie fields are simple unique identifiers, the date field could be any date type and the grade field is an integer from 1 to 5. The goal of the recommender system is to predict the rating a user *would* give to a movie they haven't yet watched and recommend movies the user hasn't yet seen that it believes the user would rate highly.

It accomplishes accurate recommendations by using the movie grading records of its users. First, it determines which users are similar to each other by computing a similarity measure between each pair of users based on the grades they have assigned to movies they have both watched. If the pair of users have given similar grades for more movies their similarity measure will be higher. The system then uses this measure to determine a ranking for each user of all other users ordered by their similarity measure. Once this is done, the system can determine the most likely grade a user will assign to a movie they haven't seen based on grades similar users have assigned to that movie. Users update their movie grades as they watch movies, so the data is constantly changing, which makes this a dynamic system as it needs to update movie preferences of users as new data becomes available to the system.

3 Question 11.2

3.1 Question

Does your favorite web search engine use a bag of words representation? How can you tell whether it does or doesn't?

3.2 Answer

My favorite search engine is Google. I am fairly certain they do not use the bag-of-words model described by the text book.

As an example, there is a well known online meme, or trend, called "y u no guy". It is an image macro used to bring attention to something and uses a popular practice from online forums of shortening words to a single letter corresponding to the phonetic pronunciation of the word, e.g. "you" becomes "u". The target message is accompanied by a cartoon man with a face of frustration and rage. See Figure 1 for the results of issuing the query "y u no" to Google's search engine.

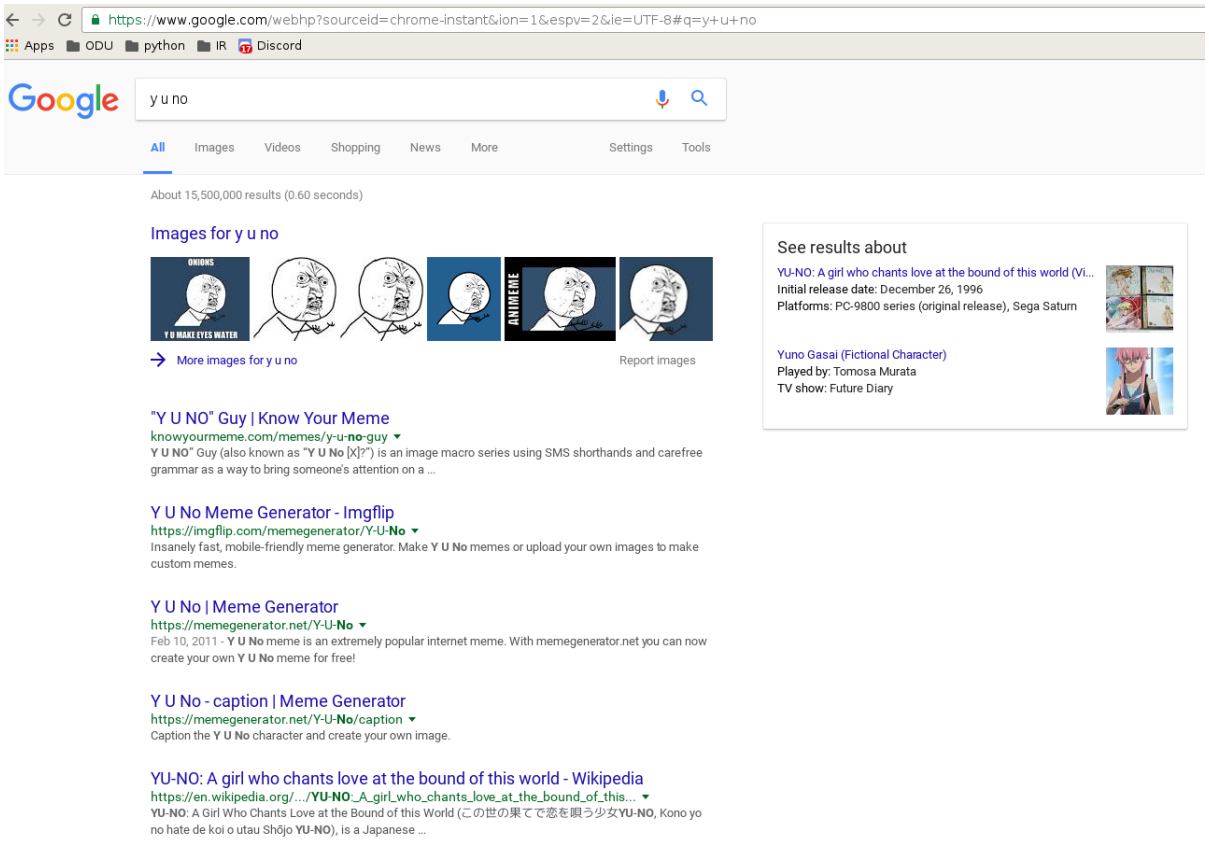

Figure 1: Google results from issuing the query "y u no"

Google recognizes these three simple terms to be part of a phrase and retrieves results based on that phrase with a high degree of accuracy.

To show that Google's engine does not use the bag-of-words model, consider this similar search with the terms "no u y".

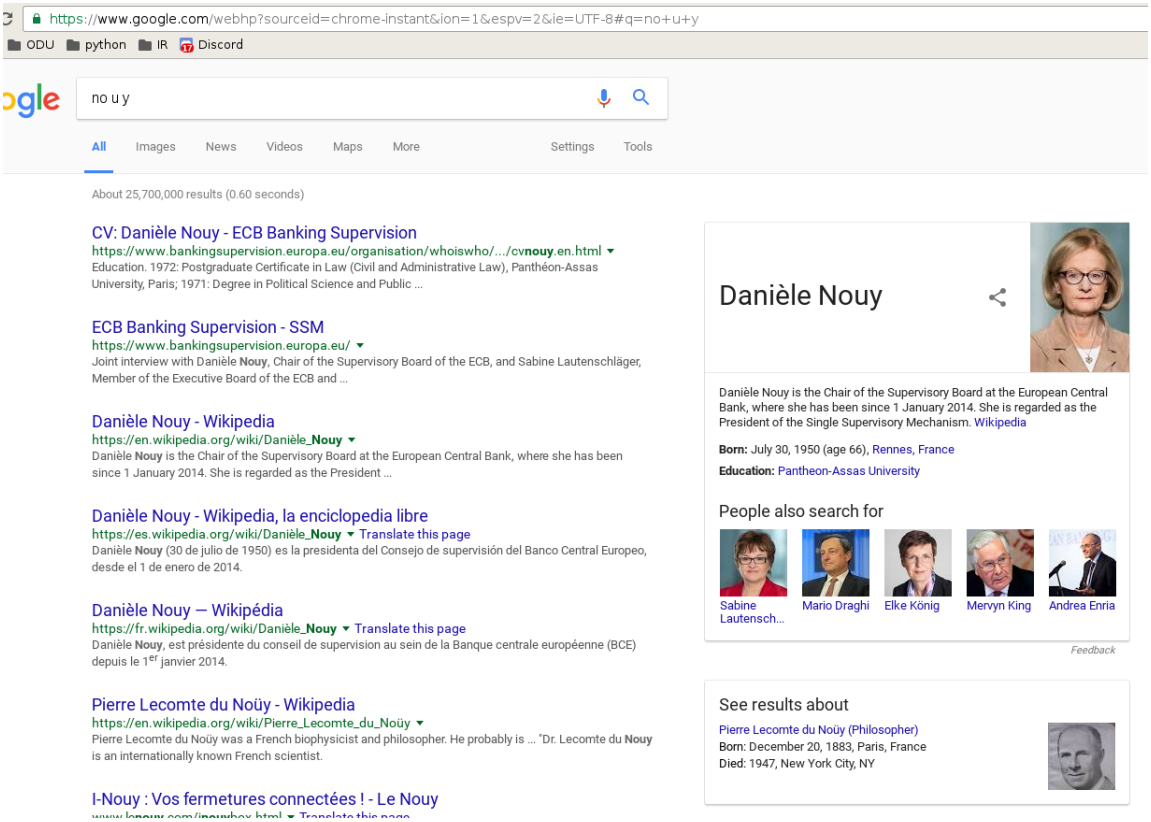

Figure 2: Google results from issuing the query "no u y"

With the same three terms in a different order, a completely different ranking is presented by the search engine. The results for this search are not regarding y u no guy at all, they are dominated by pages about the Chair of the Supervisory Board at the European Central Bank, Daniele Nouy. This shows that Google's engine accounts for query term ordering, which would not be present in a plain bag-of-words retrieval model.

4 Question 11.4

4.1 Question

Show how the linear feature-based ranking function is related to the abstract ranking model from Chapter 5.

4.2 Answer

Starting with the first mention of the abstract ranking model in the textbook, refer to Figure 3. This is an example of the abstract ranking model for a single document.

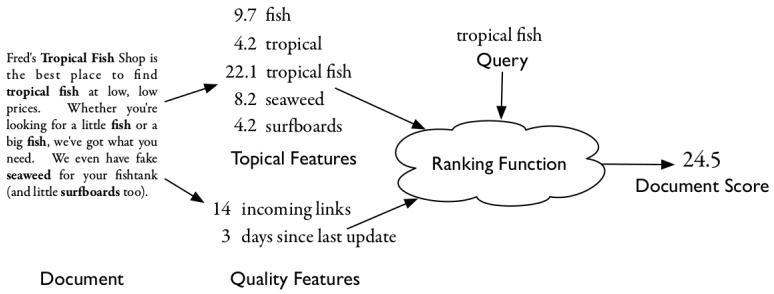

Figure 3: The components of the abstract model of ranking...

A more formal definition can be found in equation 1:

$$R(Q, D) = \sum_i g_i(Q) f_i(D) \tag{1}$$

This is a linear combination of two feature functions: f_i extracts a score from the document and g_i extracts a score from the query.

Now, for a definition of the linear feature-based retrieval model, refer to Equation 2:

$$S_\Lambda(D; Q) = \sum_j \lambda_j \cdot f_j(D, Q) + Z \tag{2}$$

here, f_j is a feature function that extracts a score from query/document pairs, $\lambda_j \in \Lambda$ are parameters, and Z is a constant. This is also a linear combination of functions that emit scores, which is similar to how the abstract ranking model works, with the addition of the parameters (Λ) and constant (Z).

5 SVM Light Example

5.1 Exercise

Work through the "Inductive SVM" example, discuss in detail the steps and resulting output.

5.2 Inductive Example

SVM Light [1] and the Inductive example were downloaded from http://www.cs.cornell.edu/People/tj/svm_light/.

Following the instructions in the Inductive example, the `svm_learn` program was invoked on the training data file train.dat, the output from which can be found in Listing 1. This produced the model file that is used in the classification step.

```
[mchaney@mchaney-l svmlight]$ ./svm_learn train.dat model
Scanning examples...done
Reading examples into memory...100..200..300..400..500..600..700..800..
900..1000..1100..1200..1300..1400..1500..1600..1700..1800..1900..2000..
OK. (2000 examples read)
Setting default regularization parameter C=1.0000
Optimizing.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
........done. (425 iterations)
Optimization finished (5 misclassified, maxdiff=0.00085).
Runtime in cpu-seconds: 0.07
Number of SV: 878 (including 117 at upper bound)
L1 loss: loss=35.67674
Norm of weight vector: |w|=19.55576
Norm of longest example vector: |x|=1.00000
Estimated VCdim of classifier: VCdim<=383.42791
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.00
XiAlpha-estimate of the error: error<=5.85% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>95.40% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>93.07% (rho=1.00,depth=0)
Number of kernel evaluations: 45954
Writing model file...done
```

Listing 1: output of the svm_learn program using the train.dat data file

After the model file was generated the example directs the user to run the `svm_classify` program on the provided test data `(filename: test.dat)` using the training model `(filename: model)` from the previous step.

5.3 Results

As the results show in Listing 2, the precision was 96.67% and the recall was 99.00%, which are very high scores for these measures.

```
[mchaney@mchaney-l svmlight]$ ./svm_classify test.dat model predictions
Reading model...OK. (878 support vectors read)
Classifying test examples..100..200..300..400..500..600..done
Runtime (without IO) in cpu-seconds: 0.00
Accuracy on test set: 97.67% (586 correct, 14 incorrect, 600 total)
Precision/recall on test set: 96.43%/99.00%
```

Listing 2: output of the svm_classify program using the model training data file

6 References

[1] Thorsten Joachims. Svm light. Available at: http://www.cs.cornell.edu/People/tj/svm_light/. Accessed: 2016/12/13.