Old Dominion University
Department of Computer Science
CS834: Introduction to Information Retrieval
Fall 2017
Assignment 5
Professor: Dr. Michael Nelson

Hussam Hallak
CS Master's Student

December 15, 2017

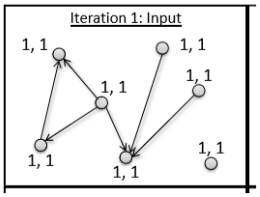
Question 1:

Exercise 10.3: Compute five iterations of HITS (see Algorithm 3) and PageRank (see Figure 4.11) on the graph in Figure 10.3. Discuss how the PageRank scores compare to the hub and authority scores produced by HITS

Answer:

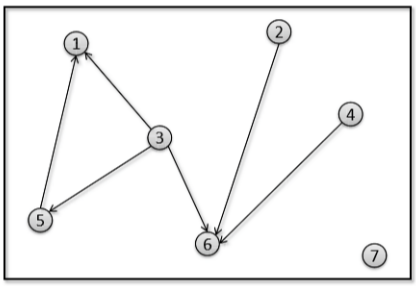
The graph below was copied from the textbook Figure 10.3.

Figure 1: Copied from the textbook Figure 10.3



I used the same node numbers in the textbook. The graph below was copied from the textbook Figure 10.4.

Figure 2: Copied from the textbook Figure 10.4



I wrote a python script and named it hp.py to calculate HITS and PageRank scores using Link Analysis.

```
1  import sys
2  import os
3  import networkx as nx
4
5  def pagerank(G, alpha=0.85, personalization=None, iterations_count=11, nstart=None, weight='weight',
       dangling=None):
6      if len(G) == 0:
7          return {}
8
9      if not G.is_directed():
10         D = G.to_directed()
11     else:
12         D = G
13     W = nx.stochastic_graph(D, weight=weight)
14     N = W.number_of_nodes()
15     if nstart is None:
16         x = dict.fromkeys(W, 1.0 / N)
17     else:
18         s = float(sum(nstart.values()))
19         x = dict((k, v / s) for k, v in nstart.items())
20
21     if personalization is None:
22         p = dict.fromkeys(W, 1.0 / N)
```

```
    else:
        missing = set(G) - set(personalization)
        if missing:
            raise nx.NetworkXError('Personalization dictionary '
                                   'must have a value for every node. '
                                   'Missing nodes %s' % missing)
        s = float(sum(personalization.values()))
        p = dict((k, v / s) for k, v in personalization.items())

    if dangling is None:
        dangling_weights = p
    else:
        missing = set(G) - set(dangling)
        if missing:
            raise nx.NetworkXError('Dangling node dictionary '
                                   'must have a value for every node. '
                                   'Missing nodes %s' % missing)
        s = float(sum(dangling.values()))
        dangling_weights = dict((k, v/s) for k, v in dangling.items())
    dangling_nodes = [n for n in W if W.out_degree(n, weight=weight) == 0.0]

    for _ in range(iterations_count):
        xlast = x
        x = dict.fromkeys(xlast.keys(), 0)
        danglesum = alpha * sum(xlast[n] for n in dangling_nodes)
        for n in x:
            for nbr in W[n]:
                x[nbr] += alpha * xlast[n] * W[n][nbr][weight]
            x[n] += danglesum * dangling_weights[n] + (1.0 - alpha) * p[n]

    return x

def hits(G, iterations_count=11, nstart=None, normalized=True):
    if type(G) == nx.MultiGraph or type(G) == nx.MultiDiGraph:
        raise Exception("hits() not defined for graphs with multiedges.")
    if len(G) == 0:
        return {},{}
    if nstart is None:
        h=dict.fromkeys(G,1.0/G.number_of_nodes())
    else:
        h=nstart
        s=1.0/sum(h.values())
        for k in h:
            h[k]*=s
    i=0
    while True:
        if i >= iterations_count: break

        hlast=h
        h=dict.fromkeys(hlast.keys(),0)
        a=dict.fromkeys(hlast.keys(),0)
        for n in h:
            for nbr in G[n]:
                a[nbr]+=hlast[n]*G[n][nbr].get('weight',1)
        for n in h:
            for nbr in G[n]:
                h[n]+=a[nbr]*G[n][nbr].get('weight',1)
        s=1.0/max(h.values())
        for n in h: h[n]*=s
        s=1.0/max(a.values())
        for n in a: a[n]*=s

        i+=1
    if normalized:
        s = 1.0/sum(a.values())
        for n in a:
            a[n] *= s
        s = 1.0/sum(h.values())
        for n in h:
            h[n] *= s
```

```
93      return h,a
94
95  if __name__ == '__main__':
96      iterations_count = 5
97      G = nx.Graph()
98      G.add_nodes_from(range(1,8))
99      G.add_edges_from([(1,2), (3,1), (3,2), (3,4), (5,4), (6,4)])
100
101     pr = pagerank(G, iterations_count=iterations_count)
102
103     print 'Pagerank Algorithm ({} iterations)'.format(iterations_count)
104     print '=============='
105     print 'Pagerank values = {}'.format(pr)
106
107     h, a = hits(G, iterations_count=iterations_count)
108
109     print 'HITS Algorithm ({} iterations)'.format(iterations_count)
110     print '=============='
111     print 'Hubs values = {}'.format(h)
112     print 'Authorities values = {}'.format(a)
113     print ''
```

Listing 1: The content of hp.py

```
1  hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/A5/Q1$ python hp.py
2  Pagerank Algorithm (5 iterations)
3  ==============
4  Pagerank values = {1: 0.15415210590313183, 2: 0.15415210590313183, 3: 0.2151836555953412, 4:
       0.2730700543561289, 5: 0.08952435340504106, 6: 0.08952435340504106, 7: 0.024393371432183873}
5  HITS Algorithm (5 iterations)
6  ==============
7  Hubs values = {1: 0.19870751068593528, 2: 0.19870751068593528, 3: 0.258854060655404, 4: 0.1789639731325056,
       5: 0.0823834724201099, 6: 0.0823834724201099, 7: 0.0}
8  Authorities values = {1: 0.20153417015341699, 2: 0.20153417015341699, 3: 0.25232450023245, 4:
       0.18816829381682937, 5: 0.07821943282194328, 6: 0.07821943282194328, 7: 0.0}
9
10 hussam@hussam-HP-Compaq-nc8430-GE542UP-ABA:~/Desktop/A5/Q1$
```

Listing 2: Running hp.py

Results

Pagerank Algorithm (5 iterations):

Pagerank values:

1: 0.15415210590313183
2: 0.15415210590313183
3: 0.2151836555953412
4: 0.2730700543561289
5: 0.08952435340504106
6: 0.08952435340504106
7: 0.024393371432183873

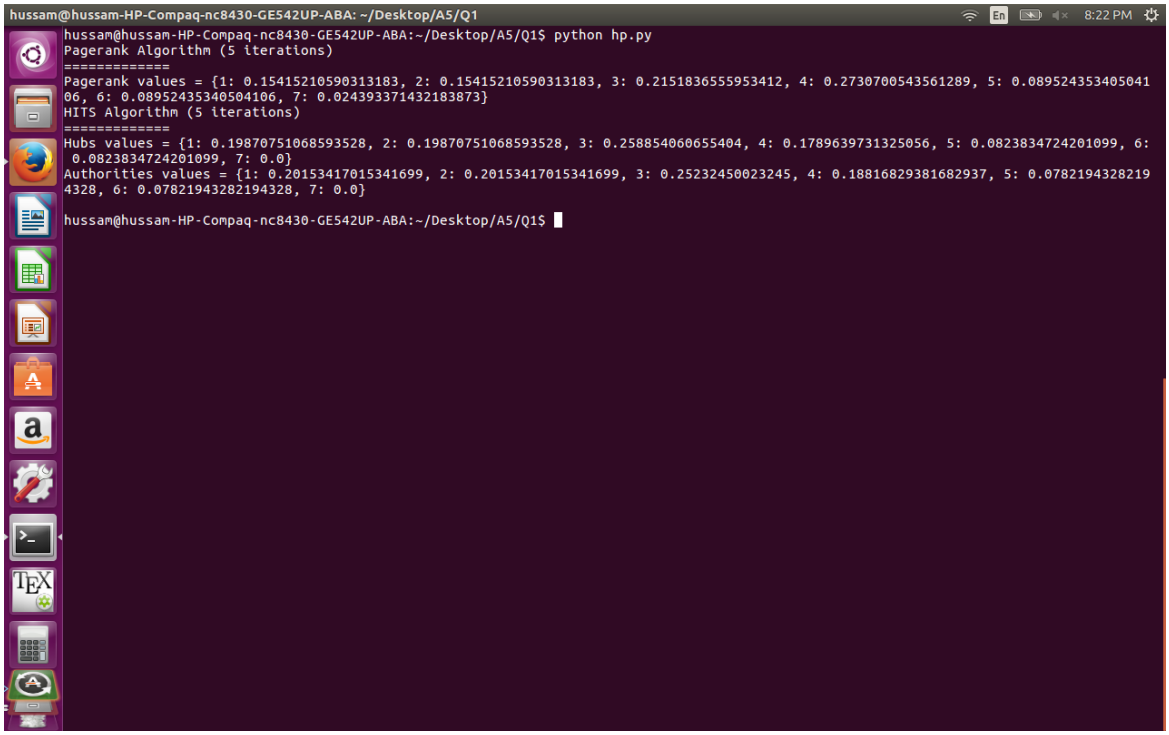
HITS Algorithm (5 iterations):

Hubs values:

1: 0.19870751068593528
2: 0.19870751068593528
3: 0.258854060655404
4: 0.1789639731325056
5: 0.0823834724201099
6: 0.0823834724201099
7: 0.0

Authorities values:
1: 0.20153417015341699
2: 0.20153417015341699
3: 0.25232450023245
4: 0.18816829381682937
5: 0.07821943282194328
6: 0.07821943282194328
7: 0.0

Figure 3: Running hp.py



Observation:

Right of the bat, it is clear that hubs and authorities ranking is consistent for all nodes.

Node 4 placed first in PageRank, but placed fourth in authorities score.

Node 3 placed second in PageRank, but placed first in authorities score.

Nodes 1 and 2 come after node 3 for both authorities score and PageRank.

Nodes 5 and 6 come after 1 and 2 for both authorities score and PageRank.

There is an obvious difference in ranking for both algorithms, but this could be due to the low number of iterations.

Question 2:

Exercise 10.5:

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.

Answer:

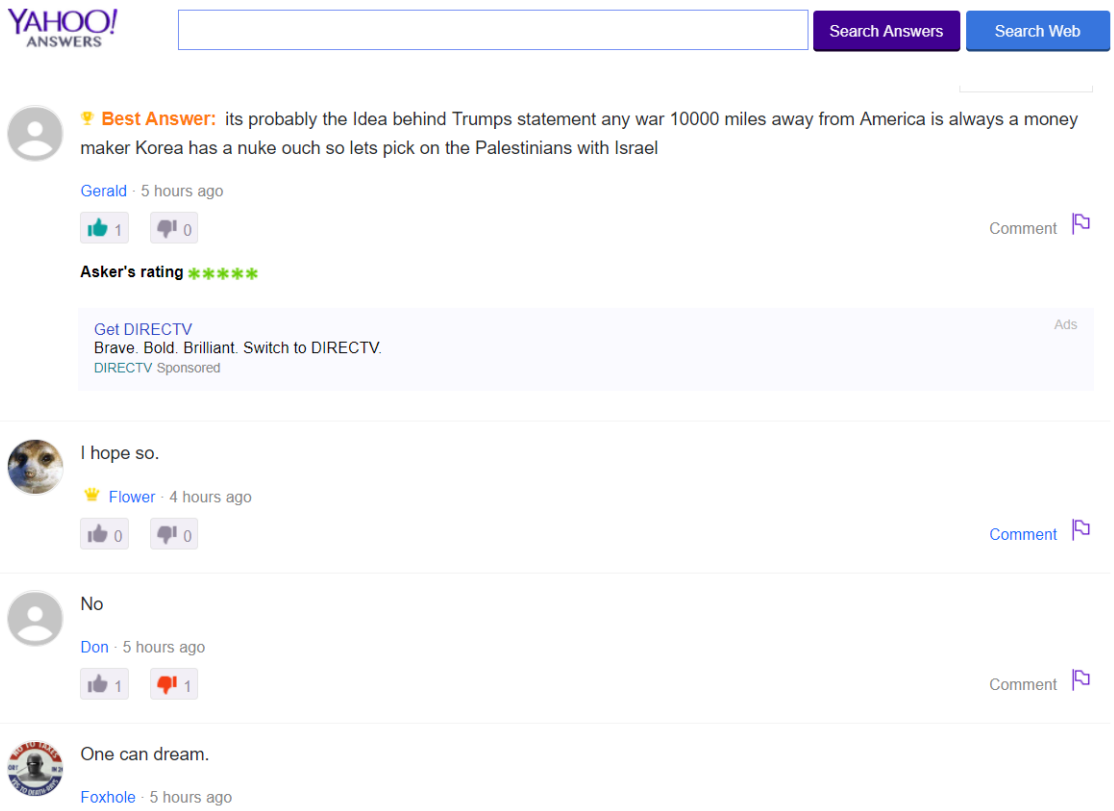
I chose to use Yahoo! Answers and asked the following question as a high quality one:

Will Turkey leave NATO and form a coalition with Russia and Iran after US backed Syrian Kurds and moved US embassy in Tel Aviv to Jerusalem?

I consider this as a high quality question because of its correct grammar and punctuation. It also targets the kind of users that care about politics who are normally serious people. The question is about the possibility of a certain outcome as a result of current events. The answer should be definitive, either yes or no.

Below is a screen shot of the answers I got:

Figure 4: Answers for high quality question



I marked the answer that has a short analysis of why Turkey might leave NATO as best answer. The answers are high quality and meant to answer the question, but they are short, which is expected.

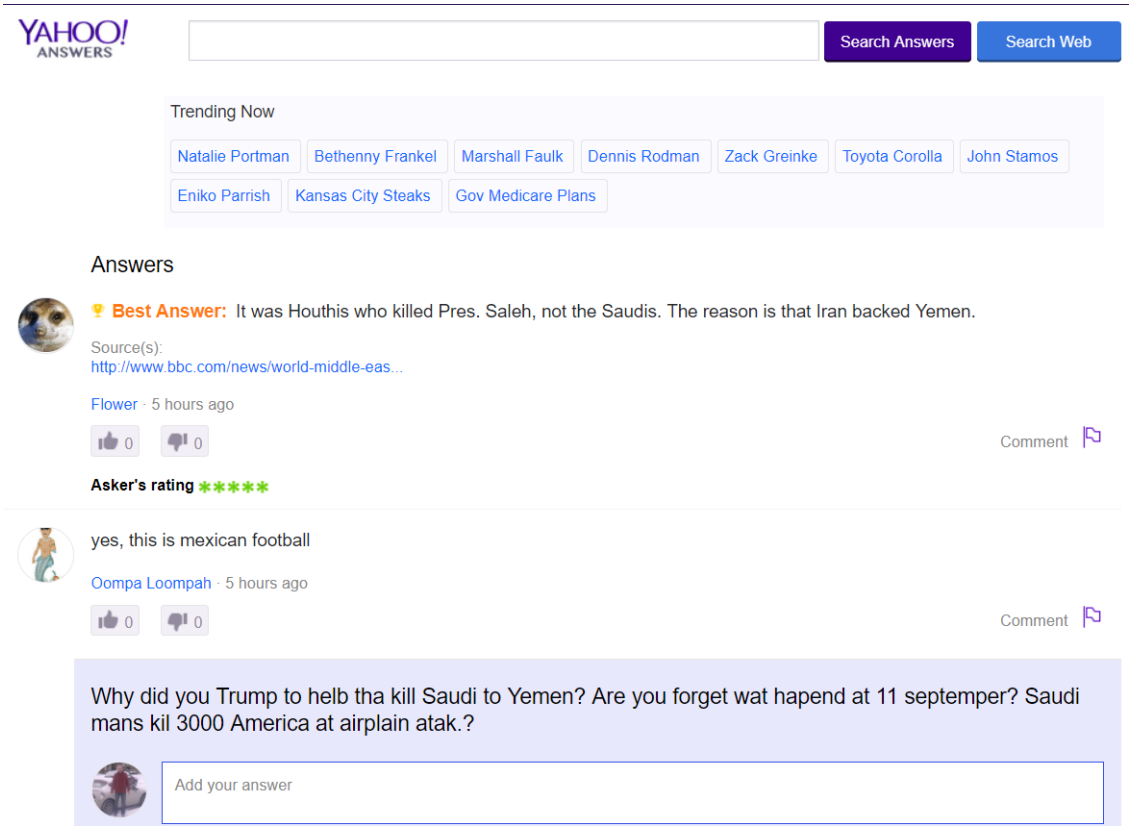
I asked the following question as a low quality one:

Why did you Trump to helb tha kill Saudi to Yemen? Are you forget wat hapend at 11 septemper? Saudi mans kil 3000 America at airplain atak.?

I discovered that asking a low quality question is not an easy task. I was finally able to do it by writing a question then injecting/removing pronouns, verbs, and preposition to produce wrong grammar. I also misspelled some words to make it worse.

Below is a screen shot of the answers I got:

Figure 5: Answers for low quality question



Surprisingly, I got an answer that is, to a certain extent, related to the topic of the question. I would call that a high quality answer because I do not think that it is possible to come up with a much better answer. The answer includes a source link from BBC.

For the other answer, I thought that was funny, but it turns out that Yahoo! put my question in the Mexican football category within sports, and that explains why I got this answer. It is, of course, a low quality answer, and it is definitely not related to the question.

Observation:

The results were as expected; low quality questions get low quality answers, and high quality questions get high quality answers.

Question 3:

Exercise 10.6:

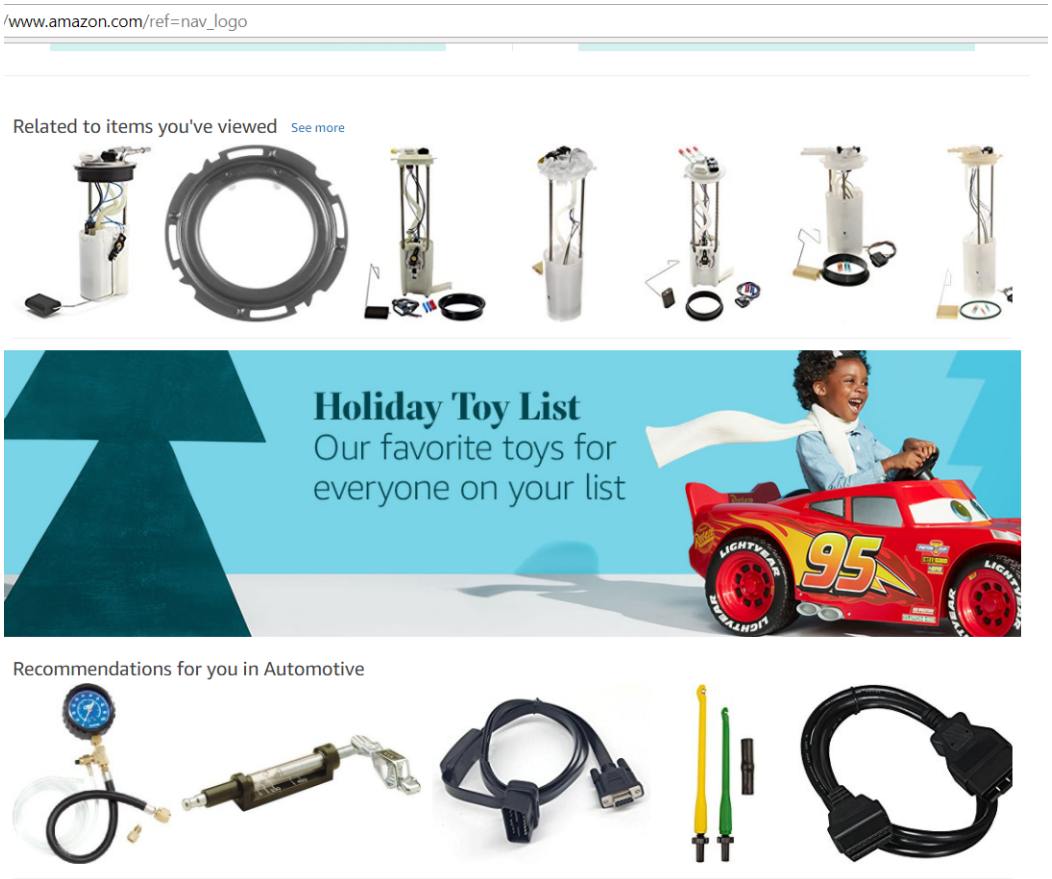
Find two examples of document filtering systems on the Web. How do they build a profile for your information need? Is the system static or adaptive?

Answer:

For my two examples I have chosen Amazon and Youtube. Both of them use an adaptive system for making recommendations.

Amazon is an e-commerce company. I use it sometimes to purchase different items including car parts, electronics, and other items. The screen shot below shows recommended items for me based on items that I viewed, fuel pumps, and items I purchased recently, a cable to connect my truck to my laptop and perform different diagnosis and scanning. Other recommended items are products that were viewed or bought by people whose purchase history is similar to mine. Amazon does such a good job as far as recommendations. I would buy all the recommended items if I had enough money to splurge.

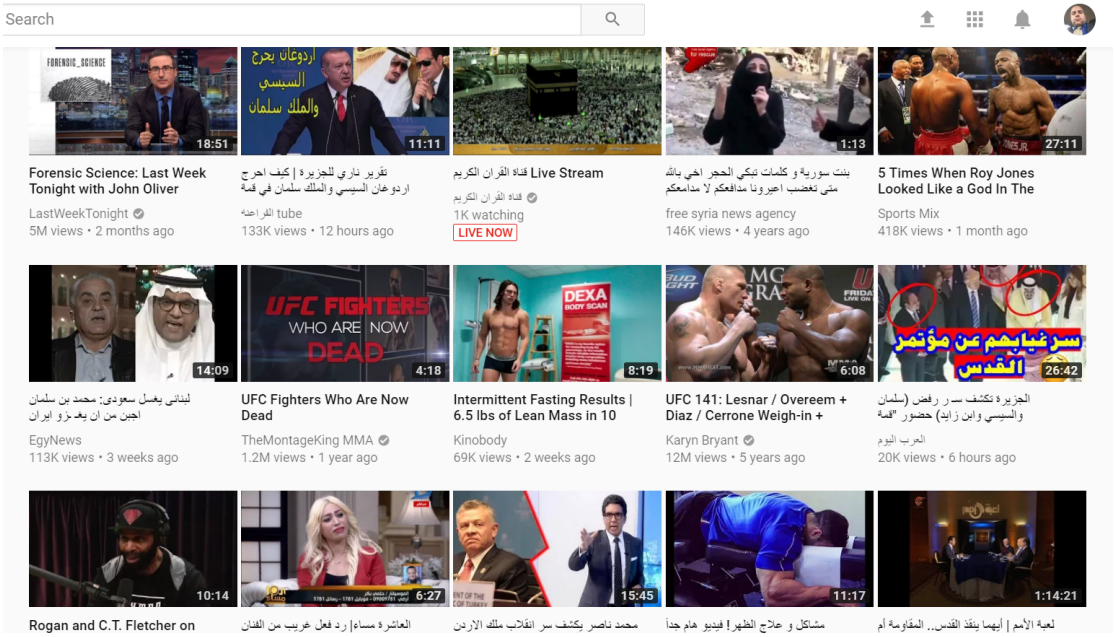
Figure 6: Recommended items by Amazon



Youtube is my favorite website. The reason is that I use it for everything including entertainment, news, music, nutrition, and fitness videos. I also watch short lectures or tutorials about concepts I do not understand in Computer Science. Youtube recommends videos for me based on other videos I watched. It also recommends channels/users for me to subscribe to based on videos I watched or other channels I am subscribed to. It also shows popular uploads from channels I am subscribed to.

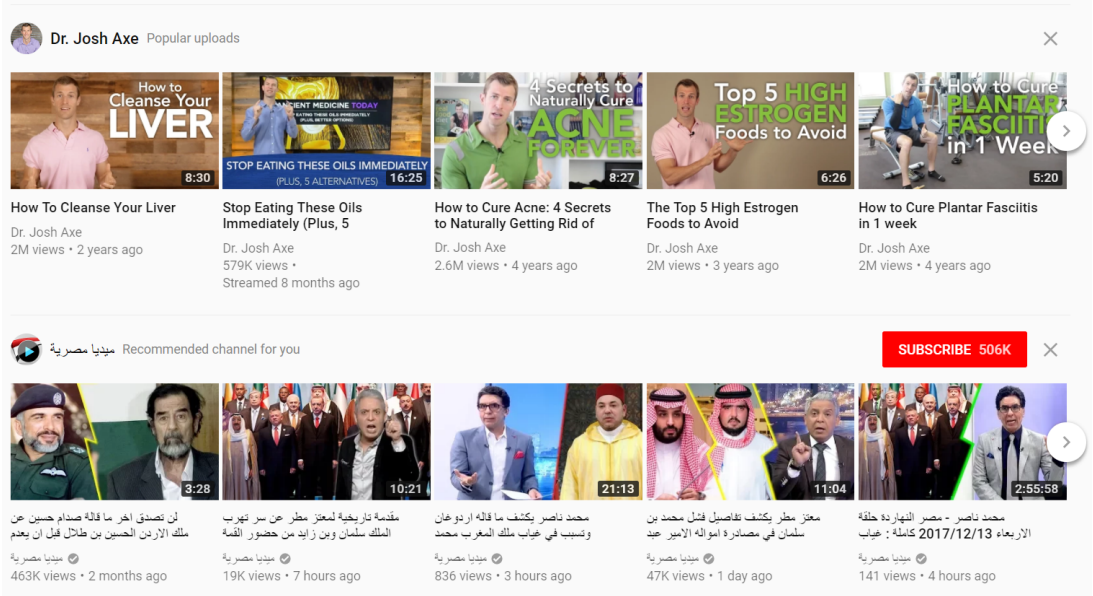
The screen shot below shows a list of recommended videos, based on videos I watched recently. The recommended videos belong to different categories including entertainment, news, music, fitness, boxing, UFC, ...etc.

Figure 7: Recommended items by Youtube



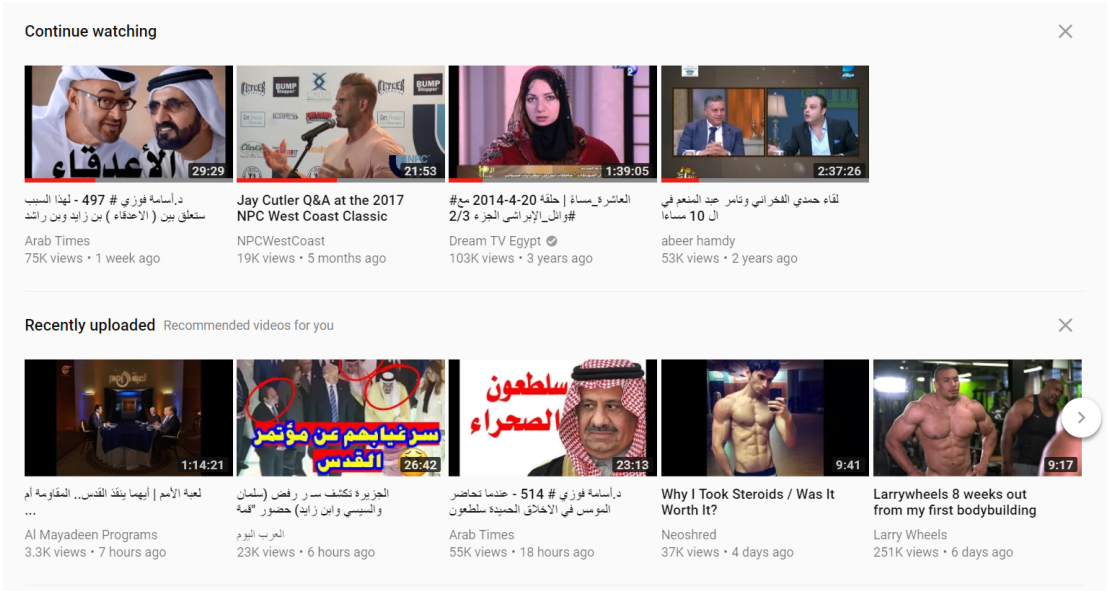
The screen shot below shows a list of popular uploads from a nutrition channel I am subscribed to, Dr. Josh Axe, as well as a recommended channel for my to subscribe to based on videos I watched and channels I am subscribed to. The recommended channel is an Arabic media channel that posts news from the middle east and other videos about politics.

Figure 8: Recommended items by Youtube



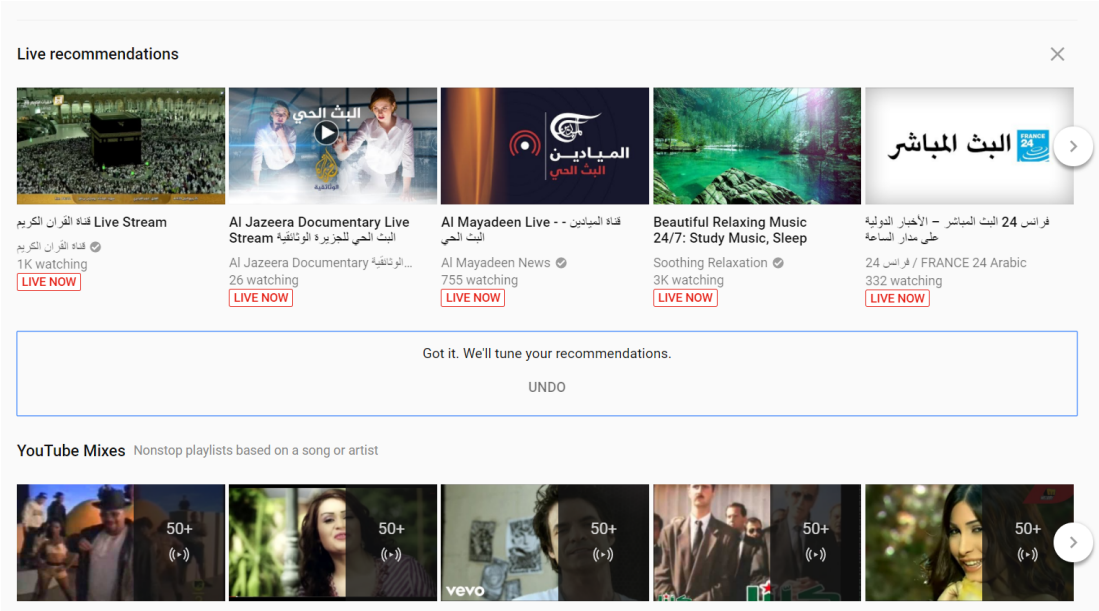
Youtube also shows a list of videos that I did not complete in case I wanted to continue to watch them. It also shows a list of recently uploaded videos that belong to categories in which I am interested based on my views history as the screen shot below shows.

Figure 9: Recommended items by Youtube



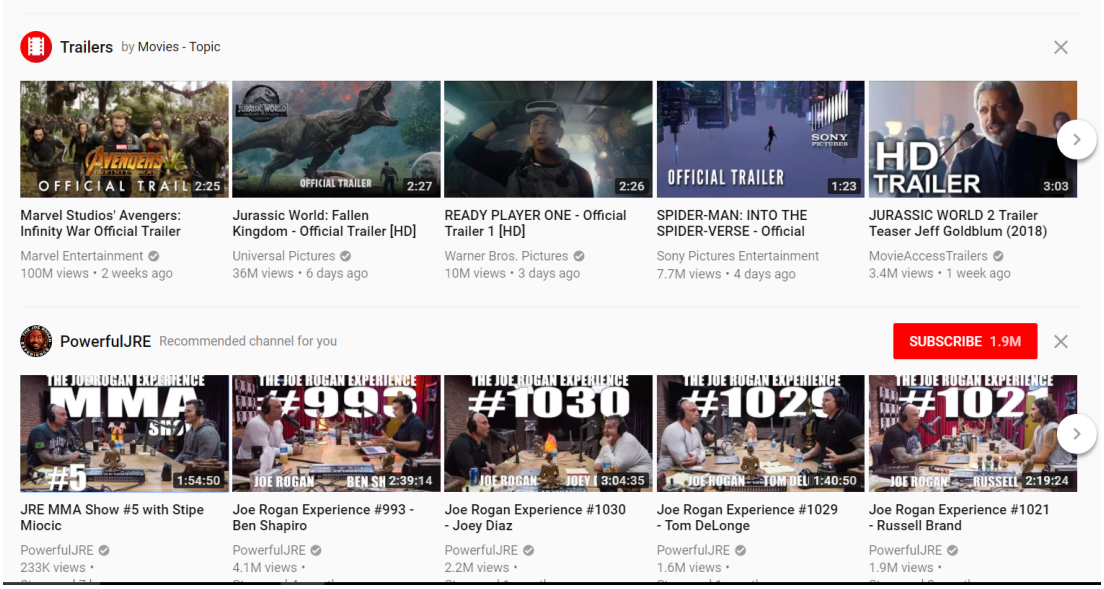
Youtube shows a list of live recommendations based on live channels I watch, mostly news. It also shows recommended a nonstop playlists ,Youtube mixes, based on a song or an atrist from my views history. I turned off recommended videos from one of the channels I am not interested in, RT Arabic. Youtube says: Got it. We'll tune your recommendations. It gave me the option to undo in case my cat turned that recommendation off for me. The screen shot below shows what I described.

Figure 10: Recommended items by Youtube



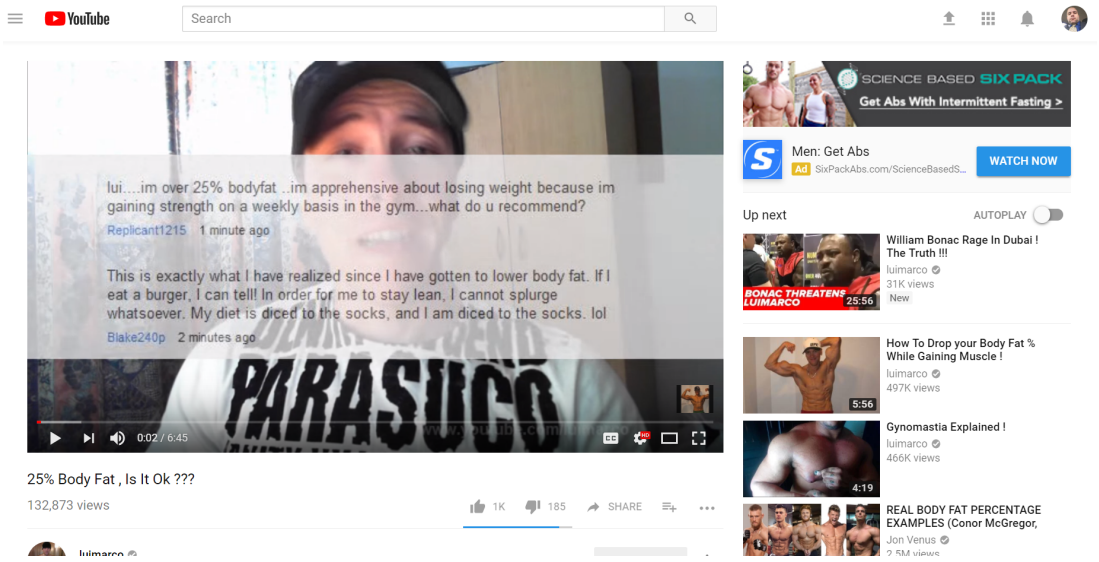
So far, it seems like Youtube is recommending the right videos for me, however, it also shows videos I am not interested in. This could be because these videos have so many views, millions, as seen in the next screen shot. I am not interested in Trailers, but Youtube shows it anyways.

Figure 11: Recommended items by Youtube



Finally, Youtube places an ad, sponsored video, in the upper right corner on video pages I watch. The users who made these videos pay Youtube to show these videos to users based on the video that is on the same page. I have AdBlock installed on my browser so I do not have to see all kinds of advertisements. I disabled AdBlock to take the screen shot below.

Figure 12: Recommended items by Youtube



Question 4:

Exercise 10.8:

Implement the nearest neighbor-based collaborative filtering algorithm. Using a publicly available collaborative filtering data set, compare the effectiveness, in terms of mean squared error, of the Euclidean distance and correlation similarity.

Answer:

I downloaded the latest MovieLens small data set, ml-latest-small, from this link:

http://files.grouplens.org/datasets/movielens/ml-latest-small.zip

This data set (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100004 ratings and 1296 tag applications across 9125 movies.

These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016.

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

The data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv under Q4 directory.

I implemented the nearest neighbor-based collaborative filtering algorithm in python and saved it in the file nn.py. It takes one command line argument, which is the ratings file in csv format. The program computes MSE of Euclidean distance and correlation similarity.

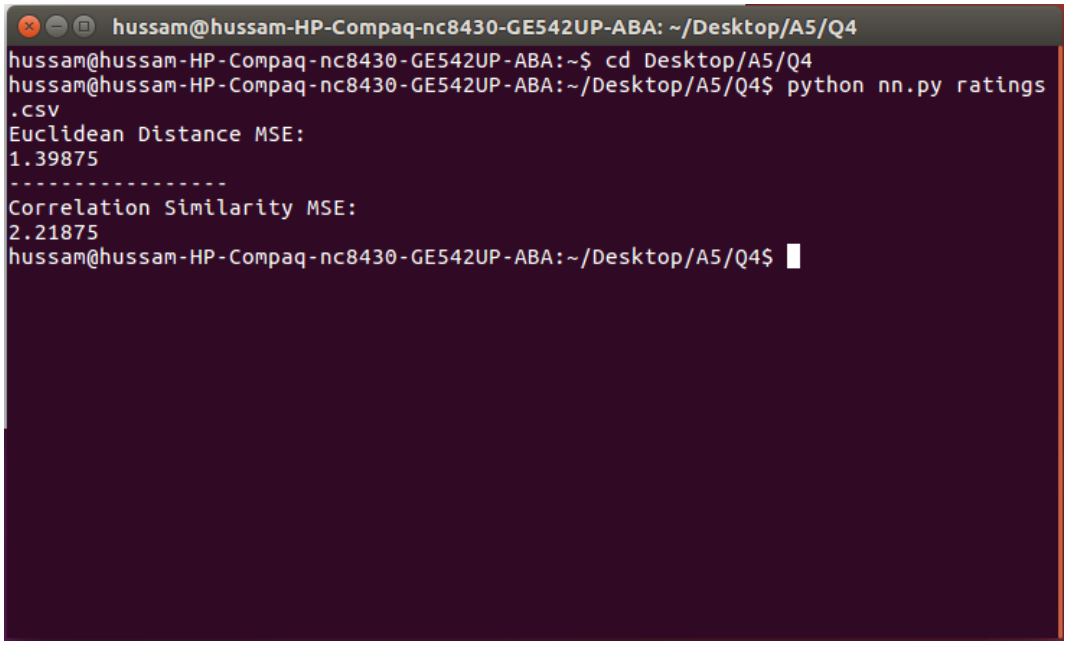
```
import sys
import io
from numpy import *
from sklearn.linear_model import *
from sklearn.metrics import *
from sklearn.neighbors import *

if len(sys.argv) != 2:
    print "Usage: python nn.py <ratings_csv_file>"
    print "Example: python nn.py ratings.csv "
    exit()

ratings = genfromtxt(sys.argv[1], delimiter=',', skip_header=1, usecols=range(3))
random.shuffle(ratings)
train, test = ratings[len(ratings)/10:, :], ratings[:len(ratings)/10, :]
neighbors = KNeighborsClassifier(n_neighbors=1)
neighbors.fit(train, train[:, 0])
distances, neighbors_indices = neighbors.kneighbors(test)
euclidean_neighbors = array([train[ns[0]] for ns in neighbors_indices])
linear_reg = LinearRegression()
linear_reg.fit(train, train[:, 0])
neighbors_idx = linear_reg.predict(test)
pearson_neighbors = array([train[int(idx)] for idx in neighbors_idx])
y_true = []
for user, movie, rating in test:
        y_true.append(rating)
y_pred_euc = []
for user, movie, rating in euclidean_neighbors:
        y_pred_euc.append(rating)
y_pred_pear = []
for user, movie, rating in pearson_neighbors:
        y_pred_pear.append(rating)

print 'Euclidean Distance MSE:'
print mean_squared_error(y_true, y_pred_euc)
```

```
36 print '-----------------'
37 print 'Correlation Similarity MSE:'
38 print mean_squared_error(y_true, y_pred_pear)
```

Listing 3: The content of nn.py

Results:

Figure 13: Running nn.py



Euclidean Distance MSE = 1.39875
Correlation Similarity MSE = 2.21875

Observation:

The MSE of Euclidean Distance is smaller than the MSE of the Correlation Similarity. Therefore the Euclidean Distance is more effective than Correlation Similarity.

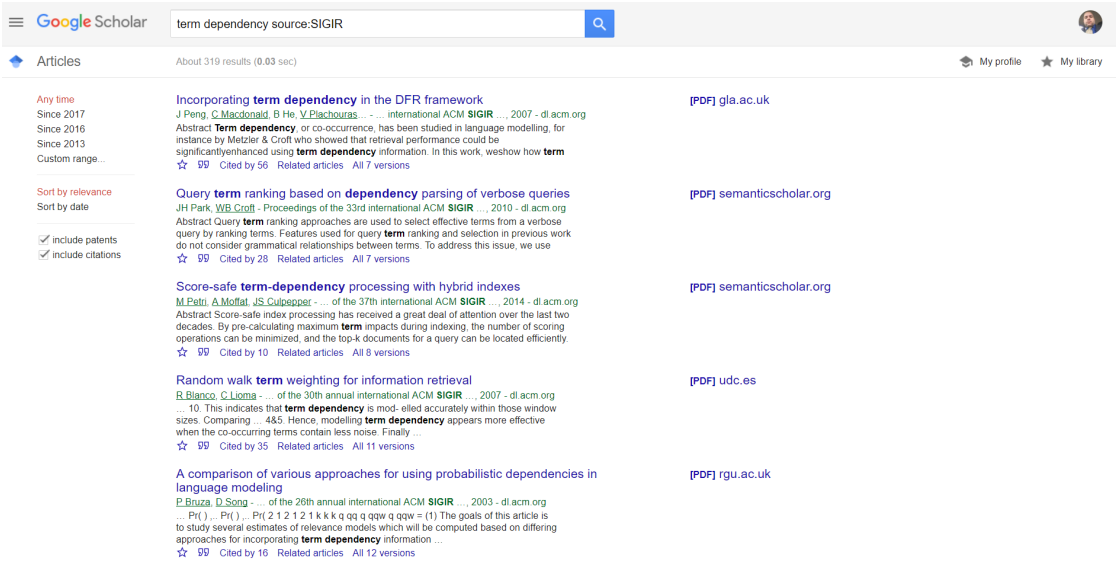
Question 5:

Exercise 11.5: How many papers dealing with term dependency can you find in the SIGIR proceedings since 2000? List their citations.

Answer:

To answer this question, I ran the query "term dependency source:SIGIR" in Google scholar.

Figure 14: Results returned by Google scholar for the query "term dependency source:SIGIR"



The query returned 319 results, however, some of the results are papers dealing with term dependency prior to 2000. I filtered the results by hand and organized them in the following table:

Question 6:

Extra Credit: SVMlight, 10 points extra credit:

see: http://www.cs.cornell.edu/People/tj/svm_light/

* 1 point:

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

Answer:

I downloaded and installed the SVM light from:

http://www.cs.cornell.edu/people/tj/svm_light/

I followed the installation instructions and downloaded the Inductive SVM example.

I also followed the instructions to run svm_learn and passed the training data file train.dat as a command line argument. It produced the model file, which I used as an input for the program svm_classify.

I saved the entire session in the file svm.txt under Q6 folder.

Below is the content of the file.

```
hhallak@atria:~$ wget http://download.joachims.org/svm_light/current/svm_light.tar.gz
--2017-12-15 03:32:51-- http://download.joachims.org/svm_light/current/svm_light.tar.gz
Resolving download.joachims.org (download.joachims.org)... 81.88.34.174, 81.88.42.187
Connecting to download.joachims.org (download.joachims.org)|81.88.34.174|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://osmot.cs.cornell.edu/svm_light/current/svm_light.tar.gz [following]
--2017-12-15 03:32:52-- http://osmot.cs.cornell.edu/svm_light/current/svm_light.tar.gz
Resolving osmot.cs.cornell.edu (osmot.cs.cornell.edu)... 128.253.51.182
Connecting to osmot.cs.cornell.edu (osmot.cs.cornell.edu)|128.253.51.182|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 51026 (50K) [application/x-gzip]
Saving to: Śsvm_light.tar.gzŠ

100%[======================================>] 51,026 203KB/s in 0.2s

2017-12-15 03:32:52 (203 KB/s) - Śsvm_light.tar.gzŠ saved [51026/51026]

hhallak@atria:~$ ls
532             git                       mid779       svm_light
779             Gorgon                    Music        svm_light.tar.gz
cs476Drwahab hhallak_cert_request  Output.odt   Templates
cs772           hhallak_privatekey.pem Pictures     Videos
cs772A2.pdf     hhallakPrivateKey.pem  Public       WINDOWS
Desktop         hhallak_publickey.pem  public_html win_user_profile
Documents       hhallakPublicKey.pem   python       workspace
Downloads       hussam                 setenvwahab
 final          M1                     sqlnet.log
hhallak@atria:~$ cp svm_light.tar.gz ./svm_light
hhallak@atria:~$ cd svm_light/
hhallak@atria:~/svm_light$ ls
svm_light.tar.gz
hhallak@atria:~/svm_light$ gunzip -c svm_light.tar.gz | tar xvf -
LICENSE.txt
Makefile
svm_learn.c
kernel.h
svm_learn.h
svm_learn_main.c
svm_classify.c
svm_loqo.c
svm_common.c
svm_common.h
svm_hideo.c
hhallak@atria:~/svm_light$ make all
gcc -c  -O3                       svm_learn_main.c -o svm_learn_main.o
```

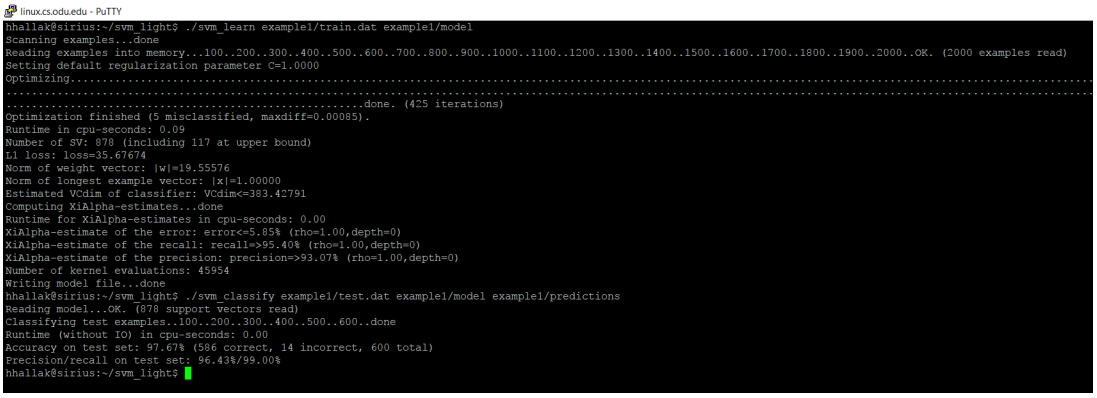
```
gcc -c -O3                    svm_learn.c -o svm_learn.o
gcc -c -O3                    svm_common.c -o svm_common.o
svm_common.c: In function ŚreadModelŠ:
svm_common.c:600:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"SVM-light Version %s\n",version_buffer);
         ^
svm_common.c:605:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%ld%*[^\n]\n", &model->kernel_parm.kernel_type);
         ^
svm_common.c:606:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%ld%*[^\n]\n", &model->kernel_parm.poly_degree);
         ^
svm_common.c:607:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%lf%*[^\n]\n", &model->kernel_parm.rbf_gamma);
         ^
svm_common.c:608:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%lf%*[^\n]\n", &model->kernel_parm.coef_lin);
         ^
svm_common.c:609:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%lf%*[^\n]\n", &model->kernel_parm.coef_const);
         ^
svm_common.c:610:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%[^#]%*[^\n]\n", model->kernel_parm.custom);
         ^
svm_common.c:612:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%ld%*[^\n]\n", &model->totwords);
         ^
svm_common.c:613:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%ld%*[^\n]\n", &model->totdoc);
         ^
svm_common.c:614:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%ld%*[^\n]\n", &model->sv_num);
         ^
svm_common.c:615:9: warning: ignoring return value of ŚfscanfŠ, declared with attribute warn_unused_result [-
    Wunused-result]
  fscanf(modelfl,"%lf%*[^\n]\n", &model->b);
         ^
svm_common.c:623:10: warning: ignoring return value of ŚfgetsŠ, declared with attribute warn_unused_result [-
    Wunused-result]
    fgets(line,(int)ll,modelfl);
          ^
gcc -c -O3                    svm_hideo.c -o svm_hideo.o
gcc -O3                    svm_learn_main.o svm_learn.o svm_common.o svm_hideo.o -o svm_learn -L.
    -lm
gcc -c -O3                    svm_classify.c -o svm_classify.o
gcc -O3                    svm_classify.o svm_common.o -o svm_classify -L. -lm
hhallak@atria:~/svm_light$ ls
kernel.h       svm_classify.c  svm_common.o svm_learn.c     svm_learn.o
LICENSE.txt svm_classify.o  svm_hideo.c  svm_learn.h       svm_light.tar.gz
Makefile       svm_common.c svm_hideo.o  svm_learn_main.c svm_loqo.c
svm_classify  svm_common.h svm_learn    svm_learn_main.o
hhallak@atria:~/svm_light$ wget http://download.joachims.org/svm_light/examples/example1.tar.gz
--2017-12-15 03:38:15-- http://download.joachims.org/svm_light/examples/example1.tar.gz
Resolving download.joachims.org (download.joachims.org)... 81.88.34.174, 81.88.42.187
Connecting to download.joachims.org (download.joachims.org)|81.88.34.174|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://osmot.cs.cornell.edu/svm_light/examples/example1.tar.gz [following]
--2017-12-15 03:38:15-- http://osmot.cs.cornell.edu/svm_light/examples/example1.tar.gz
Resolving osmot.cs.cornell.edu (osmot.cs.cornell.edu)... 128.253.51.182
Connecting to osmot.cs.cornell.edu (osmot.cs.cornell.edu)|128.253.51.182|:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 1223606 (1.2M) [application/x-gzip]
Saving to: Śexample1.tar.gzŠ

100%[======================================>] 1,223,606 1020KB/s in 1.2s

2017-12-15 03:38:17 (1020 KB/s) - Śexample1.tar.gzŠ saved [1223606/1223606]

hhallak@atria:~/svm_light$ ls
example1.tar.gz  svm_classify.c  svm_hideo.c svm_learn_main.c
kernel.h         svm_classify.o  svm_hideo.o svm_learn_main.o
LICENSE.txt      svm_common.c svm_learn   svm_learn.o
Makefile         svm_common.h svm_learn.c svm_light.tar.gz
svm_classify     svm_common.o svm_learn.h svm_loqo.c
hhallak@atria:~/svm_light$ gunzip -c example1.tar.gz | tar xvf -
example1/
example1/train.dat
example1/test.dat
example1/words
hhallak@atria:~/svm_light$ ./svm_learn example1/train.dat example1/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: 0.10
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
hhallak@atria:~/svm_light$ ./svm_classify example1/test.dat example1/model example1/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 4: The content of svm.txt

The results were printed on the screen:
Accuracy on test set: 97.67
Precision/recall on test set: 96.43
The results show that these scores are very high for these measures.

Figure 15: Inductive SVM Example



References

[1] Stackoverflow. https://stackoverflow.com/questions/tagged/python.

[2] http://convertjson.com/xml-to-json.htm

[3] https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html

[4] https://sourceforge.net/p/lemur/wiki/Galago%20Query%20Language/?version=2

[5] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot