HW2

Recommendations

Overview

For this part I used two datasets from Yahoo Labs Ratings and Classification data. The first was a user rating system, where users rated songs they listened to. The second was a click tracker which checked which pages users visited. For each, I ran a recommender with a few configurations for the various options:

- simple recommender (user based, k nearest)
- user based, item based, als
- k-nearest and threshold
- Pearson correlation, cosine measure, Euclidean distance, Spearman (on a smaller data set), Tanimoto, log-likelihood
- Slope one

Results

Songs

All of these were tested by training on 80% of the data and evaluating on 20% of the data (except ones involving spearman). I report average absolute difference, precision, and recall. A discussion of the results follows below.

Configuration 1

user based, k nearest, pearson correlation

AAD: 1.343

P:.82

R:.85

Configuration 2

item based, k nearest, pearson correlation

AAD:1.105

P:.85

R:.86

Configuration 3

als, k nearest, pearson correlation

AAD:.95

P:.89

R:.94

Other Configurations

- Config 1 with cosine
- Config 1 with euclidean distance
- Config 1 with spearman
- Config 1 with tanimoto
- Config 1 with log likelihood
- Config 1 with slope-one

Clicks

Configuration 1

user based, k nearest, pearson correlation

AAD:.95

P:.87

R:.76

Configuration 2

item based, k nearest, pearson correlation

AAD:1.02

P:.84

R:.86

Configuration 3

als, k nearest, pearson correlation

AAD:.85

P:.92

R:.88

Comments and Observations

User vs Item vs ALS

The item recommender ran slightly faster for music, which make sense given that there were more users than items. It ran significantly faster for clicks, which also makes sense given that there were more users than items. In terms of performance, user based performed the worst for songs, but still wasn't too bad in terms of precision in recall. ALS performed the best, and slightly behind it was user based. It was interesting that for clicks, the performance of user-based and item-based was flipped. However, these data sets are very different. The music database is different than the click database. It has more users and more items, and the ratio of items to users is much higher.

Pearson vs cosine vs euclidean distance vs spearman vs tanimoto vs log-likelihood

These were tested for the song data, and in general it was difficult to draw conclusions between them. They all produced results in the 1.026 to 1.432 range, with Pearson and cosine performing in the middle (1.343 and 1.325), euclidean distance performing slightly better (1.026), and tanimoto (1.432) and log-likelihood (1.394) performing slightly worse. These differences are pretty small, so its tough to say one does significantly better than the other. They also vary based on how we partition the data 80%/20%, so they vary each time the algorithm is run. The one outlier is the Spearman, which performed about the same, but took a significantly longer run time despite running on less than half the data.

SlopeOne vs Generic

Not only was the Slope one recommender faster, it performed generally better than the Generic Recommender. While most of the configurations with the generic recommender for the song data had evaluations of 1.0+, with the slopeone recommender I had evalutions that were an average of .2 points better, as well as improved precision and recall.

Nearest neighbor vs threshold

I compared the fixed n nearest neighbors to a threshold neighborhood. Threshold could be tweaked to perform better than n nearest neighbor's best performing parameters.

Clustering

Overview

Here we cluster some sets of data. The first are a number of data items from the Reuters set provided. Next are wikipedia articles, and finally some tweets. For this part, I tested: - K-means - canopy - fuzzy k means

The reuters data was provided, the wikipedia data I found online along with a script to parse it with mahout. The sanitized tweet data was taken from COMS1006. For each of these, I ran clustering algorithms following the *Mahout in Action* book. I did a little bit of optimization by hand to get better performance, but not too much. I just wanted to get a feel for how the parameters influence performance. I provide a number of interesting clusters, as well as an analysis of the performance below.

Results

Reuters

Here I tried to see if I could find clusters similar to ones in the example book to ensure that the algorithm was working correctly. I then compared these across clustering algorithms

From book

Id: 11736: Top Terms: debt, banks, brazil, bank, billion, he, payments, billion dlrs, interest, foreign

Id: 3475: Top Terms: iranian, iran, iraq, iraqi, news agency, agency, news, gulf, war, offensive

Id: 20861: Top Terms: crude, barrel, oil, postings, crude oil, 50 cts, effective, raises, bbl, cts

K-means

Some clusters:

Id: 14379: Top Terms: currency, interest, inflation, debt, money, bank, dollar, market, stock , equity,

Id: 11798: Top Terms: war, iran, invade, iranian, iraqi, blockade, bomb, bombing, casualties, killed

Id:17451: Top Terms: oil, crude oil, barrel, cts, tanker, gallon, rose, bbl, brent, refinery

Canopy

Some clusters:

Id: 13485: Top Terms: iran, war, iranian, invasion, fighting, iraq, iraqi, forces, troops, killed

Id: 18947: Top Terms: price, trade, decline, barrel, market, stock, exchange, oil, crude oil, dollar

Fuzzy K means

Some clusters:

Id: 16427: Top Terms: money, dollar, price, currency, trade, inflation, interest, exchange, equity, stock

Id: 10438: Top Terms: iraq, iran, iranian, iraqi, war, oil, price, bomb, tanker, fighting

Id:19341: Top Terms: oil, barrel, crude, crude oil, trade, tanker, war, price, dollar, cts

Wikipedia

This was run on a large data dump I found online. I used scripts and modified the code for the reuters data to parse the wikipedia entries in the same way. The results are siginificantly less interesting, I have included some clusters below.

K-means

Some clusters:

Id: 36728: Top Terms: Egypt, Egyptian, hieroglyphics, pharaoh, pyramid, Nile, ra, Achaemenids, Ramesses, Nile River

Id: 53646: Top Terms: Mesozoic, mammals, Pangaea, Cenozoic, Jurassic, paleontologist, million years, dinosaurs, extinction event, Cretaceous

Tweets

This part was just for fun, I ran K-means on several million tweets. Here are some weird clusters.

Id: 19483:

Top Terms: Iraq, Iraqi, isis, caliph, islamic state, jihadist, jihadi, islamic, extremist, caliphate,

Id: 16946: Top Terms:Giants, world series, series, baseball, mlb, royals, win, Kansas City, inning, pitch

Id: 21614: Top Terms:ebola, West Africa, CDC, scare, die, vomitting, quarantine, epidemic, Liberia, Houston

Comments and Observations

Canopy clustering had a much faster runtime than k means, however some clusters seemed to be almost random in nature. This is apparently the expected behavior. Some clusters had a lot of common words, like "the" or "it" or pronouns like "he" "his" "her" "it". Other clusters had words that were far to specific to be in any significant number of articles, like "foreign equity" and "speculative investment". K Means had a lot of parameters to optimize for, which was difficult because running through the algorithms took on the order of minutes to run, so it was not as simple as for the recommendations. Fuzzy K means was the most interesting to me because you got to see the overlapping between different topic areas. For example, the oil prices for the Reuters data were related to the Iran-Iraq war going on, which involved targeting of oil tankers. When you just run k-means, the common words are split into one cluster or another, while fuzzy k means not only only shows you related words for a single cluster, it shows you how that cluster might be related to a separate cluster via shared words. This is interesting from an analysis perspective. If we look at a word like "fear" and use twitter data, we may see that it pops up under the "ISIS" cluster, the "Ebola" cluster, and probably many other clusters like "Recession", "Unemployment rate", etc. thus we can gain insight into what it is that people are afraid of. We can do the same thing with "happy" or "want", in order to gain some insight into how different topics are related.