In [27]:
```
import graphlab as gl
import numpy as np
import matplotlib.pyplot as plt
```

In [148]:
```
r = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5-grp9/data/reviews_Kindle_Store.json', delimiter='\n', header=False)

reviews = r.unpack(unpack_column='X1',column_name_prefix='')
reviews = reviews.unpack(unpack_column='helpful',column_name_prefix='X')
reviews.rename({'X.0':'upvotes','X.1':'downvotes'})
reviews['reviewTime'] = reviews['reviewTime'].str_to_datetime(str_format="%m %d, %Y")
reviews['tfidf'] = gl.text_analytics.tf_idf(reviews['reviewText'])
reviews['tfidf'] = reviews['tfidf'].dict_trim_by_keys(gl.text_analytics.stopwords(), True)

d = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5-grp9/data/meta_Kindle_Store.json', delimiter='\n', header=False)

meta = d.unpack(unpack_column='X1',column_name_prefix='')
msf = reviews.join(meta, on='asin', how='inner')
```

```
Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/data/reviews_Kindle_Store.json

Parsing completed. Parsed 100 lines in 1.0601 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[dict]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Read 73750 lines. Lines per second: 35263.8

Read 370288 lines. Lines per second: 47279.9

Read 682265 lines. Lines per second: 48145.9

Read 990624 lines. Lines per second: 49016.8

Read 1311842 lines. Lines per second: 49649.1

Read 1635900 lines. Lines per second: 50764.1
```

```
Read 1878365 lines. Lines per second: 49403.4

Read 2184041 lines. Lines per second: 49644.9

Read 2477488 lines. Lines per second: 49899.2

Read 2768812 lines. Lines per second: 50321.1

Read 3050814 lines. Lines per second: 49703

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/reviews_Kindle_Store.json

Parsing completed. Parsed 3205467 lines in 64.317 secs.

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/meta_Kindle_Store.json

Parsing completed. Parsed 100 lines in 1.69598 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[dict]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Read 34790 lines. Lines per second: 9416.32

Read 158032 lines. Lines per second: 14540.6

Read 276927 lines. Lines per second: 15224.5

Read 352318 lines. Lines per second: 15167.8

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/meta_Kindle_Store.json

Parsing completed. Parsed 434702 lines in 27.7522 secs.
```

In [147]:
```
msf.head()
```

Out[147]:

asin	brand	categories	description	imUrl
1603420304	None	[[Books, Cookbooks, Food & Wine, Quick & Easy], ...	In less time and for less money than it takes to ...	http://ecx.images-amazon.com/images/I/51IEqP ...

B0002IQ15S	None	[[Kindle Store, Kindle Accessories, Power ...	This universal DC adapter powers/charges portable ...	http://ecx.images-amazon.com/images/I/21QFJM ...
B000F83SZQ	None	[[Books, Literature & Fiction], [Books, ...	None	http://ecx.images-amazon.com/images/I/51yLqHe ...
B000F83TEQ	None	[[Books, Literature & Fiction], [Books, ...	None	http://ecx.images-amazon.com/images/I/2136NB ...
B000F83STC	None	[[Books, Literature & Fiction, Erotica], ...	None	http://g-ecx.images-amazon.com/images/G/01/x ...
B000FA5RE4	None	[[Books, Reference, Dictionaries & ...	Updated 2003 version with corrections ...	http://ecx.images-amazon.com/images/I/21XTM6 ...
B000FA5NSO	None	[[Books, Literature & Fiction], [Books, Sci ...	None	http://ecx.images-amazon.com/images/I/51N45E ...
B000FA5UXC	None	[[Books, Literature & Fiction], [Books, Sci ...	Madoc Tamlin is a man with an unusual problem. ...	http://ecx.images-amazon.com/images/I/51q4iur5
B000FA5T4W	None	[[Books, Business & Money, Accounting], ...	None	http://ecx.images-amazon.com/images/I/51eWyE ...
B000FA5SHK	None	[[Books, Science &	None	http://ecx.images-amazon.

In [3]:
```
msf.column_names()
```

Out[3]:
```
['asin',
 'overall',
 'reviewText',
 'reviewTime',
 'reviewerID',
 'reviewerName',
 'summary',
 'unixReviewTime',
 'upvotes',
 'downvotes',
 'tfidf',
 'brand',
 'categories',
 'description',
 'imUrl',
 'price',
 'related',
 'salesRank',
 'title']
```

Get an SArray of the concatenated text in the `summary`, `reviewText`, and `description` fields.

In [21]:
```
docs = msf.apply(lambda x: str(x['summary']) + ' ' + str(x['reviewText
']) + ' ' + str(x['description']))
```

Create a function to count words from a `docs` SArray that outputs a `docs_sf` SFrame with associated word counts

In [26]:
```
def get_word_frequency(docs):
    """
    Returns the frequency of occurrence of words in an SArray of documents
    Args:
    docs: An SArray (of dtype str) of documents
    Returns:
    An SFrame with the following columns:
     'word'      : Word used
     'count'     : Number of times the word occured in all documents.
     'frequency' : Relative frequency of the word in the set of input documents.
    """

    # Use the count_words function to count the number of words.
    docs_sf = gl.SFrame()
    docs_sf['words'] = gl.text_analytics.count_words(docs)

    # Stack the dictionary into individual word-count pairs.
    docs_sf = docs_sf.stack('words',
                            new_column_name=['word', 'count'])

    # Count the number of unique words (remove None values)
    docs_sf = docs_sf.groupby('word', {'count': gl.aggregate.SUM('count')})
    docs_sf['frequency'] = docs_sf['count'] / docs_sf["count"].sum()
    return docs_sf
```

In [ ]:
```
docs_sf = get_word_frequency(docs)
```

In [ ]:
```
def predict(document_bow, word_topic_counts, topic_counts, vocab,
            alpha=0.1, beta=0.01, num_burnin=5):
    """
    Make predictions for a single document.
    Parameters
    ----------
    document_bow : dict
        Dictionary with words as keys and document frequencies as counts.
    word_topic_counts : numpy array, num_vocab x num_topics
        Number of times a given word has ever been assigned to a topic.
    topic_counts : numpy vector of length num_topics
        Number of times any word has been assigned to a topic.
    vocab : dict
        Words are keys and unique integer is the value.
    alpha : float
        Hyperparameter. See topic_model docs.
```

```
    beta : float
        Hyperparameter. See topic_model docs.
    num_burnin : int
        Number of iterations of Gibbs sampling to perform at predict t
ime.
    Returns
    -------
    out : numpy array of length num_topics
        Probabilities that the document belongs to each topic.
    """
    num_vocab, num_topics = word_topic_counts.shape

    # proportion of each topic in this test doc
    doc_topic_counts = np.zeros(num_topics)
    # Assignment of each unique word
    doc_topic_assignments = []

    # Initialize assignments and counts
    # NB: we are assuming document_bow doesn't change.
    for i, (word, freq) in enumerate(document_bow.iteritems()):
        if word not in vocab:  # skip words not present in training se
t
            continue
        topic = np.random.randint(0, num_topics-1)
        doc_topic_assignments.append(topic)
        doc_topic_counts[topic] += freq

    # Sample topic assignments for the test document
    for burnin in range(num_burnin):
        for i, (word, freq) in enumerate(document_bow.iteritems()):
            if word not in vocab:
                continue
            word_id = vocab[word]

            # Get old topic and decrement counts
            topic = doc_topic_assignments[i]
            doc_topic_counts[topic] -= freq

            # Sample a new topic
            gamma = np.zeros(num_topics)  # store probabilities
            for k in range(num_topics):
                gamma[k] = (doc_topic_counts[k] + alpha) * (word_topic
_counts[word_id, k] + beta) / (topic_counts[k] + num_vocab * beta)
            gamma = gamma / gamma.sum()  # normalize to probabilities
            topic = np.random.choice(num_topics, 1, p=gamma)

            # Use new topic to increment counts
            doc_topic_assignments[i] = topic
            doc_topic_counts[topic] += freq
```

```
    # Create predictions
    predictions = np.zeros(num_topics)
    total_doc_topic_counts = doc_topic_counts.sum()
    for k in range(num_topics):
        predictions[k] = (doc_topic_counts[k] + alpha) / (total_doc_to
pic_counts + num_topics * alpha)
    return predictions / predictions.sum()


    if __name__ == '__main__':
    docs = gl.SFrame({'text': [{'first': 5, 'doc': 1}, {'second': 3, '
doc': 5}]})
    m = gl.topic_model.create(docs)

    # Get test document in bag of words format
    document_bow = docs['text'][0]

    # Input: Global parameters from trained model

    # Number of times each word in the vocabulary has ever been assign
ed to topic k (in any document). You can make an approximate version o
f this by multiplying m['topics'] by some large number (e.g. number of
tokens in corpus) that indicates how strong you "believe" in these top
ics. Make it into counts by flooring it to an integer.
    prior_strength = 1000000
    word_topic_counts = np.array(m['topics']['topic_probabilities'])
    word_topic_counts = np.floor(prior_strength * word_topic_counts)

    # Number of times any word as been assigned to each topic.
    topic_counts = word_topic_counts.sum(0)

    # Get vocabulary lookup
    num_topics = m['num_topics']
    vocab = {}
    for i, w in enumerate(m['topics']['vocabulary']):
        vocab[w] = i
    num_vocab = len(vocab)

    # Make prediction on test document
    probs = predict(document_bow, word_topic_counts, topic_counts, voc
ab)
```

In [16]:
```
print msf[1]['summary']
print msf[1]['reviewText']
print msf[1]['description']
```

```
Okay for true beginners
So, I bought this book a few days ago and have tried three recipes s
o far.  The first was a total flop.  There must be an error, but be
forewarned, do NOT make the Blueberry Coffee Cake as it comes out as
inedible mush--WAY too much water.  The other two recipes (mac and c
heese and grilled cheese with tomato) were decent for quick lunches
or dinners.  They were average in taste, but considering the short a
mount of time it took to make them, I'm okay with that.  All in all,
it's a nice idea book to get creative with everyday ingredients, but
with errors and only average taste, I give it three stars.
In less time and for less money than it takes to order pizza, you ca
n make it yourself!Three harried but heatlh-conscious college studen
ts compiled and tested this collection of more than 200 tasty, heart
y, inexpensive recipes anyone can cook -- yes, anyone!Whether you're
short on cash, fearful of fat, counting your calories, or just miss
home cooking, The Healthy College Cookbook offers everything you nee
d to make good food yourself.
```

In [8]:
```
wc = gl.text_analytics.count_words(docs, to_lower=True)
```

In [13]:
```
trimmer = gl.toolkits.feature_engineering.RareWordTrimmer(threshold=2)

# Fit and transform the data.
transformed_sf = trimmer.fit_transform(wc)
```

```
---------------------------------------------------------------------
-------
ToolkitError                                Traceback (most recent cal
l last)
<ipython-input-13-0457887f7cbe> in <module>()
      2
      3 # Fit and transform the data.
----> 4 transformed_sf = trimmer.fit_transform(wc)

/Users/galen/anaconda/envs/gl-env/lib/python2.7/site-packages/graphl
ab/toolkits/feature_engineering/_doc_utils.pyc in fit_transform(self
, data)
     31
     32 def fit_transform(self, data):
---> 33     return Transformer.fit_transform(self, data)
     34
     35 def republish_docs(cls):

/Users/galen/anaconda/envs/gl-env/lib/python2.7/site-packages/graphl
ab/toolkits/feature_engineering/_feature_engineering.pyc in fit_tran
sform(self, data)
    319         {examples}
    320         """
--> 321         _raise_error_if_not_sframe(data, "data")
    322         _mt._get_metric_tracker().track(self.__class__.__mod
ule__ + '.fit_transform')
    323         return self.__proxy__.fit_transform(data)

/Users/galen/anaconda/envs/gl-env/lib/python2.7/site-packages/graphl
ab/toolkits/_internal_utils.pyc in _raise_error_if_not_sframe(datase
t, variable_name)
    395
    396     if not isinstance(dataset, _SFrame):
--> 397       raise ToolkitError(err_msg % variable_name)
    398
    399 def _raise_error_if_sframe_empty(dataset, variable_name="SFr
ame"):

ToolkitError: Input data is not an SFrame. If it is a Pandas DataFra
me, you may use the to_sframe() function to convert it to an SFrame.
```

In [11]:
```
len(wc)
```

Out[11]: 3205467

In [ ]:
```
# Use the count_words function to count the number of words.
docs_sf = gl.SFrame()
docs_sf['words'] = gl.text_analytics.count_words(docs)

# Stack the dictionary into individual word-count pairs.
docs_sf = docs_sf.stack('words',
                        new_column_name=['word', 'count'])

# Count the number of unique words (remove None values)
docs_sf = docs_sf.groupby('word', {'count': gl.aggregate.SUM('count')}
)
docs_sf['frequency'] = docs_sf['count'] / docs_sf["count"].sum()
```

In [ ]:
```

```

Run CTM with Spark

In [1]:
```
import findspark
findspark.init('/Users/Zoe/spark-2.1.0-bin-hadoop2.7/')
from pyspark.sql import SparkSession

from pyspark.context import SparkContext
sc = SparkContext('local')
spark = SparkSession(sc)
```

Prepare data

In [2]:
```
review_df = spark.read.json("/Users/Zoe/Documents/Spring2017/GR5243/My
Prjs/localData/prj5/reviews_Kindle_Store.json")
#meta_df = spark.read.json("/Users/Zoe/Documents/Spring2017/GR5243/MyP
rjs/localData/prj5/meta_Kindle_Store.json")
```

In [3]:
```
df = review_df.select(review_df.asin,review_df.overall,review_df.revie
werID)
```

In [6]:
```
review_df = 0
meta_df = 0
```

In [5]:
```
df.take(5)
```

Out[5]:
```
[Row(asin=u'1603420304', overall=4.0, reviewerID=u'A2GZ9GFZV1LWB0'),
 Row(asin=u'1603420304', overall=3.0, reviewerID=u'A1K7VSUDCVAPW8'),
 Row(asin=u'1603420304', overall=4.0, reviewerID=u'A35J5XRE5ZT6H2'),
 Row(asin=u'1603420304', overall=4.0, reviewerID=u'A3DGZNFSMNWSX5'),
 Row(asin=u'1603420304', overall=5.0, reviewerID=u'A2CVDQ6H36L4VL')]
```

In [7]:
```
asins_code = df.select('asin').distinct().rdd.zipWithIndex()
users_code = df.select('reviewerID').distinct().rdd.zipWithIndex()
```

In [8]:
```
asins_df = spark.createDataFrame(asins_code.map(lambda r: (r[0][0],r[1
]+1)),['asin','item'])
users_df = spark.createDataFrame(users_code.map(lambda r: (r[0][0],r[1
]+1)),['reviewerID','user'])
```

In [9]:
```
Ratings = df.select(df.asin,df.overall,df.reviewerID).join(asins_df,"a
sin").join(users_df,"reviewerID")
```

In [10]:
```
Ratings = Ratings.select(Ratings.user, Ratings.item, Ratings.overall.a
lias('rating'))
```

In [19]:
```
row1 = Ratings.agg({"user": "max", "item":"max"}).collect()
```

In [20]:
```
row1
```

Out[20]:
```
[Row(max(item)=430529, max(user)=1406889)]
```

In [12]:
```
ItemTopics = spark.read.load('/Users/Zoe/Documents/Spring2017/GR5243/M
yPrjs/localData/prj5/predictions.csv',
                      format='com.databricks.spark.csv',
                      header='true',
                      inferSchema='true')
```

In [13]:
```
ItemTopicsRDD = asins_df.join(ItemTopics,"asin").drop("asin").rdd.map(
lambda r: (r[0],[r[i] for i in range(1,51)]))
```

In [14]:
```
ItemTopics = spark.createDataFrame(ItemTopicsRDD,['item','topic'])
```

In [23]:
```
Full = Ratings.join(ItemTopics, "item")
```

In [24]:
```
subFull = Full.limit(20)
```

In [ ]:
```
subFull.collect()
```

In [ ]:
```

```

Train CTM on Data

In [15]:
```
from pyspark.sql.functions import collect_list
from time import time
import numpy as np
from numpy.random import rand
from numpy import matrix
```

In [22]:
```
def CTM_train(Full,I,J,K,LAMBDA,max_iter=10,n_partition=6):
    '''
    '''

    # define update functions
    def updateU(i,v_ind,R,V,LAMBDA):
        '''
        '''
        r = v_ind.shape[0]
        K = V.shape[1]

        A = V[v_ind,:].T.dot(V[v_ind,:]) + LAMBDA*r*np.eye(K)
        b = V[v_ind,:].T.dot(R).T

        return (np.linalg.solve(A, b)).T

    def updateV(j,u_ind,R,U,LAMBDA,Th):
        '''
        '''
        r = u_ind.shape[0]
        K = U.shape[1]

        A = U[u_ind,:].T.dot(U[u_ind,:]) + LAMBDA*r*np.eye(K)
        b = U[u_ind,:].T.dot(R).T + LAMBDA*r*Th.reshape([K,1])

        return (np.linalg.solve(A, b)).T

    print('pre-compute block information...')
    Full = Full.repartition(n_partition)
    U_map = Full.groupBy("user").agg(collect_list("item").alias('items
'),collect_list("rating").alias('ratings')).sort('user')
    V_map = Full.groupBy("item").agg(collect_list("user").alias('users
'),collect_list("rating").alias('ratings'), first('topic').alias('topi
```

```
c')).sort('item')
    U_map = U_map.repartition(n_partition)
    V_map = V_map.repartition(n_partition)

    print('initialize parameters...')
    U = matrix(rand(I,K))
    V = matrix(rand(J,K))

    Us = sc.broadcast(U)
    Vs = sc.broadcast(V)

    print('update parameters...')
    for i in range(max_iter):


        st = time()
        U = U_map.rdd.map(lambda r: updateU(r[0],np.array(r[1]),np.arr
ay(r[2]),Vs.value,LAMBDA)).reduce(lambda a,b: np.vstack((a,b)))
        Us = sc.broadcast(U)



        V = V_map.rdd.map(lambda r: updateV(r[0],np.array(r[1]),np.arr
ay(r[2]),Us.value,LAMBDA,np.array(r[3]))).reduce(lambda a,b: np.vstack
((a,b)))
        Vs = sc.broadcast(V)
        ed = time()



        print('Finish iteration round: '+str(i)+', use time: '+str(rou
nd(ed-st,4))+'s.\n')

    return (U,V)
```

In [21]:
```
# carefully set number of threads to improve performance
U,V = CTM_train(Full,1406889,430529,50,LAMBDA=0.02,max_iter=10,n_parti
tion=200)
```

```
pre-compute block information...
initialize parameters...
update parameters...

---------------------------------------------------------------------
-------
Py4JJavaError                             Traceback (most recent cal
l last)
<ipython-input-21-89aac2a58056> in <module>()
      1 # carefully set number of threads to improve performance
----> 2 U,V = CTM_train(Ratings,ItemTopics,1406889,430529,50,LAMBDA=
0.02,max_iter=10,U_threads=200,V_threads=200)
```

```
<ipython-input-16-804eb679013b> in CTM_train(R, Th, I, J, K, LAMBDA,
max_iter, U_threads, V_threads)
     44
     45         st = time()
---> 46         U = U_map.rdd.map(lambda r: updateU(r[0],np.array(r[
1]),np.array(r[2]),Vs.value,LAMBDA)).reduce(lambda a,b: np.vstack((a
,b)))
     47         Us = sc.broadcast(U)
     48

/Users/Zoe/spark-2.1.0-bin-hadoop2.7/python/pyspark/rdd.pyc in reduc
e(self, f)
    833             yield reduce(f, iterator, initial)
    834
--> 835         vals = self.mapPartitions(func).collect()
    836         if vals:
    837             return reduce(f, vals)

/Users/Zoe/spark-2.1.0-bin-hadoop2.7/python/pyspark/rdd.pyc in colle
ct(self)
    807         """
    808         with SCCallSiteSync(self.context) as css:
--> 809             port = self.ctx._jvm.PythonRDD.collectAndServe(s
elf._jrdd.rdd())
    810         return list(_load_from_socket(port, self._jrdd_deser
ializer))
    811

/Users/Zoe/spark-2.1.0-bin-hadoop2.7/python/lib/py4j-0.10.4-src.zip/
py4j/java_gateway.py in __call__(self, *args)
   1131         answer = self.gateway_client.send_command(command)
   1132         return_value = get_return_value(
-> 1133             answer, self.gateway_client, self.target_id, sel
f.name)
   1134
   1135         for temp_arg in temp_args:

/Users/Zoe/spark-2.1.0-bin-hadoop2.7/python/pyspark/sql/utils.pyc in
deco(*a, **kw)
     61     def deco(*a, **kw):
     62         try:
---> 63             return f(*a, **kw)
     64         except py4j.protocol.Py4JJavaError as e:
     65             s = e.java_exception.toString()

/Users/Zoe/spark-2.1.0-bin-hadoop2.7/python/lib/py4j-0.10.4-src.zip/
py4j/protocol.py in get_return_value(answer, gateway_client, target_
id, name)
    317                 raise Py4JJavaError(
    318                     "An error occurred while calling {0}{1}{
```

```
2}.\n".
--> 319                     format(target_id, ".", name), value)
    320             else:
    321                 raise Py4JError(

Py4JJavaError: An error occurred while calling z:org.apache.spark.ap
i.python.PythonRDD.collectAndServe.
: org.apache.spark.SparkException: Job aborted due to stage failure:
Task 2 in stage 64.0 failed 1 times, most recent failure: Lost task
2.0 in stage 64.0 (TID 4507, localhost, executor driver): TaskResult
Lost (result lost from block manager)
Driver stacktrace:
        at org.apache.spark.scheduler.DAGScheduler.org$apache$spark$
scheduler$DAGScheduler$$failJobAndIndependentStages(DAGScheduler.sca
la:1435)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortSta
ge$1.apply(DAGScheduler.scala:1423)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$abortSta
ge$1.apply(DAGScheduler.scala:1422)
        at scala.collection.mutable.ResizableArray$class.foreach(Res
izableArray.scala:59)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.
scala:48)
        at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGSch
eduler.scala:1422)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTa
skSetFailed$1.apply(DAGScheduler.scala:802)
        at org.apache.spark.scheduler.DAGScheduler$$anonfun$handleTa
skSetFailed$1.apply(DAGScheduler.scala:802)
        at scala.Option.foreach(Option.scala:257)
        at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFail
ed(DAGScheduler.scala:802)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.d
oOnReceive(DAGScheduler.scala:1650)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.o
nReceive(DAGScheduler.scala:1605)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.o
nReceive(DAGScheduler.scala:1594)
        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.sca
la:48)
        at org.apache.spark.scheduler.DAGScheduler.runJob(DAGSchedul
er.scala:628)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1
918)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1
931)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1
944)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:1
958)
```

```
        at org.apache.spark.rdd.RDD$$anonfun$collect$1.apply(RDD.sca
la:935)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOper
ationScope.scala:151)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOper
ationScope.scala:112)
        at org.apache.spark.rdd.RDD.withScope(RDD.scala:362)
        at org.apache.spark.rdd.RDD.collect(RDD.scala:934)
        at org.apache.spark.api.python.PythonRDD$.collectAndServe(Py
thonRDD.scala:453)
        at org.apache.spark.api.python.PythonRDD.collectAndServe(Pyt
honRDD.scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Metho
d)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodA
ccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Delegatin
gMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:2
44)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.
java:357)
        at py4j.Gateway.invoke(Gateway.java:280)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractComman
d.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:214)
        at java.lang.Thread.run(Thread.java:745)
```

Predict on test data

In [ ]:
```
def CTM_predict(test,U,V):
    '''
    '''


    preds = test.rdd.map(lambda r: ((r[0],r[1]),U[r[0],:].dot(V[r[1],:
].T)[0,0]))
    return preds
```

In [ ]:
```
preds = CTM_predict(Ratings,U,V)
```

Evaluation

1. MSE

In [ ]:
```
def evaluate(test,preds,N):
    '''
    '''


    se = test.rdd.map(lambda r: ((r[0],r[1]),r[2])).join(preds).map(la
mbda r: (r[1][0]-r[1][1])**2).reduce(lambda a,b: a+b)
    return se/N
```

In [ ]:
```
N = Ratings.count()
evaluate(test,preds,N)
```

2. Does it works for individual user?

In [149]:
```
to_use = msf[['reviewerID','asin','overall','imUrl','categories']]
```

Step 1: Randomly choose an active user (in order to make sure he has make enough reviews to predict)

In [ ]:
```
import graphlab.aggregate as agg
User = msf.groupby(key_columns='reviewerID',
                           operations={'count': agg.COUNT()})

ActiveUser = User.sort('count', ascending = False)
```

In [ ]:
```
# We will choose one from who was rank 20 to 30 in regard of their num
ber of reviews
ActiveUser[20:30]
```

In [151]:
```
# For example, we will choose 'reviewerID' == 'A2YJ8VP1SSHJ7' and we c
an extract what books has he made reviews
UserBook = to_use.filter_by('A2YJ8VP1SSHJ7','reviewerID')
# We can also choose other users
#UserBook = msf.filter_by('ANSX922QNYA67','reviewerID')
#UserBook = msf.filter_by('A1DA6E4FNRSAWN','reviewerID')
```

Step 2: Find the top 10 books he has given high rates

In [153]:
```
UserRate = UserBook.sort('overall', ascending = False)
UserRate10 = UserRate[1:10]
UserRate10
```

Out[153]:

reviewerID	asin	overall	imUrl	catego
A2YJ8VP1SSHJ7	B0027VXV7Y	5.0	http://ecx.images-amazon.com/images/I/513GN71u...	[[Boo Literat & Fictio [Books
A2YJ8VP1SSHJ7	B002HJ1WS6	5.0	http://ecx.images-amazon.com/images/I/5195xNx8...	[[Boo Literat & Fictio Unite States
A2YJ8VP1SSHJ7	B002R5B0WI	5.0	http://ecx.images-amazon.com/images/I/514jHoot...	[[Boo Literat & Fictio [Bool Sci
A2YJ8VP1SSHJ7	B0030CMJEK	5.0	http://ecx.images-amazon.com/images/I/51gn4USN...	[[Boo Literat & Fictio Erotica
A2YJ8VP1SSHJ7	B0030H269S	5.0	http://ecx.images-amazon.com/images/I/51DTPmD5...	[[Boo Gay Lesbi Literat & Fict ...
A2YJ8VP1SSHJ7	B003370JHG	5.0	http://ecx.images-amazon.com/images/I/51l3GNZX...	[[Boo Literat & Fictio [Books
A2YJ8VP1SSHJ7	B0039NMTFO	5.0	http://ecx.images-amazon	[[Boo Literat

			com/images/I/51M39PxE ...	Literat & Fictio [Books
A2YJ8VP1SSHJ7	B003AKY45Y	5.0	http://ecx.images-amazon. com/images/I/51wM80FP ...	[[Boo Literat & Fictio [Books
A2YJ8VP1SSHJ7	B003RRYC8Y	5.0	http://ecx.images-amazon.	[[Boo Literat

In [154]:
```
# See categories of those books
UserCategory = UserRate10[['categories']]
UserCategory[0]
UserCategory[1]
UserCategory[4]
# It seems that this user's interests are concentrated on the "romance story/fiction story". When we choose other users
# we can also see that large amount of users' interests are concentrated in one or two areas. That's why we can recommend
# books based on their previous reviews.
```

Out[154]:
```
{'categories': [['Books', 'Gay & Lesbian', 'Literature & Fiction', '
Erotica'],
  ['Books',
   'Literature & Fiction',
   'Anthologies & Literary Collections',
   'General'],
  ['Books', 'Literature & Fiction', 'Erotica'],
  ['Books', 'Romance', 'Contemporary'],
  ['Books', 'Romance', 'Lesbian Romance'],
  ['Kindle Store',
   'Kindle eBooks',
   'Literature & Fiction',
   'Anthologies & Literature Collections'],
  ['Kindle Store',
   'Kindle eBooks',
   'Literature & Fiction',
   'Contemporary Fiction',
   'Romance'],
  ['Kindle Store',
   'Kindle eBooks',
   'Literature & Fiction',
   'Erotica',
   'LGBT',
   'Lesbian'],
  ['Kindle Store',
   'Kindle eBooks',
   'Literature & Fiction',
   'Genre Fiction',
   'Gay & Lesbian',
   'Lesbian'],
  ['Kindle Store', 'Kindle eBooks', 'Romance', 'Lesbian Romance']]}
```

In [155]:
```
# We can also show the cover of those books to see them more directly.
def showProductImages(url_sarray):
    image_sarray = url_sarray.apply(lambda x: gl.Image(x))
    gl.canvas.set_target('ipynb')
    image_sarray.show()
showProductImages(UserRate10['imUrl'])
```

```
Downloading http://ecx.images-amazon.com/images/I/513GN71uBvL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/891f9bef-a66f-47ab-87be-648881979241.jpg

Downloading http://ecx.images-amazon.com/images/I/5195xNx8wpL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/3001b37b-aa04-4557-8618-e2ec81bf5ac8.jpg
```

Downloading http://ecx.images-amazon.com/images/I/514jHootN6L._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/5e81c086-8442-4e28-9000-2f7e652458df.jpg

Downloading http://ecx.images-amazon.com/images/I/51gn4USNCHL._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/10e86a88-0fbc-45c9-869b-2c4d3602f7e6.jpg

Downloading http://ecx.images-amazon.com/images/I/51DTPmD5t0L._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/c4b29580-c149-49c4-8533-9c73363a6f83.jpg

Downloading http://ecx.images-amazon.com/images/I/51l3GNZXeEL._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/5debce4d-c655-409f-bd6f-aa17444352f1.jpg

Downloading http://ecx.images-amazon.com/images/I/51M39PxEHGL._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/e5ff3205-6173-4329-870a-fd1749d294ad.jpg

Downloading http://ecx.images-amazon.com/images/I/51wM80FPl5L._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/4fa22a6c-e272-4a39-a60e-f63ce7e6eb0c.jpg

Downloading http://ecx.images-amazon.com/images/I/51OcXeRwEsL._BO2,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/927/ba93eb2c-74ef-4902-8401-4b1d17f22859.jpg

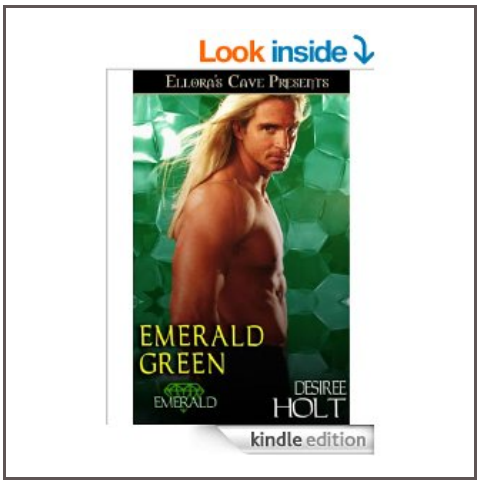
All 9 images in <*SArray*>


Look inside
nocturne BITES
DEMON SEDUCTION
Lisa Renee Jones
kindle edition

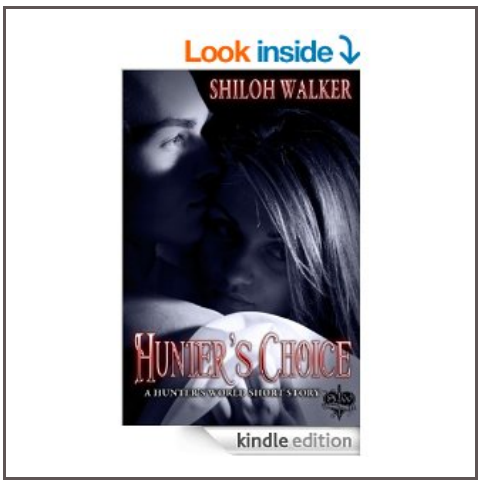

Look inside
nocturne
SENTINELS
Lion Heart
DORANNA DURGIN
BOOK 2
kindle edition


Look inside
CHRISTOPHER BUECHELER
THE BLOOD THAT BONDS
kindle edition


Look inside
Ellora's Cave Moderne
Beg for It
MINX MALONE
kindle edition


Look inside
Ellora's Cave Presents
EMERALD GREEN
EMERALD
DESIREE HOLT
kindle edition


Look inside
Ellora's Cave Again
TAMING THE Beast
EVANGELINE ANDERSON
kindle edition



Step 3: Find the books we recommond to him

In [131]:
```
# Load the item and user matrix
ItemMatrix = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5
-grp9/data/part-00011', delimiter=',', header=False)
UserMatrix = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5
-grp9/data/part-00011', delimiter=',', header=False)
ItemMap = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5-gr
p9/data/item_map.csv', delimiter=',', header=True)
UserMap = gl.SFrame.read_csv('/Users/yijiapan/Desktop/Spr2017-proj5-gr
p9/data/user_map.csv', delimiter=',', header=True)
```

```
Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/part-00011

Parsing completed. Parsed 100 lines in 0.624574 secs.
```

```
------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[float,float,float,float,float,float,float,float,f
loat,float,float,float,float,float,float,float,float,float,float,flo
at,float,float,float,float,float,float,float,float,float,float,float
,float,float,float,float,float,float,float,float,float,float,float,f
loat,float,float,float,float,float,float,float,float]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/part-00011

Parsing completed. Parsed 35840 lines in 0.797602 secs.

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/part-00011

Parsing completed. Parsed 100 lines in 0.529489 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[float,float,float,float,float,float,float,float,f
loat,float,float,float,float,float,float,float,float,float,float,flo
at,float,float,float,float,float,float,float,float,float,float,float
,float,float,float,float,float,float,float,float,float,float,float,f
loat,float,float,float,float,float,float,float,float]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/part-00011

Parsing completed. Parsed 35840 lines in 0.794367 secs.

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/item_map.csv

Parsing completed. Parsed 100 lines in 0.363553 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[str,int]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------
```

```
Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/item_map.csv

Parsing completed. Parsed 430520 lines in 0.324587 secs.

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/user_map.csv

Parsing completed. Parsed 100 lines in 1.73675 secs.

------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[str,int]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/yijiapan/Desktop/Spr2017-proj5-grp9/dat
a/user_map.csv

Parsing completed. Parsed 1406710 lines in 1.19459 secs.
```

In [158]:
```
# Find u
urow = UserMap.filter_by('A2YJ8VP1SSHJ7','reviewerID')
unum = urow[["user"]]
```

In [159]:
```
unum
```

Out[159]:

user
400529

[1 rows x 1 columns]

In [161]:
```
u = UserMatrix.filter_by(400529.0,'X1')
```

In [105]:
```
# Find the predict rates of this user
ItemMatrix2 = ItemMatrix.to_dataframe().values
UserMatrix2 = u.to_dataframe().values
```

In [109]:
```
PredsRateMatrix = UserMatrix2.dot(ItemMatrix2.T)
```

In [231]:
```
First10_book = np.argsort(PredsRateMatrix.reshape((35840,)))[::-1][:10
]
```

In [228]:
```
#J_pred_bad = np.argsort(PredsRateMatrix.reshape((35840,)))[:10]
```

In [163]:
```
First10_bookmap = ItemMap.filter_by(First10_book,"item")
First10_bookasin = First10_bookmap[['asin']]
```

In [236]:
```
#First10_bookmap_bad = ItemMap.filter_by(J_pred_bad,"item")
#First10_bookasin_bad = First10_bookmap_bad[['asin']]
```

In [166]:
```
PredsRate10 = to_use.join(First10_bookasin,how="inner",on="asin")
```

In [237]:
```
#PredsRate10_bad = to_use.join(First10_bookasin_bad,how="inner",on="as
in")
```

In [245]:
```
# Find the category of those recommend books
PredsRate10.unique()[['categories']][5]
```

Out[245]:
```
{'categories': [['Books', 'Literature & Fiction', 'United States'],
  ['Books', 'Romance', 'Contemporary'],
  ['Kindle Store', 'Kindle eBooks', 'Romance']]}
```

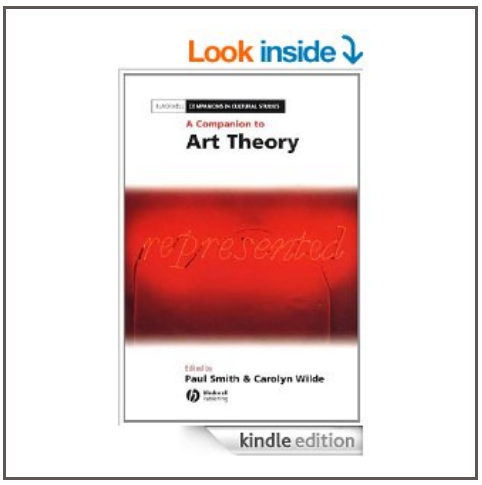
In [238]:
```
# Show the cover of those books to see them more directly.
def showProductImages(url_sarray):
    image_sarray = url_sarray.apply(lambda x: gl.Image(x))
    gl.canvas.set_target('ipynb')
    image_sarray.show()
showProductImages(PredsRate10['imUrl'].unique())
```

```
Downloading http://ecx.images-amazon.com/images/I/41tIDgGgQjL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/c3f32d2e-08a4-4388-a7d3-29f410e58f95.jpg

Downloading http://ecx.images-amazon.com/images/I/41Cr5nU4ajL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/0839728f-065b-45c1-9f80-ed44086c7ee7.jpg

Downloading http://ecx.images-amazon.com/images/I/51-awec9vNL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/67ee57d2-48bc-4249-bda5-2a3a0f9073dd.jpg

Downloading http://ecx.images-amazon.com/images/I/41NKyPsrgvL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/fc6120da-445c-4e81-a124-ef5e0927f753.jpg
```

```
Downloading http://ecx.images-amazon.com/images/I/51KEn-I%2B1AL._BO2
,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4
,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan
/927/862045c5-5ac4-4905-8b32-70602a1ba864.jpg

Downloading http://ecx.images-amazon.com/images/I/412B5iF2PzL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/731ac7d5-d312-4f53-a9a2-85e2c0d48494.jpg

Downloading http://ecx.images-amazon.com/images/I/513ppuaNvpL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/330ac986-8805-4279-b7d5-a4f34aab84dc.jpg

Downloading http://ecx.images-amazon.com/images/I/51boJyIGi%2BL._BO2
,204,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4
,BottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan
/927/767e803e-605a-4638-b0c0-4e5a6346e59c.jpg

Downloading http://ecx.images-amazon.com/images/I/51ZlNLsjbyL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/0d5d4c81-83c7-47ee-a8ac-d6f242953d74.jpg

Downloading http://ecx.images-amazon.com/images/I/41foZmhNNrL._BO2,2
04,203,200_PIsitb-sticker-v3-big,TopRight,0,-55_SX278_SY278_PIkin4,B
ottomRight,1,22_AA300_SH20_OU01_.jpg to /var/tmp/graphlab-yijiapan/9
27/89b80277-f98e-4129-abfa-6cd4f2aa2eb6.jpg
```

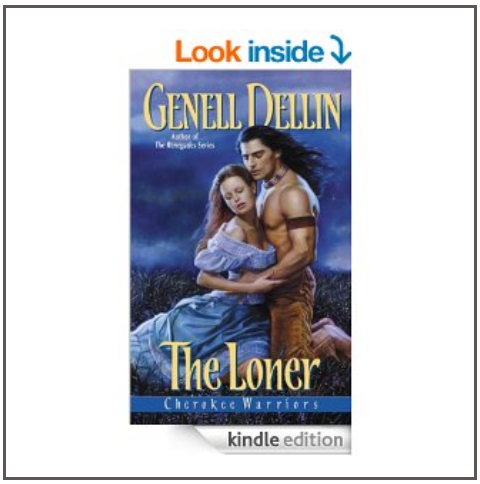
All 10 images in <*SArray*>

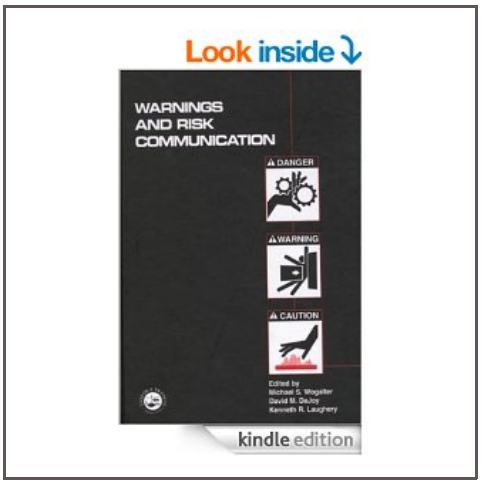

Look inside
CHEWING
GUM
THE FORTUNES OF TASTE
MICHAEL REGLIFT
kindle edition

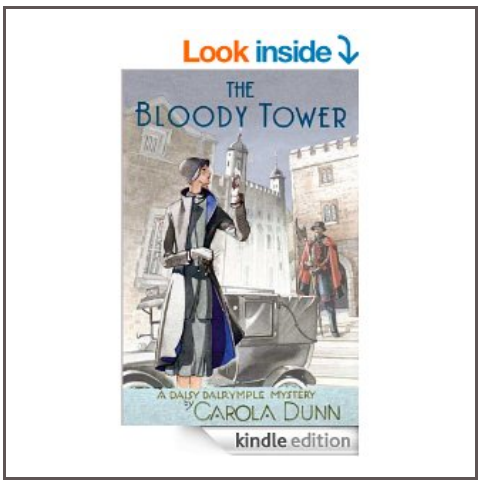

Look inside
A Companion to
Art Theory
represented
Paul Smith & Carolyn Wilde
kindle edition

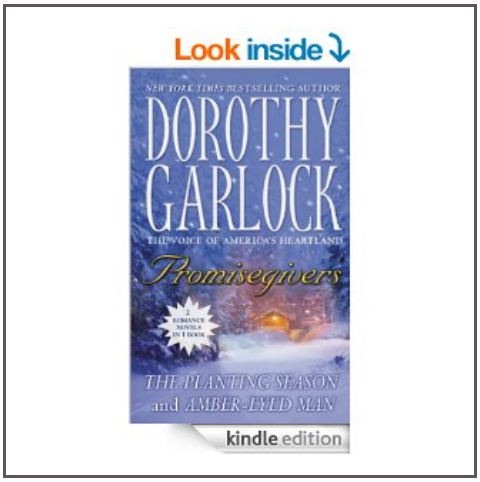

Look inside
Romance
BARBARA
HANNAY
The Bridesmaid's Best Man
Special Treat!
kindle edition

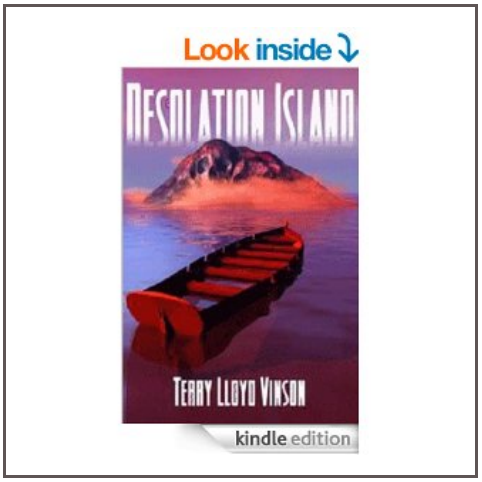

Look inside
EVERLASTING LOVE
THE VOW
REBECCA WINTERS
kindle edition


Look inside
GENELL DELLIN
The Loner
Cherokee Warriors
kindle edition


Look inside
WARNINGS
AND RISK
COMMUNICATION
DANGER
WARNING
CAUTION
kindle edition







Plot for two dimensions

In [279]:
```
# UserMatrix2
UserMatrix2
Topic2 = np.argsort(UserMatrix2.reshape((51,)))[::-1][:3]
```

In [281]:
```
Topic2
```

Out[281]: array([0, 11, 40])

In [199]:
```
UserRate10.__class__
```

Out[199]: graphlab.data_structures.sframe.SFrame

In [206]:
```
J_real = ItemMap.join(UserRate10,how="inner",on="asin")['item']
```

In [223]:
```
J_real = np.array([17206,32587,62927,69306])
```

In [282]:
```
to_plot = np.hstack((np.vstack((ItemMatrix2[:5,:],ItemMatrix2[-5:,:]))
,np.array([1,1,1,1,1,-1,-1,-1,-1,-1]).reshape((10,1))))
```

In [283]:
```
u = np.hstack((UserMatrix2,np.array((0)).reshape((1,1))))
```

In [285]:
```
to_plot = np.vstack((u,to_plot))
```

In [286]:
```
to_plot = to_plot[:,np.array([11,40,51])]
```

In [224]:
```
ItemMatrix2[J_real,:]
```

```
---------------------------------------------------------------------
-------
IndexError                                Traceback (most recent cal
l last)
<ipython-input-224-b359698da5d4> in <module>()
----> 1 ItemMatrix2[J_real,:]

IndexError: index 62927 is out of bounds for axis 0 with size 35840
```

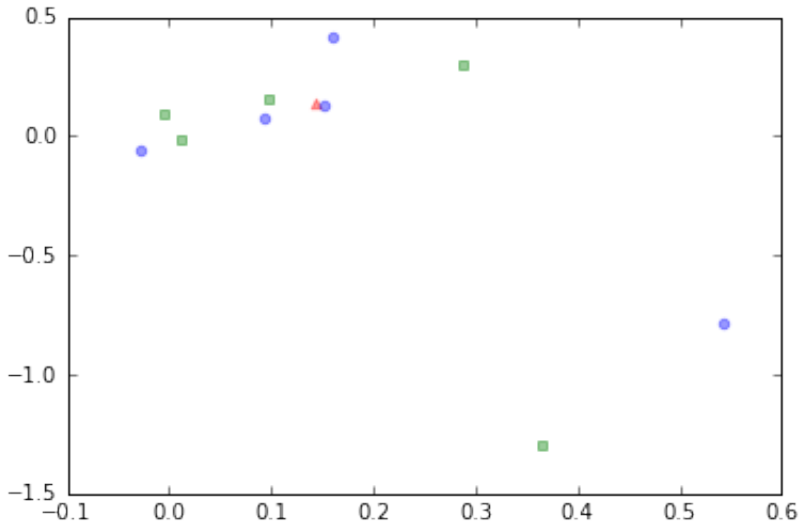
In [ ]:
```
#vTwo2 = PredsRate10trans.filter_by(Topic2,"item")
```

In [25]:
```
#from mpl_toolkits.axes_grid.axislines import SubplotZero

#if 1:
    #fig = plt.figure(1)
    #ax = SubplotZero(fig, 111)
    #fig.add_subplot(ax)

    #for direction in ["xzero", "yzero"]:
        #ax.axis[direction].set_axisline_style("-|>")
        #ax.axis[direction].set_visible(True)

    #for direction in ["left", "right", "bottom", "top"]:
        #ax.axis[direction].set_visible(False)

    #x = np.linspace(-1.2, 1.2, 2)
    #ax.plot(x,x,'w')

    # The axis texts
    #ax.text(1.5,-0.2,'realistic',fontsize=15)
    #ax.text(-2,-0.2,'unrealistic',fontsize=15)
    #ax.text(-0.2,1.7,'serious',fontsize=15)
    #ax.text(-0.2,-1.7,'funny',fontsize=15)
    #ax.text(-1.23,2.3,'Individual Recommendation', fontsize = 21, fon
tweight = 'bold')
    #ax.plot(vTwo1[0],vTow1[1],'ro')
    #ax.plot(vTwo2[0],vTow2[1],'bs')

    # The vector of books
    #ax.text(v[0][0],v[0][1],'book1',fontsize=12,bbox={'facecolor':'ye
llow','alpha':0.5,'pad':4})
    #ax.text(v[1][0],v[1][1],'book2',fontsize=12,bbox={'facecolor':'ye
llow','alpha':0.5,'pad':4})

    # The vector of users
    #ax.text(u[0],u[1],'user1',fontsize=12,bbox={'facecolor':'green','
alpha':0.2,'pad':4})
    #ax.text(u[1][0],u[1][1],'user2',fontsize=12,bbox={'facecolor':'gr
een','alpha':0.2,'pad':4})

    #plt.show()
```

```
  File "<ipython-input-25-384576553c35>", line 2
    u1=u.sort(u)
              ^
SyntaxError: invalid syntax
```

In [292]:
```
%matplotlib inline
fig, ax = plt.subplots()
ax.scatter(to_plot[0,0], to_plot[0,1], color='r', marker='^', alpha=.4
)
ax.scatter(to_plot[1:6,0], to_plot[1:6,1], color='b', alpha=.4)
ax.scatter(to_plot[6:,0], to_plot[6:,1], color='g', marker='s', alpha=
.4)
```

Out[292]: <matplotlib.collections.PathCollection at 0x1275c38d0>



In [ ]: