Table of Contents:

Importing libraries

In [1]:
```
import pandas as pd
import pickle as pkl
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.metrics.pairwise import cosine_similarity
import sqlalchemy as sa
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
import nltk.data
%matplotlib inline
```

Installing libraries with appropriate versions (if missing)

In [ ]:
```
import pip
def install(package):
   pip.main(['install', package])

install('sklearn')
```

In [ ]:
```
import sklearn
sklearn.__version__
```

Importing data

If the data is not too large, then loading it all in at once is a good idea.

If the data is very large, then streaming from a database (or a file) is a good idea. The gensim methods allow to use an iterator which can read from a file/db.

Reading data from database

In [ ]:
```
%%time

class ReviewStream(object):
    def __init__(self, dbname, table, column):
        self.dbname = dbname
        self.table = table
        self.column = column
        self.engine = sa.create_engine("sqlite:///{}".format(self.d
bname))
        self.stemmer=PorterStemmer()
        self.stop = set(stopwords.words('english') + list(string.pu
nctuation))
        self.sent_tokenizer = nltk.data.load('nltk:tokenizers/punkt
/english.pickle')
        return

    def w_tokenize(self, sent):
        #print('sent', [w for w in word_tokenize(sent)])
        #return [self.stemmer.stem(w) for w in word_tokenize(sent)
if not w in self.stop]
        return [self.stemmer.stem(word) for word in nltk.tokenize.w
ord_tokenize(sent) if word.isalnum() and not word in self.stop]

    def sent_process(self, text):
        raw_sentences = sent_tokenizer.tokenize(text.strip())
        return [self.w_tokenize(sent) for sent in raw_sentences if
len(sent) > 0]

    def __iter__(self):
        sql = "select beer_id, user_id, r_text from reviews limit -
1".format(self.column, self.table)
        #sql = "select beer_id, user_id, r_text from reviews".forma
t(self.column, self.table)
        for i, (beer_id, user_id, text,) in enumerate(self.engine.e
xecute(sql)):
            #for sent_idx, sent in enumerate(self.sent_process(text
)):
            if i % 100000 == 0:
                print(i)
            yield {'beer_id': beer_id, 'user_id': user_id,
                   #'sent_idx': sent_idx,
                   'text': self.w_tokenize(text)}

dbname, table, column  = "beeradvocate.db", "reviews", "r_text"
rs = ReviewStream(dbname, table, column)
df = pd.DataFrame(r for r in rs)
```

In [ ]:
```
%%time

def read_data(fname):
    if fname.endswith('.db'):
        sql = "select beer_id, user_id, r_text from reviews limit -
1"
        engine = sa.create_engine("sqlite:///" + dbname)
        conn = engine.connect()
        res = [r.values() for r in conn.execute(sql).fetchall()]

    elif fname.endswith('.csv') or fname.endswith('.csv.gz'):
        df = pd.read_csv(fname, engine='c', compression='infer', me
mory_map=True)
        df['r_text'].replace('', np.nan, inplace=True)
        df.dropna(inplace=True)
        stop = set(stopwords.words('english') + list(string.punctua
tion))
        stemmer=PorterStemmer()
        def _w_tokenize(sent):
            return [stemmer.stem(word) for word in nltk.tokenize.wo
rd_tokenize(sent) if word.isalnum() and not word in stop]

        df['text_process'] = df['r_text'].apply(lambda x: _w_tokeni
ze(x))
        df.to_pickle('r_process_all.pkl')
        return df
    elif fname.endswith('.pkl'):
        df = pd.read_pickle(fname)
        return df
    return None

#df = read_data("out.csv.gz")
#df = read_data('beeradvocate.db')
df.to_pickle('r_process_all.pkl')
```

Reading data from pickle

In [2]:
```
%%time

df = pd.read_pickle('r_process_all.pkl')
df= df[df.astype(str)['text'] != '[]']
print(len(df))
```

```
1518116
CPU times: user 7min 58s, sys: 11.8 s, total: 8min 10s
Wall time: 8min 10s
```

In [3]:
```
df.head()
```

Out[3]:

	beer_id	text	user_id
0	47986	[lot, foam, lot, smell, banana, lactic, tart, ...	stcules
1	48213	[dark, red, color, light, beig, foam, averag, ...	stcules
2	48215	[almost, total, black, beig, foam, quit, compa...	stcules
3	47969	[golden, yellow, color, white, compact, foam, ...	stcules
4	64883	[accord, websit, style, caldera, cauldron, cha...	johnmichaelsen

Word2Vec

Gensim iterator

In [4]:
```
class RGen(object):
    def __iter__(self):
        for user_id, g in df.groupby('user_id', sort=False):
            words = np.hstack(g.text.values).tolist()
            #if words:
            yield words
#print(list(RGen()))
```

Word2Vec with GoogleNews

In [5]:
```
%%time

import gensim
from gensim.models import Word2Vec

num_features = 300    # Word vector dimensionality
min_word_count = 5   # Minimum word count
num_workers = 4       # Number of threads to run in parallel
context = 10          # Context window size
downsampling = 1e-3

print("Training model...")
#Training the initial model from our data
ggl_model = Word2Vec(RGen(), workers=num_workers, size=num_features
, min_count = min_word_count, window = context, sample = downsampli
ng)
ggl_model.intersect_word2vec_format('GoogleNews-vectors-negative300
.bin.gz', binary=True)

# ggl_model.save('gglmodel.model')
```

```
CPU times: user 112 ms, sys: 30.2 ms, total: 142 ms
Wall time: 136 ms
```

Loading saved ggl model

In [6]:
```
from gensim.models import Word2Vec

ggl_model = Word2Vec.load('gglmodel.model')
ggl_model.most_similar('stout')
```

Out[6]:
```
[(u'stouter', 0.5638942718505859),
 (u'stingy', 0.5373303890228271),
 (u'beefy', 0.46627628803253174),
 (u'smallish', 0.461425244808197),
 (u'tall', 0.433956116437912),
 (u'stolid', 0.4323907792568207),
 (u'strong', 0.4257262349128723),
 (u'potent', 0.4184947609901428),
 (u'stiff', 0.4169541597366333),
 (u'solid', 0.41479194164276123)]
```

Testing Context Size

In [7]:
```
%%time

m5 = Word2Vec(df.text.values, workers=8, size=300, min_count = min_
word_count, window = 5, sample = downsampling)
m10 = Word2Vec(df.text.values, workers=8, size=300, min_count = min
_word_count, window = 10, sample = downsampling)
m15 = Word2Vec(df.text.values, workers=8, size=300, min_count = min
_word_count, window = 15, sample = downsampling)
```

```
CPU times: user 1h 58min 42s, sys: 59.9 s, total: 1h 59min 42s
Wall time: 16min 36s
```

In [8]:
```
#m10.predict_output_word(['best', 'beer'])
m10.most_similar('amstel')
```

Out[8]:
```
[(u'heini', 0.6900782585144043),
 (u'heineken', 0.6871074438095093),
 (u'hein', 0.6815861463546753),
 (u'keyston', 0.6695038080215454),
 (u'beck', 0.6602358818054199),
 (u'tecat', 0.6592646837234497),
 (u'busch', 0.6506742238998413),
 (u'corona', 0.6466759443283081),
 (u'heinekin', 0.6382101774215698),
 (u'coor', 0.6217682361602783)]
```

Word2vec on each reviews

Train Word2vec models

In [ ]:
```
%%time

w2vmodels_r = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    m = Word2Vec(df.text.values, workers=8, size=size, min_count =
min_word_count, window = context, sample = downsampling)
    m.save('model_{}.w2vrmodel'.format(size))
    w2vmodels_r[size] = m
```

Load saved Word2vec models

In [9]:
```
%%time

w2vmodels_r = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    w2vmodels_r[size] = Word2Vec.load('model_{}.w2vrmodel'.format(s
ize))  #w2vmodel

print(w2vmodels_r)
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
{100: <gensim.models.word2vec.Word2Vec object at 0x10ead8310>, 100
0: <gensim.models.word2vec.Word2Vec object at 0x374d90750>, 10: <g
ensim.models.word2vec.Word2Vec object at 0x10ef1a7d0>, 300: <gensi
m.models.word2vec.Word2Vec object at 0x3458a4a10>, 50: <gensim.mod
els.word2vec.Word2Vec object at 0x34114b2d0>, 500: <gensim.models.
word2vec.Word2Vec object at 0x36ca36950>}
CPU times: user 1.43 s, sys: 615 ms, total: 2.05 s
Wall time: 1.99 s
```

Similar words with varying dimension

In [10]:
```
for s in w2vmodels_r.keys():
    print(s, w2vmodels_r[s].most_similar('stout', topn=3))
```

```
(100, [(u'porter', 0.8498043417930603), (u'ri', 0.7985371351242065
), (u'stouti', 0.5629050731658936)])
(1000, [(u'porter', 0.756035327911377), (u'ri', 0.6556090712547302
), (u'stouti', 0.46970367431640625)])
(10, [(u'porter', 0.9837263822555542), (u'ri', 0.9442464113235474)
, (u'oatmeal', 0.9263836145401001)])
(300, [(u'porter', 0.7941805124282837), (u'ri', 0.702453076839447)
, (u'stouti', 0.49984514713287354)])
(50, [(u'porter', 0.9247586727142334), (u'ri', 0.8238557577133179)
, (u'stouti', 0.6712391376495361)])
(500, [(u'porter', 0.773469865322113), (u'ri', 0.6763601899147034)
, (u'stouti', 0.47444769740104675)])
```

Word2vec on concat reviews

In [11]:
```
df2 = df.groupby('user_id').agg({'text': 'sum'})
df2['user_id'] = df2.index
```

Train Word2vec models

In [ ]:
```
%%time
w2vmodels_c = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    m = Word2Vec(df2.text.values, workers=8, size=size, min_count =
min_word_count, window = context, sample = downsampling)
    m.save('model_{}.w2vmodels_c'.format(size))
    w2vmodels_c[size] = m
```

Load Word2vec models

In [12]:
```
%%time

w2vmodels_c = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    w2vmodels_c[size] = Word2Vec.load('model_{}.w2vmodels_c'.format
(size))  #w2vmodel

print(w2vmodels_c)
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
{100: <gensim.models.word2vec.Word2Vec object at 0x4ad377890>, 100
0: <gensim.models.word2vec.Word2Vec object at 0x4bed5f310>, 10: <g
ensim.models.word2vec.Word2Vec object at 0x34114b590>, 300: <gensi
m.models.word2vec.Word2Vec object at 0x4b040e750>, 50: <gensim.mod
els.word2vec.Word2Vec object at 0x34114b510>, 500: <gensim.models.
word2vec.Word2Vec object at 0x4b5ed43d0>}
CPU times: user 5.92 s, sys: 725 ms, total: 6.64 s
Wall time: 6.55 s
```

Similar words with varying dimension

In [13]:
```
for s in w2vmodels_c.keys():
    print(s, w2vmodels_c[s].most_similar('stout', topn=3))
```

```
(100, [(u'porter', 0.7773493528366089), (u'ri', 0.7481129765510559
), (u'mackeson', 0.6159349083900452)])
(1000, [(u'porter', 0.6898748278617859), (u'ri', 0.638052105903625
5), (u'stouti', 0.5087007284164429)])
(10, [(u'porter', 0.9506092071533203), (u'ri', 0.9290263652801514)
, (u'impi', 0.9197987914085388)])
(300, [(u'porter', 0.7092608213424683), (u'ri', 0.6778067946434021
), (u'mackeson', 0.5336452722549438)])
(50, [(u'porter', 0.8475256562232971), (u'ri', 0.7952302098274231)
, (u'wrassler', 0.6839162111282349)])
(500, [(u'porter', 0.6939135789871216), (u'ri', 0.6513720750808716
), (u'stouti', 0.5126355290412903)])
```

TFIDF User similarities

TFIDF on review text

In [14]:
```
%%time

user_groups = df.groupby('user_id')

# we have already tokenized and analysed
tfidf_vectorizer = TfidfVectorizer(tokenizer=lambda x: x, analyzer=
lambda x: x)

tfidf_mat_u = tfidf_vectorizer.fit_transform((np.hstack(g.text.valu
es) for (row, g) in user_groups))
```

```
CPU times: user 1min 19s, sys: 1.53 s, total: 1min 20s
Wall time: 1min 20s
```

User indices

In [15]:
```
# handy to have some mappings between the index of the user id, and
the actual user_id (string)
user_idx = {i:user_id for i, (user_id, g) in enumerate(user_groups)
}
user_idx_inv = {v:k for k,v in user_idx.items()}
```

TFIDF most similar users and scores

In [ ]:
```
%%time

def find_similar(tfidf_matrix=None, idx=0, top_n = 5):
    sims = linear_kernel(tfidf_matrix[idx:idx+1], tfidf_matrix).fla
tten()
    related = [i for i in sims.argsort()[::-1] if i != idx]
    return [(idx, sims[idx]) for idx in related][0:top_n]

def _gen(tfidf_matrix=None):
    for i in user_idx.keys():
        if (i%1000==0):
            print i
        res = find_similar(tfidf_matrix=tfidf_mat_u, idx=i, top_n=1
)[0]
        yield {'user_id':user_idx[i], 'tfidf_sim_user':user_idx[res
[0]], 'tfidf_sim': res[1]}

# we create a data frame for the user-user similarities
df_tfidf_sims = pd.DataFrame(_gen(tfidf_matrix=tfidf_mat_u))
```

Read pre-generated scores

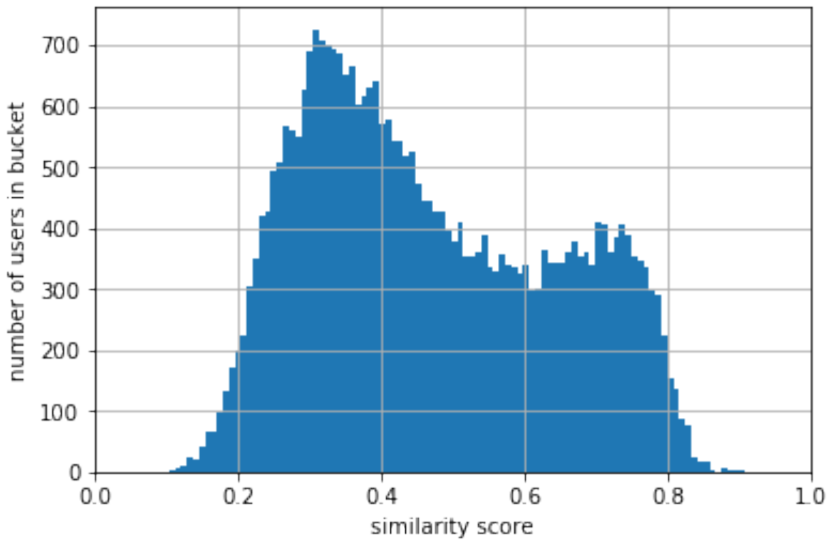
In [16]:
```
# df_tfidf_sims.to_pickle('df_tfidf_sims.pkl')

df_tfidf_sims = pd.read_pickle('df_tfidf_sims.pkl')

# The most similar uses have similarity of ~0.7, which is quite sim
iliar
df_tfidf_sims = df_tfidf_sims.sort_values('tfidf_sim', ascending=Fa
lse)
df_tfidf_sims.head(10)
```

Out[16]:

	tfidf_sim	tfidf_sim_user	user_id
961	0.908219	smartypints	alleykatking
27437	0.908219	alleykatking	smartypints
32850	0.896226	hoppheadipa	zseeanz
13539	0.896226	zseeanz	hoppheadipa
28292	0.894227	stoutman	stevete13
28448	0.894227	stevete13	stoutman
25460	0.884146	stoutman	rootdog316
1302	0.883743	pasdachuri	anthony1
23040	0.883743	anthony1	pasdachuri
14840	0.882520	anthony1	jayrod20

In [17]:
```
# user-user similarity histogram
# most users have similarity of about 0.3 which is pretty much the similarity of random text
df_tfidf_sims.tfidf_sim.hist(bins=100)
plt.xlabel('similarity score')
plt.ylabel('number of users in bucket')
plt.xlim(0,1)
```

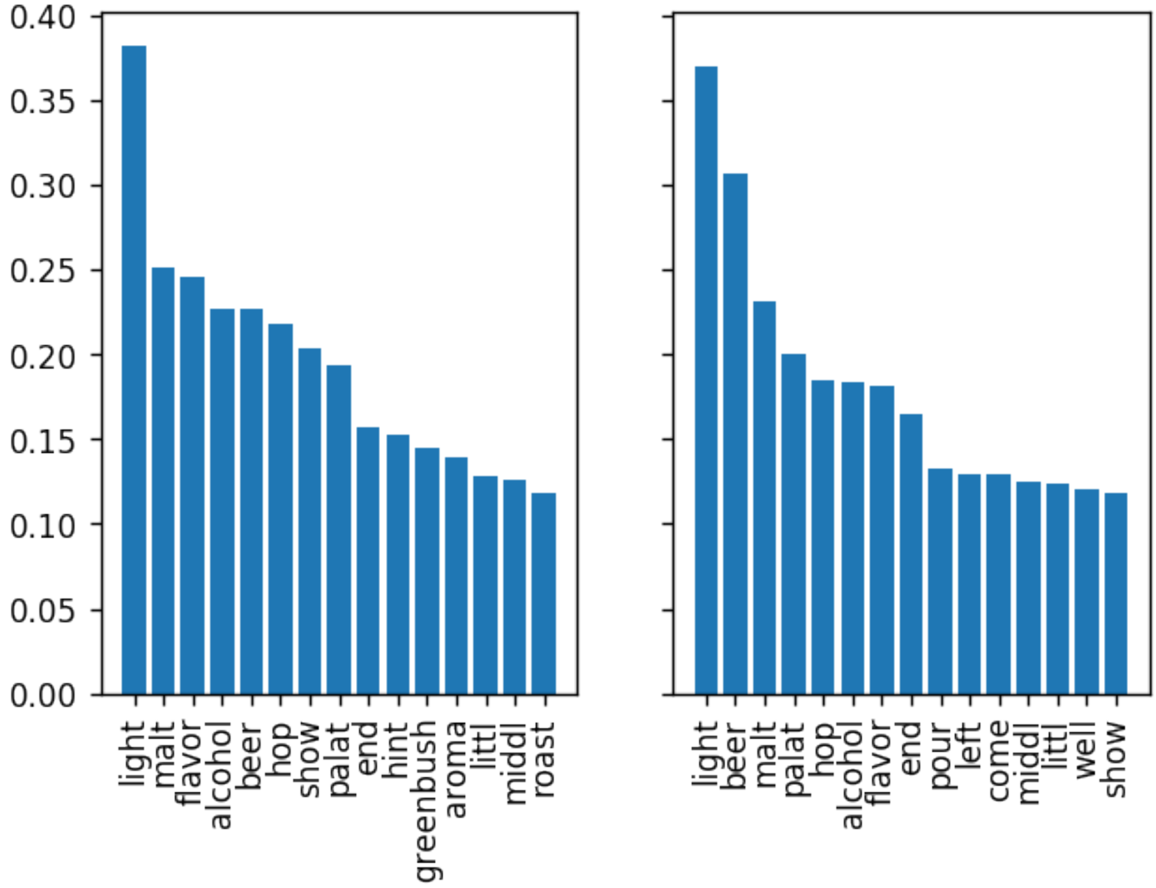
Out[17]: (0, 1)



Sanity check similar users

In [18]:
```
def get_user_user_sim(user_a=None, user_b=None, top_n=10):
    text_a = np.hstack(df[df.user_id == user_a].text.values)
    text_b = np.hstack(df[df.user_id == user_b].text.values)
    #print(text_a, '\n', text_b)
    d_a = pd.Series(tfidf_mat_u.getrow(user_idx_inv[user_a]).toarra
y().flatten(), index = tfidf_vectorizer.get_feature_names()).sort_v
alues(ascending=False)
    d_b = pd.Series(tfidf_mat_u.getrow(user_idx_inv[user_b]).toarra
y().flatten(), index = tfidf_vectorizer.get_feature_names()).sort_v
alues(ascending=False)

    # just to check the sim's are right!
    print('sim:', linear_kernel(tfidf_mat_u[user_idx_inv[user_a]],
tfidf_mat_u).flatten()[user_idx_inv[user_b]])

    fig, axes = plt.subplots(ncols=2, sharey=True)
    ind = np.arange(top_n)
    axes[0].bar(ind, d_a[:top_n].values)
    axes[0].set_xticks(ind)
    axes[0].set_xticklabels(d_a[:top_n].index, rotation=90)

    axes[1].bar(ind, d_b[:top_n].values)
    axes[1].set_xticks(ind)
    axes[1].set_xticklabels(d_b[:top_n].index, rotation=90)

mpl.rcParams['figure.dpi'] = 120

row_idx = 1
#user_a, user_b = 'jcdiflorio', 'extrastout'
user_a, user_b, sim_a_b = df_tfidf_sims.iloc[row_idx].user_id, df_t
fidf_sims.iloc[row_idx].tfidf_sim_user, df_tfidf_sims.iloc[row_idx]
.tfidf_sim
print(user_a, user_b, sim_a_b)
get_user_user_sim(user_a, user_b, top_n=15)
```

```
(u'smartypints', u'alleykatking', 0.90821907494276211)
('sim:', 0.90821907494276211)
```



TFIDF on beer categories

Read beer category data

In [19]:
```
eng = sa.create_engine("sqlite:////Users/cagatay/Desktop/cagatay_th
esis/beeradvocate.db")
sql = "select user_id, beer_style from reviews join beers on beers.
beer_id = reviews.beer_id"

#read into dataframe
u_beer = pd.read_sql(sql,eng)
u_beer.beer_style = u_beer.beer_style.replace({' ': '_'}, regex=Tru
e)
```

TFIDF most similar users and scores

In [ ]:
```
%%time

user_beers = u_beer.groupby('user_id')
user_b_idx = {i:user_id for i, (user_id, g) in enumerate(user_beers
)}

# we have already tokenized and analysed
tfidf_vectorizer = TfidfVectorizer(tokenizer=lambda x: x, analyzer=
lambda x: x)

b_tfidf_mat_u = tfidf_vectorizer.fit_transform((np.hstack(g.beer_st
yle) for (row, g) in user_beers))

def find_similar(tfidf_matrix=None, idx=0, top_n = 5):
    sims = linear_kernel(tfidf_matrix[idx:idx+1], tfidf_matrix).fla
tten()
    related = [i for i in sims.argsort()[::-1] if i != idx]
    return [(idx, sims[idx]) for idx in related][0:top_n]

def _gen(tfidf_matrix=None):
    for i in user_b_idx.keys():
        if (i%1000 ==0):
            print i
        res = find_similar(tfidf_matrix=b_tfidf_mat_u, idx=i, top_n
=1)[0]
        yield {'user_id':user_b_idx[i], 'tfidf_sim_user':user_b_idx
[res[0]], 'tfidf_sim': res[1]}

# we create a data frame for the user-user similarities
df_tfidf_sims_st = pd.DataFrame(_gen(tfidf_matrix=b_tfidf_mat_u))
```

Read pre-generated scores

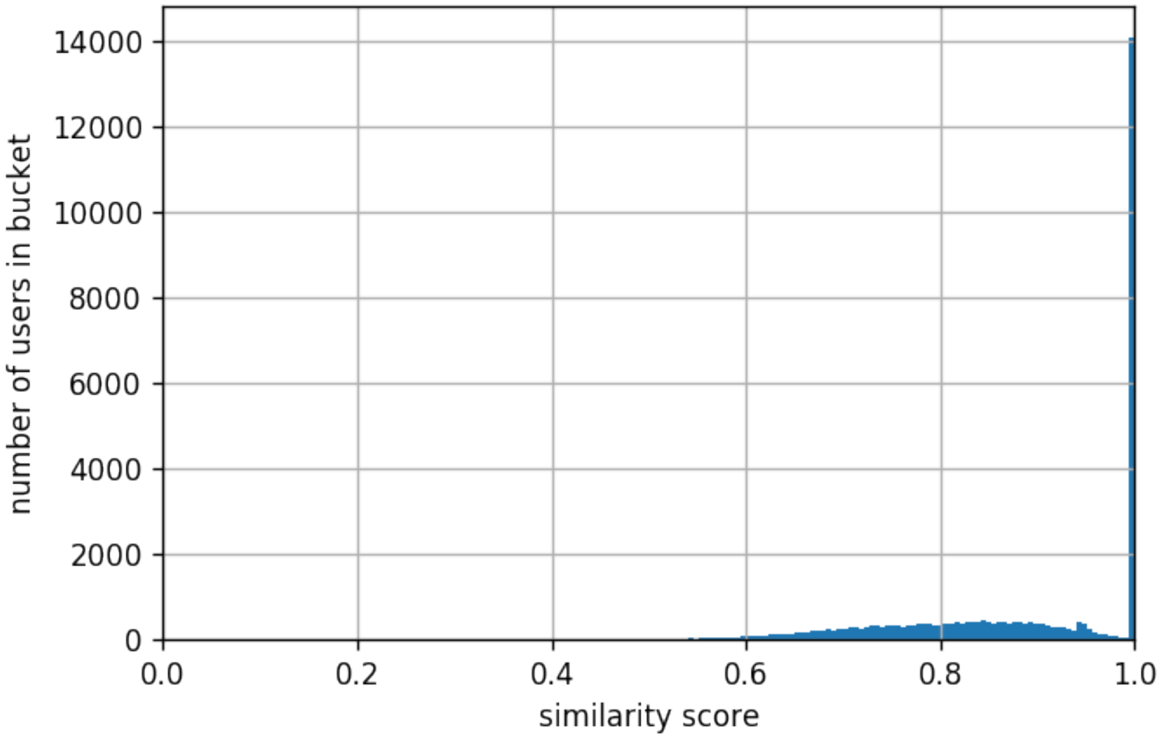
In [20]:
```
# df_tfidf_sims_st.to_pickle('df_tfidf_sims_st.pkl')
df_tfidf_sims_st = pd.read_pickle('df_tfidf_sims_st.pkl')
df_tfidf_sims_st = df_tfidf_sims_st.sort_values('tfidf_sim', ascend
ing=False)
df_tfidf_sims_st.head(10)
```

Out[20]:

	tfidf_sim	tfidf_sim_user	user_id
10094	1.0	downeastjunyah	electric27
31181	1.0	vincentlargo	van55
28653	1.0	downeastjunyah	sundevilstudent
9934	1.0	downeastjunyah	edmundfitzgerald
32255	1.0	downeastjunyah	wornout
22340	1.0	ineedaride	nvcjc
9697	1.0	ineedaride	durandal777
16945	1.0	downeastjunyah	kbrinson
9122	1.0	downeastjunyah	doublebagger
31350	1.0	van55	vincentlargo

In [21]:
```
# user-user similarity histogram
df_tfidf_sims_st.tfidf_sim.hist(bins=100)
plt.xlabel('similarity score')
plt.ylabel('number of users in bucket')
plt.xlim(0,1)
```

Out[21]: (0, 1)



TFIDF on beers

Read beer names data

In [22]:
```
eng = sa.create_engine("sqlite:////Users/cagatay/Desktop/cagatay_thesis/beeradvocate.db")
sql = "select user_id, beer_name from reviews join beers on beers.beer_id = reviews.beer_id"

#read into dataframe
u_beer = pd.read_sql(sql,eng)
u_beer.beer_name = u_beer.beer_name.replace({' ': '_'}, regex=True)
```

TFIDF most similar users and scores

In [32]:
```
%%time

user_beers = u_beer.groupby('user_id')
user_b_idx = {i:user_id for i, (user_id, g) in enumerate(user_beers
)}

# we have already tokenized and analysed
tfidf_vectorizer = TfidfVectorizer(tokenizer=lambda x: x, analyzer=
lambda x: x)

b_tfidf_mat_u = tfidf_vectorizer.fit_transform((np.hstack(g.beer_na
me) for (row, g) in user_beers))

def find_similar(tfidf_matrix=None, idx=0, top_n = 5):
    sims = linear_kernel(tfidf_matrix[idx:idx+1], tfidf_matrix).fla
tten()
    related = [i for i in sims.argsort()[::-1] if i != idx]
    return [(idx, sims[idx]) for idx in related][0:top_n]

def _gen(tfidf_matrix=None):
    for i in user_b_idx.keys():
        #if (i%1000 ==0):
        #    print i
        res = find_similar(tfidf_matrix=b_tfidf_mat_u, idx=i, top_n
=1)[0]
        yield {'user_id':user_b_idx[i], 'tfidf_sim_user':user_b_idx
[res[0]], 'tfidf_sim': res[1]}

# we create a data frame for the user-user similarities
df_tfidf_sims_bn = pd.DataFrame(_gen(tfidf_matrix=b_tfidf_mat_u))
```

```
CPU times: user 28min 26s, sys: 25.6 s, total: 28min 52s
Wall time: 28min 50s
```

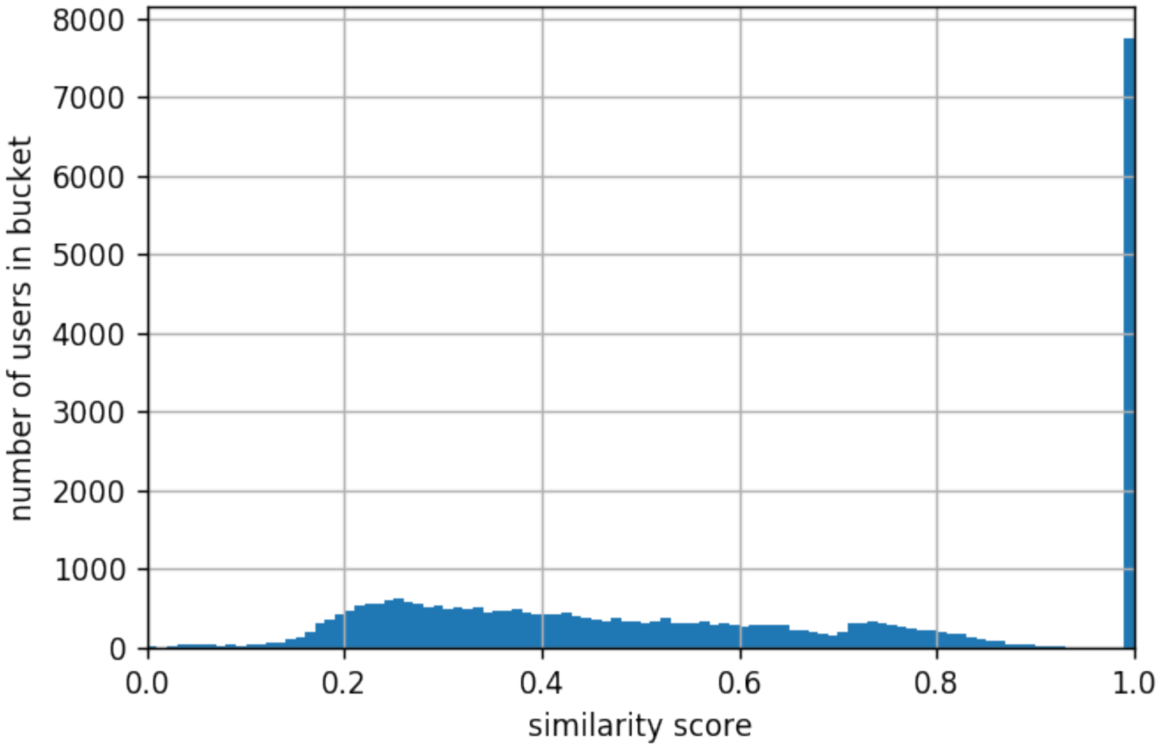
In [33]:
```
df_tfidf_sims_bn.to_pickle('df_tfidf_sims_bn.pkl')
df_tfidf_sims_bn = pd.read_pickle('df_tfidf_sims_bn.pkl')
df_tfidf_sims_bn = df_tfidf_sims_bn.sort_values('tfidf_sim', ascend
ing=False)
df_tfidf_sims_bn.head(10)
```

Out[33]:

	tfidf_sim	tfidf_sim_user	user_id
30819	1.0	vobr0002	tvolt1
31422	1.0	tvolt1	vobr0002
7987	1.0	hopgoddess76	dave097
6906	1.0	rilbert	coltrane4me
27261	1.0	tbow05	skmo
27743	1.0	sme4r	souleman
24770	1.0	rippingstyle	redsahx
2149	1.0	ppibrian	bartzilla
24034	1.0	bartzilla	ppibrian
25118	1.0	redsahx	rippingstyle

In [35]:
```
# user-user similarity histogram
df_tfidf_sims_bn.tfidf_sim.hist(bins=100)
plt.xlabel('similarity score')
plt.ylabel('number of users in bucket')
plt.xlim(0,1)
```

Out[35]: (0, 1)



LSI

In [26]:
```
%%time

from gensim import corpora, models, similarities

print("LSI processing...")
dictionary = corpora.Dictionary(RGen())
dictionary.save('/tmp/lsi.dict')  # store the dictionary, for futur
e reference
print(dictionary)

corpus = [dictionary.doc2bow(text) for text in RGen()]

lsi = models.LsiModel(corpus, id2word=dictionary, num_topics=10)
#lsi.save('/tmp/model.lsi')
```

```
LSI processing...
Dictionary(190991 unique tokens: [u'foufoun', u'woodi', u'woodl',
u'woodm', u'spideri']...)
CPU times: user 4min 11s, sys: 4.86 s, total: 4min 16s
Wall time: 3min 37s
```

In [ ]:
```
%%time
def find_similar(sim_matrix=None, idx=0, top_n = 1):
    score=sorted(sims[idx])[-2]
    related = np.where(sims[idx]==score)[0]
    return [(idx, score) for idx in related][0:top_n]

def _gen(sim_matrix=None):
    for i in user_idx.keys():
        #res = find_similar(sim_matrix=None, idx=i, top_n=1)[0]
        #same users as tfidf sim score
        lsi_sim_user = df_tfidf_sims[df_tfidf_sims['user_id']==user
_idx[i]].tfidf_sim_user.iloc[0]
        res = lsi_sims[i,user_idx_inv[lsi_sim_user]]
        yield {'user_id':user_idx[i], 'lsi_sim_user': lsi_sim_user,
'lsi_sim': res}

lsi_index = similarities.MatrixSimilarity(lsi[corpus]) # transform
corpus to LSI space and index it

lsi_sims = np.array(list(lsi_index))

# we create a data frame for the user-user similarities
df_lsi_sims = pd.DataFrame(_gen(sim_matrix=lsi_sims))
```

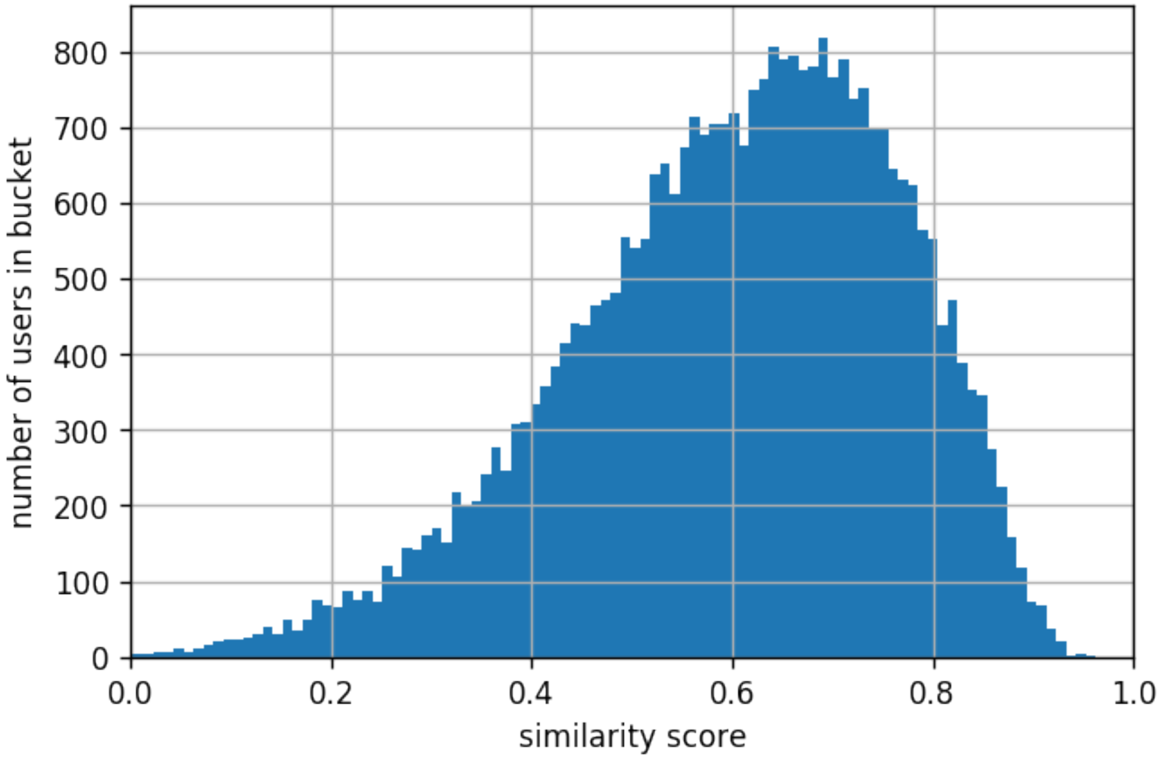
In [36]:
```
# df_lsi_sims.to_pickle('df_lsi_sims')
df_lsi_sims = pd.read_pickle('df_lsi_sims')
df_lsi_sims = df_lsi_sims.sort_values('lsi_sim', ascending=False)
df_lsi_sims.head(5)
```

Out[36]:

	lsi_sim	lsi_sim_user	user_id
20360	0.962881	coachdonovan	misfit1034
15507	0.952370	marvin213	jimboween
11686	0.949211	bbm	geetlord
10073	0.949096	cask1	elcervecero
5752	0.946625	hayes31	cawthonb

In [38]:
```
# user-user similarity histogram
# most users have similarity of about 0.3 which is pretty much the similarity of random text
df_lsi_sims.lsi_sim.hist(bins=100)
plt.xlabel('similarity score')
plt.ylabel('number of users in bucket')
plt.xlim(0,1)
```

Out[38]: (0, 1)



Doc2Vec

Train Doc2Vec beers

In [ ]:
```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument, LabeledS
entence

beer_docs = [TaggedDocument(words=row.text, tags=[row.beer_id]) for
index, row in  df.iterrows()]

beer_models = {}

print("Training doc vectors")
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    m = Doc2Vec(beer_docs, workers=8, size=size, min_count = min_wo
rd_count, window = context, sample = downsampling)
    m.save('d2v_beer_model_{}.model'.format(size))
    beer_models[size] = m
```

Load Doc2Vec beers

In [39]:
```
%%time

from gensim.models.doc2vec import Doc2Vec, TaggedDocument, LabeledS
entence

#This is currently only for d2vmodels (which has multiple tags)
beer_models = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    beer_models[size] = Doc2Vec.load('d2v_beer_model_{}.model'.form
at(size))

print(beer_models)
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
{100: <gensim.models.doc2vec.Doc2Vec object at 0x50e9dc990>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x582c9a450>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x50eddd6d0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x56e0d5590>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x10d8841d0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x57966e4d0>}
CPU times: user 6.82 s, sys: 988 ms, total: 7.8 s
Wall time: 7.69 s
```

In [41]:
```
for s in beer_models.keys():
    print(s, beer_models[s].docvecs.most_similar('246', topn=3))
print(beer_models)
```

```
(100, [(u'449', 0.8820996284484863), (u'567', 0.8763251304626465),
(u'580', 0.8560251593589783)])
(1000, [(u'449', 0.7039251327514648), (u'436', 0.6848270893096924)
, (u'2435', 0.6701992750167847)])
(10, [(u'567', 0.9925632476806641), (u'2435', 0.9854874014854431),
(u'266', 0.9840229749679565)])
(300, [(u'449', 0.7736694812774658), (u'1426', 0.7425159215927124)
, (u'436', 0.7361774444580078)])
(50, [(u'449', 0.9477828145027161), (u'1790', 0.9287354350090027),
(u'567', 0.9273858070373535)])
(500, [(u'449', 0.732648491859436), (u'436', 0.7055214047431946),
(u'1426', 0.6938748359680176)])
{100: <gensim.models.doc2vec.Doc2Vec object at 0x50e9dc990>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x582c9a450>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x50eddd6d0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x56e0d5590>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x10d8841d0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x57966e4d0>}
```

Doc2Vec users

Train Doc2Vec users

In [ ]:
```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument, LabeledS
entence

user_docs = [TaggedDocument(words=row.text, tags=[row.user_id]) for
index, row in  df.iterrows()]

user_models = {}

print("Training doc vectors")
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    m = Doc2Vec(user_docs, workers=8, size=size, min_count = min_wo
rd_count, window = context, sample = downsampling)
    m.save('d2v_user_model_{}.model'.format(size))
    user_models[size] = m
```

Load Doc2Vec users

In [42]:
```
%%time

#This is currently only for d2vmodels (which has multiple tags)
user_models = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    user_models[size] = Doc2Vec.load('d2v_user_model_{}.model'.form
at(size))

print(user_models)
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
{100: <gensim.models.doc2vec.Doc2Vec object at 0x5dedd70d0>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x5f10c1e90>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x51060d9d0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x5e134c050>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x5dedcdfd0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x5eb152ed0>}
CPU times: user 6.75 s, sys: 885 ms, total: 7.63 s
Wall time: 7.52 s
```

In [46]:
```
for s in user_models.keys():
    print(s, user_models[s].docvecs.most_similar('beerfoolish', top
n=3))
print(user_models)
```

```
(100, [(u'thyde606', 0.93559730052948), (u'tootall5', 0.9315415620
803833), (u'shooterco', 0.9273054599761963)])
(1000, [(u'thyde606', 0.8885596990585327), (u'brewmenace', 0.82828
74822616577), (u'shaman2788', 0.8185030817985535)])
(10, [(u'rush2112', 0.9941115975379944), (u'chadsexington1', 0.993
8574433326721), (u'kallay', 0.9929192662239075)])
(300, [(u'thyde606', 0.8899272084236145), (u'shaman2788', 0.866809
606552124), (u'thefileclerk', 0.8628619909286499)])
(50, [(u'thyde606', 0.9573890566825867), (u'bricksandivey', 0.9566
80417060852), (u'lecithin', 0.956180989742279)])
(500, [(u'thyde606', 0.898536205291748), (u'goodnamestaken', 0.853
5416722297668), (u'wtfuggles', 0.8531938791275024)])
{100: <gensim.models.doc2vec.Doc2Vec object at 0x5dedd70d0>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x5f10c1e90>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x51060d9d0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x5e134c050>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x5dedcdfd0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x5eb152ed0>}
```

Model Accuracy

In [47]:
```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument, LabeledS
entence
from gensim.models import Word2Vec

w2vmodels_r = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    w2vmodels_r[size] = Word2Vec.load('model_{}.w2vrmodel'.format(s
ize))  #w2vmodel
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
```

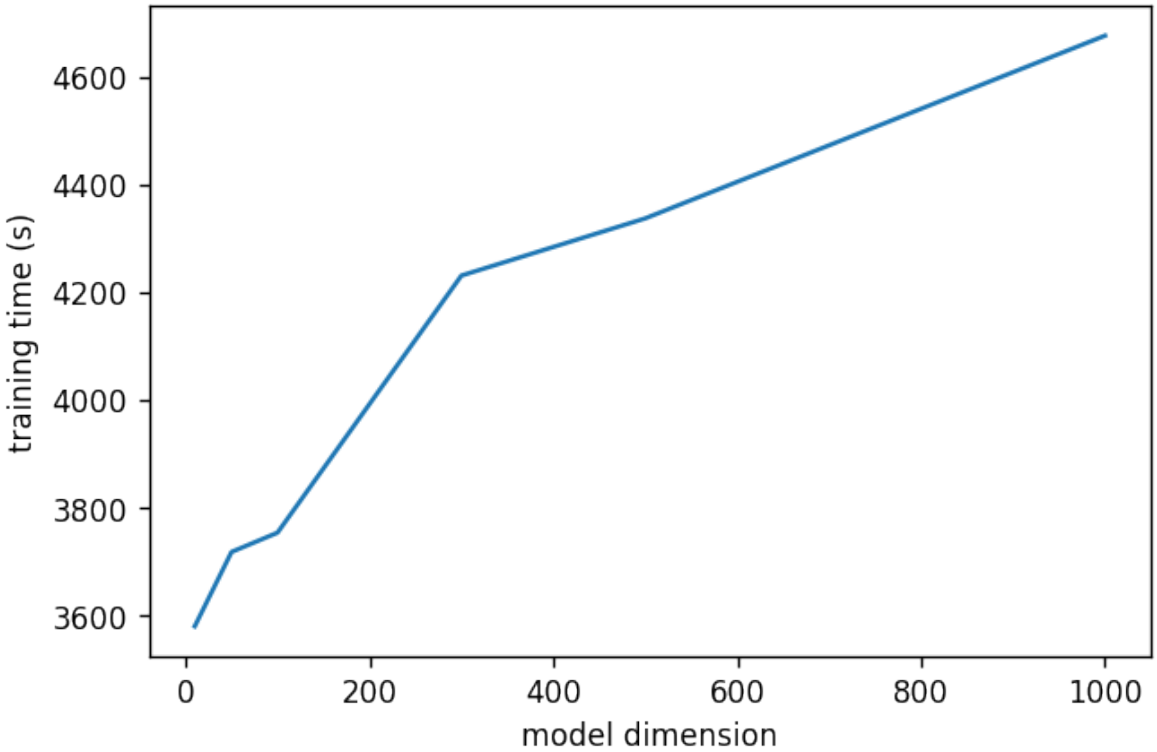
In [50]:
```
acc = w2vmodels_r[1000].accuracy('w2vanalogies.txt')
for x in acc:
    print x['section'] + " correct: " + str(len(x['correct'])) + "
incorrect: " + str(len(x['incorrect']))
```

```
capital-common-countries correct: 34 incorrect: 122
capital-world correct: 50 incorrect: 164
currency correct: 1 incorrect: 51
city-in-state correct: 63 incorrect: 451
family correct: 17 incorrect: 93
gram1-adjective-to-adverb correct: 0 incorrect: 0
gram2-opposite correct: 0 incorrect: 12
gram3-comparative correct: 424 incorrect: 632
gram4-superlative correct: 157 incorrect: 655
gram5-present-participle correct: 0 incorrect: 0
gram6-nationality-adjective correct: 73 incorrect: 465
gram7-past-tense correct: 0 incorrect: 0
gram8-plural correct: 6 incorrect: 0
gram9-plural-verbs correct: 0 incorrect: 0
total correct: 825 incorrect: 2645
```

Size effect on cossim diff and time

In [62]:
```
Dims = [10, 50, 100, 300, 500, 1000]
Diff = [42955368.690556161, 34741168.86718303, 31903039.441816971,
30409380.466192909, 30242322.913648352, 30052859.968903661]
Times = [3580.949791908264, 3718.8304538726807, 3754.622626066208,
4231.634207010269, 4337.781428098679, 4676.798229932785]

plt.xlabel('model dimension')
plt.ylabel('training time (s)')
plt.plot(Dims, Times)

plt.xlabel('model dimension')
plt.ylabel('difference score')
plt.plot(Dims, Diff)
```

Out[62]: [<matplotlib.lines.Line2D at 0x3681f95d0>]



Doc2Vec beers+users

Train Doc2Vec beers + users

In [ ]:
```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument, LabeledS
entence

docs = [TaggedDocument(words=row.text, tags=[row.user_id, row.beer_
id]) for index, row in  df.iterrows()]

d2vmodels = {}
print("Training doc vectors")
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    m = Doc2Vec(docs, workers=8, size=size, min_count = min_word_co
unt, window = context, sample = downsampling)
    m.save('d2v_model_{}.model'.format(size))
    d2vmodels[size] = m
```

Load Doc2Vec beers + users

In [63]:
```
%%time

#This is currently only for d2vmodels (which has multiple tags)
d2vmodels = {}
for size in [10, 50, 100, 300, 500, 1000]:
    print('size', size)
    d2vmodels[size] = Doc2Vec.load('d2v_model_{}.model'.format(size
))

print(d2vmodels)
```

```
('size', 10)
('size', 50)
('size', 100)
('size', 300)
('size', 500)
('size', 1000)
{100: <gensim.models.doc2vec.Doc2Vec object at 0x367b3b550>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x5ddb2f7d0>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x368201fd0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x4693cca10>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x368201ed0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x46c9cb990>}
CPU times: user 2.31 s, sys: 1.1 s, total: 3.41 s
Wall time: 3.28 s
```

In [68]:
```
for s in d2vmodels.keys():
    print(s, d2vmodels[s].docvecs.most_similar('beerfoolish', topn=
3))
    print(s, d2vmodels[s].docvecs.most_similar('246', topn=3))
print(d2vmodels)
```

```
(100, [(u'thyde606', 0.9019837975502014), (u'bdeast1', 0.861627101
8981934), (u'ouroborus', 0.858999490737915)])
(100, [(u'449', 0.8980844020843506), (u'567', 0.888591468334198),
(u'580', 0.872567892074585)])
(1000, [(u'thyde606', 0.7682191133499146), (u'ocpathfinder', 0.703
8609385490417), (u'muchloveforhops3', 0.6808160543441772)])
(1000, [(u'449', 0.6828659772872925), (u'2435', 0.6572326421737671
), (u'436', 0.648945689201355)])
(10, [(u'25311', 0.9960821866989136), (u'clr231', 0.99334931373596
19), (u'camzela', 0.9924380779266357)])
(10, [(u'580', 0.9903159141540527), (u'567', 0.9877042770385742),
(u'2842', 0.9852696061134338)])
(300, [(u'thyde606', 0.8545657396316528), (u'scottf2345', 0.757624
5665550232), (u'alodge', 0.7570635676383972)])
(300, [(u'449', 0.7771278619766235), (u'1053', 0.751703679561615),
(u'1426', 0.7470483183860779)])
(50, [(u'chugs13', 0.9261205792427063), (u'ilovestouts', 0.9242473
840713501), (u'monkeefish', 0.921308159828186)])
(50, [(u'449', 0.9343066215515137), (u'567', 0.9272139072418213),
(u'2563', 0.923660933971405)])
(500, [(u'thyde606', 0.8349712491035461), (u'nkronma', 0.755933284
7595215), (u'39672', 0.7507829070091248)])
(500, [(u'449', 0.7250908017158508), (u'436', 0.696280837059021),
(u'1426', 0.687870979309082)])
{100: <gensim.models.doc2vec.Doc2Vec object at 0x367b3b550>, 1000:
<gensim.models.doc2vec.Doc2Vec object at 0x5ddb2f7d0>, 10: <gensim
.models.doc2vec.Doc2Vec object at 0x368201fd0>, 300: <gensim.model
s.doc2vec.Doc2Vec object at 0x4693cca10>, 50: <gensim.models.doc2v
ec.Doc2Vec object at 0x368201ed0>, 500: <gensim.models.doc2vec.Doc
2Vec object at 0x46c9cb990>}
```

Calculating Similarities

W2V_r user similarities

In [69]:
```
%%time

def _user_wv(model=None, user_id=None):
    for word in np.hstack(df.query("user_id == @user_id",  engine='
python').text.values):
        if word in model.wv.vocab:
            yield model.wv[word]

wv_r_mat_300 = np.array([np.mean(list(_user_wv(model=w2vmodels_r[30
0], user_id=user_id)), axis=0) for user_id in user_idx.values()])
wv_r_sim_300 = cosine_similarity(wv_r_mat_300, wv_r_mat_300)
```

```
CPU times: user 32min 19s, sys: 37.4 s, total: 32min 57s
Wall time: 32min 44s
```

W2V_c user similarities

In [70]:
```
%%time

def _user_wv(model=None, user_id=None):
    for word in np.hstack(df.query("user_id == @user_id",  engine='
python').text.values):
        if word in model.wv.vocab:
            yield model.wv[word]

wv_c_mat_300 = np.array([np.mean(list(_user_wv(model=w2vmodels_c[30
0], user_id=user_id)), axis=0) for user_id in user_idx.values()])
wv_c_sim_300 = cosine_similarity(wv_c_mat_300, wv_c_mat_300)
```

```
CPU times: user 32min 19s, sys: 38.8 s, total: 32min 58s
Wall time: 32min 45s
```

W2V_ggl user similarities

In [71]:
```
%%time

def _user_wv(model=None, user_id=None):
    for word in np.hstack(df.query("user_id == @user_id",  engine='
python').text.values):
        if word in model.wv.vocab:
            yield model.wv[word]

wv_ggl_mat = np.array([np.mean(list(_user_wv(model=ggl_model, user_
id=user_id)), axis=0) for user_id in user_idx.values()])
wv_ggl_sim = cosine_similarity(wv_ggl_mat, wv_ggl_mat)
```

```
CPU times: user 32min 22s, sys: 39 s, total: 33min 1s
Wall time: 32min 47s
```

D2V_u user similarities

In [72]:
```
%%time

def _user_dv(model=None, user_id=None):
    for word in np.hstack(df.query("user_id == @user_id",  engine='
python').text.values):
        if word in model.wv.vocab:
            yield model.wv[word]

dv_u_mat_300 = np.array([np.mean(list(_user_dv(model=user_models[30
0], user_id=user_id)), axis=0) for user_id in user_idx.values()])
dv_u_sim_300 = cosine_similarity(dv_u_mat_300, dv_u_mat_300)
```

```
CPU times: user 32min 24s, sys: 38.2 s, total: 33min 2s
Wall time: 32min 50s
```

D2V_bu user similarities

In [73]:

```
%%time

def _user_dv(model=None, user_id=None):
    for word in np.hstack(df.query("user_id == @user_id",  engine='
python').text.values):
        if word in model.wv.vocab:
            yield model.wv[word]

dv_bu_mat_300 = np.array([np.mean(list(_user_dv(model=d2vmodels[300
], user_id=user_id)), axis=0) for user_id in user_idx.values()])
dv_bu_sim_300 = cosine_similarity(dv_bu_mat_300, dv_bu_mat_300)
```

```
CPU times: user 32min 25s, sys: 38.3 s, total: 33min 3s
Wall time: 32min 51s
```

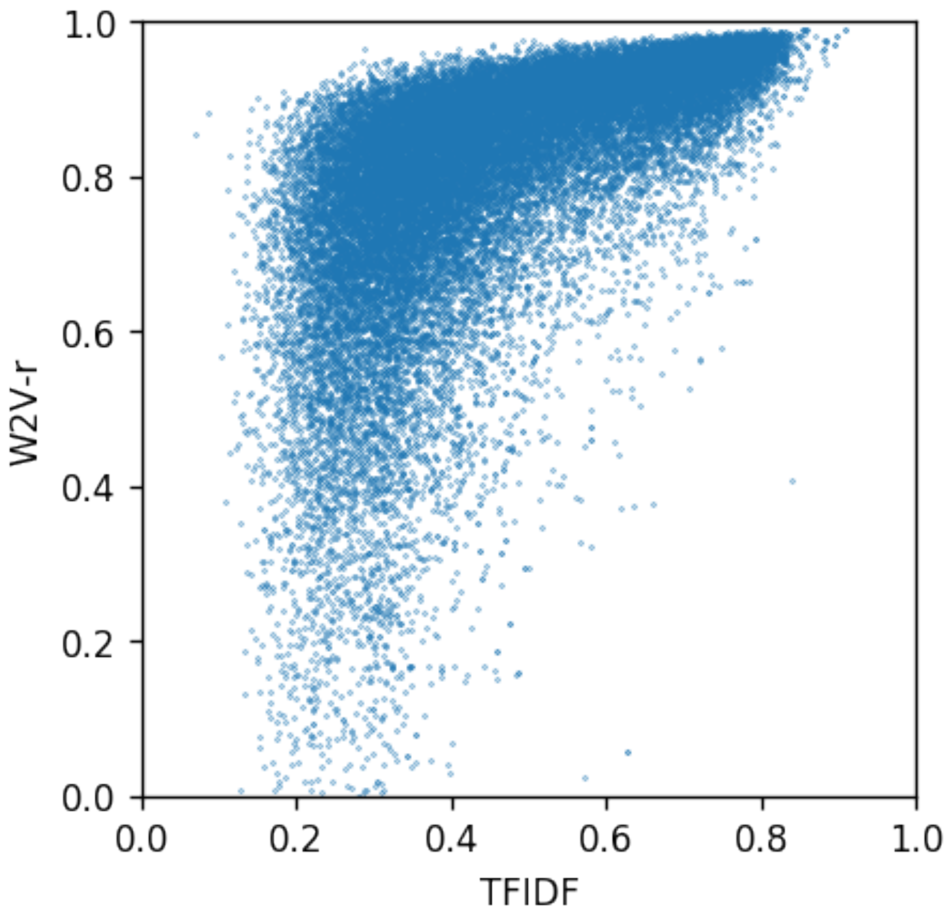
Comparing similarities

TFIDF/W2V_r Similarity

In [75]:

```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.tfidf_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('TFIDF')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

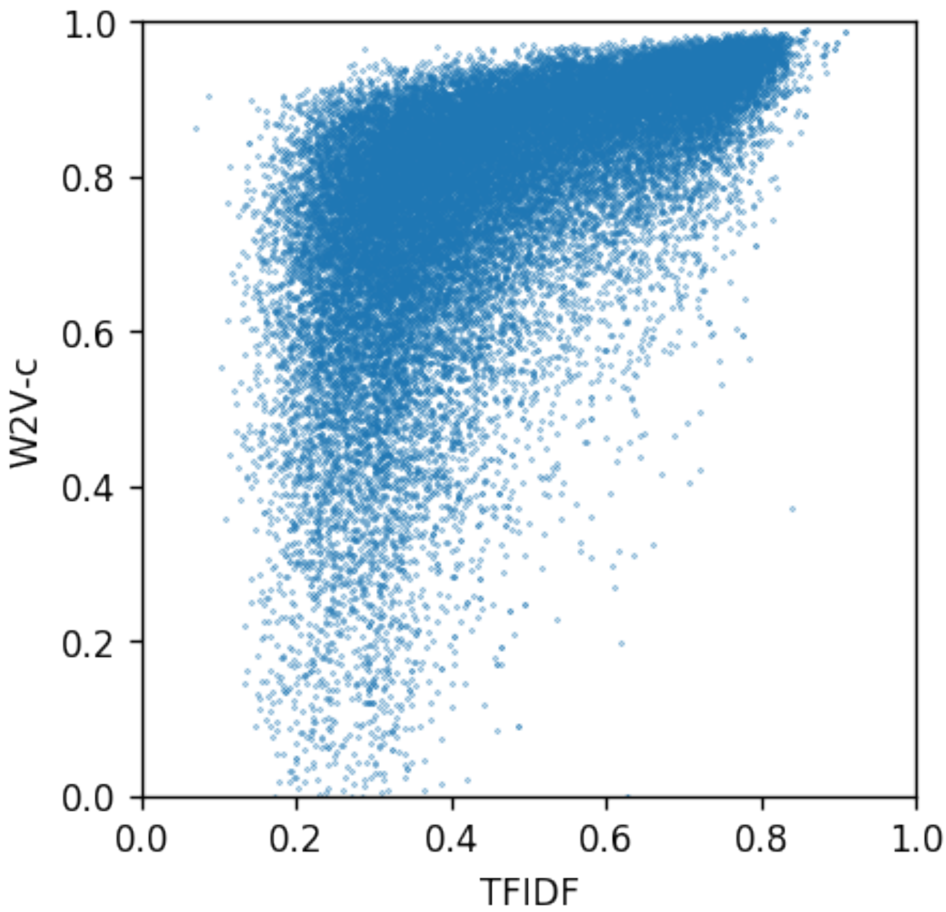
Out[75]: (0, 1)



TFIDF/W2V_c Similarity

In [76]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_c_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.tfidf_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('TFIDF')
plt.ylabel('W2V-c')
plt.xlim(0,1)
plt.ylim(0,1)
```

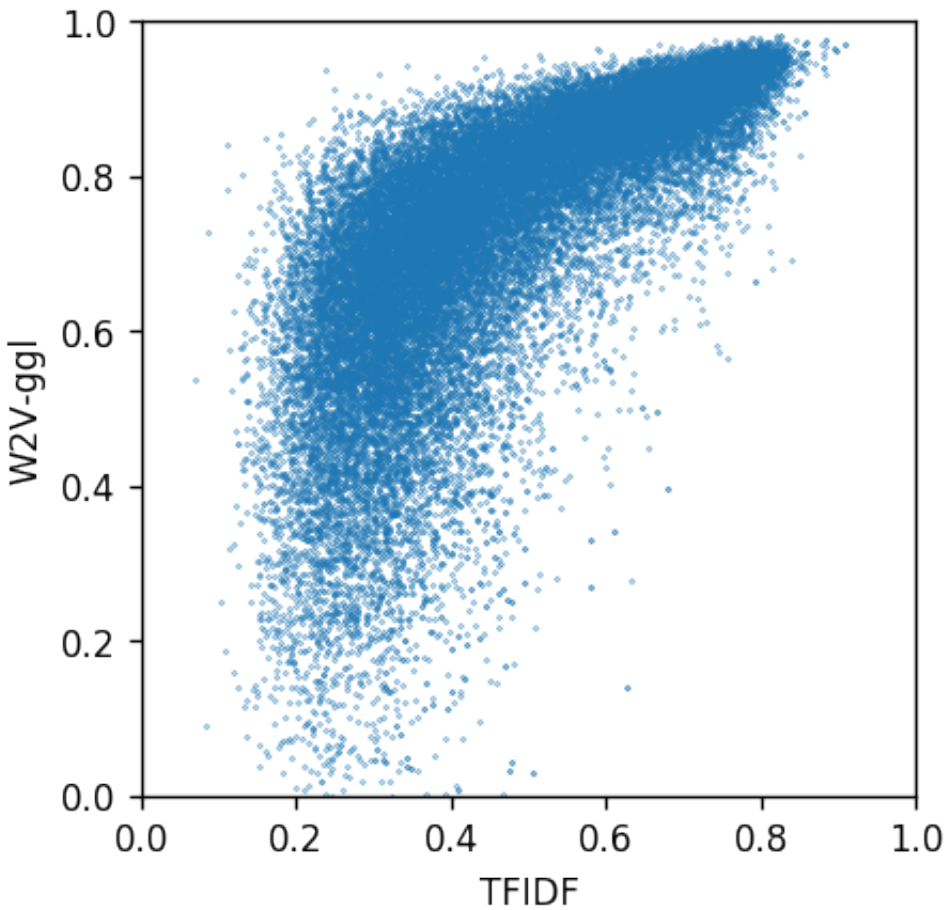
Out[76]: (0, 1)



TFIDF/W2V_ggl Similarity

In [78]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_ggl_sim[user_idx_inv[u_a]].flatten()[user_idx_i
nv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.tfidf_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('TFIDF')
plt.ylabel('W2V-ggl')
plt.xlim(0,1)
plt.ylim(0,1)
```

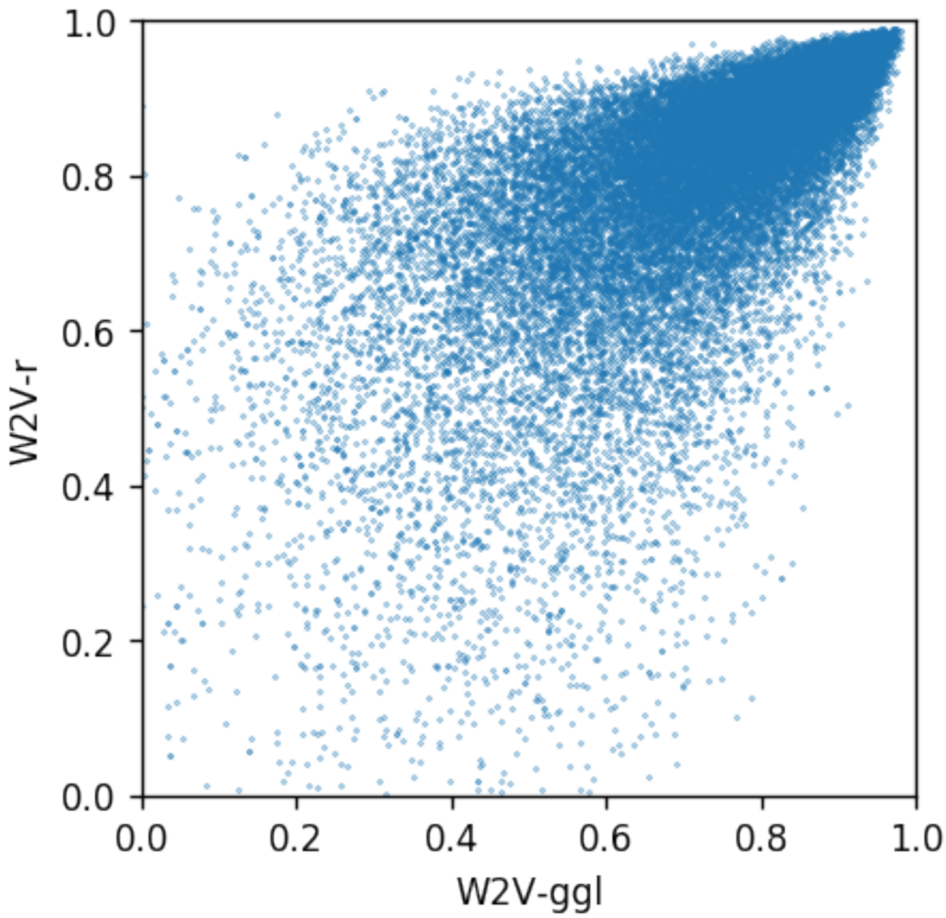
Out[78]: (0, 1)



W2V_ggl/W2V_r Similarity

In [79]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_ggl_cos_sim = wv_ggl_sim[user_idx_inv[u_a]].flatten()[user_i
dx_inv[u_b]]
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([wv_ggl_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('W2V-ggl')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

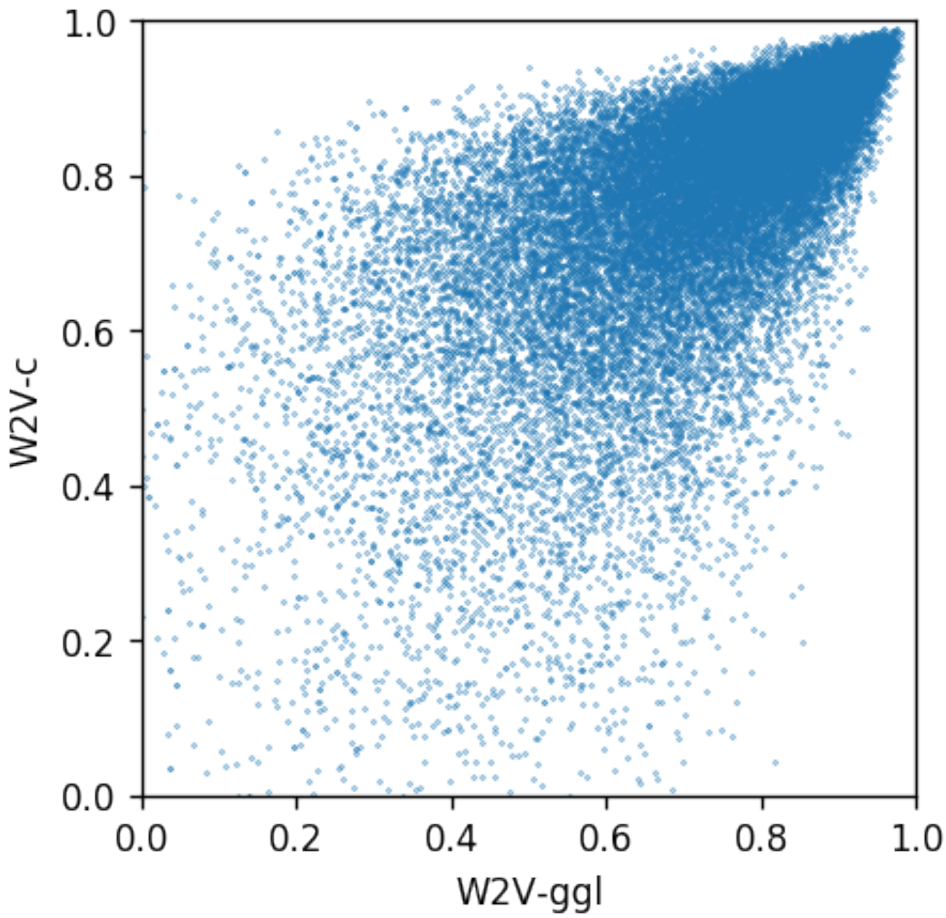
Out[79]: (0, 1)



W2V_ggl/W2V_c Similarity

In [81]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_ggl_cos_sim = wv_ggl_sim[user_idx_inv[u_a]].flatten()[user_i
dx_inv[u_b]]
    wv_cos_sim = wv_c_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([wv_ggl_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('W2V-ggl')
plt.ylabel('W2V-c')
plt.xlim(0,1)
plt.ylim(0,1)
```

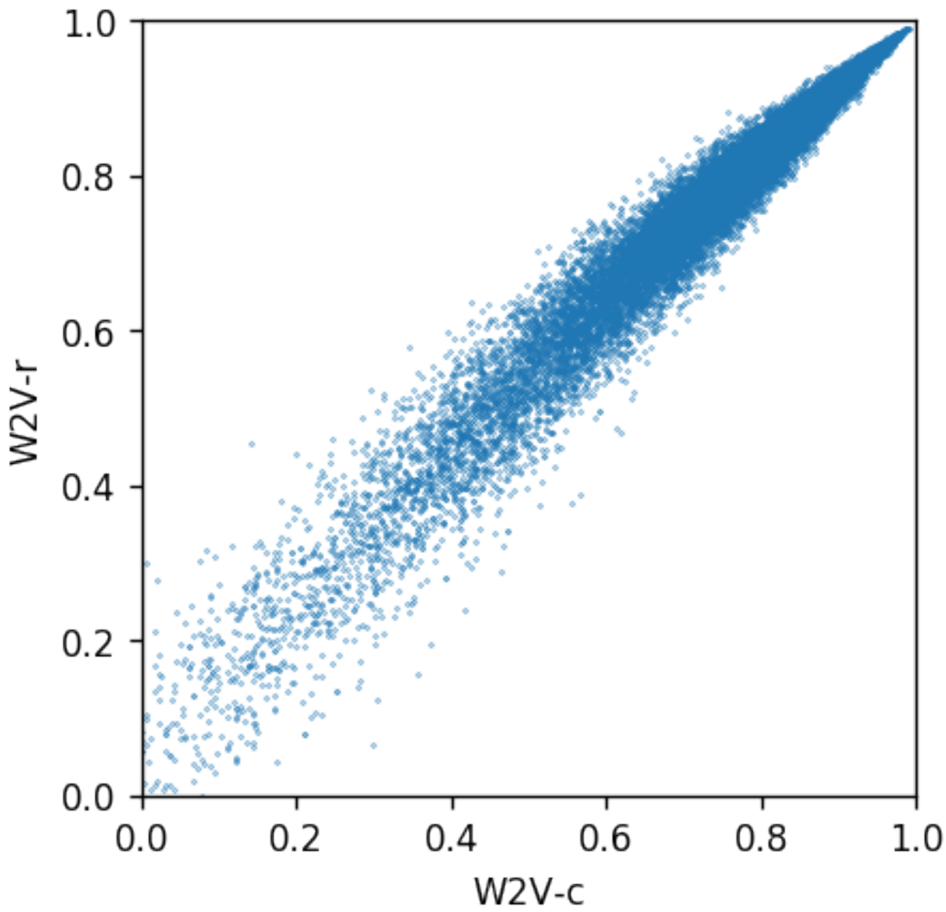
Out[81]: (0, 1)



W2V_c/W2V_r Similarity

In [82]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_ggl_cos_sim = wv_c_sim_300[user_idx_inv[u_a]].flatten()[user
_idx_inv[u_b]]
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([wv_ggl_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('W2V-c')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

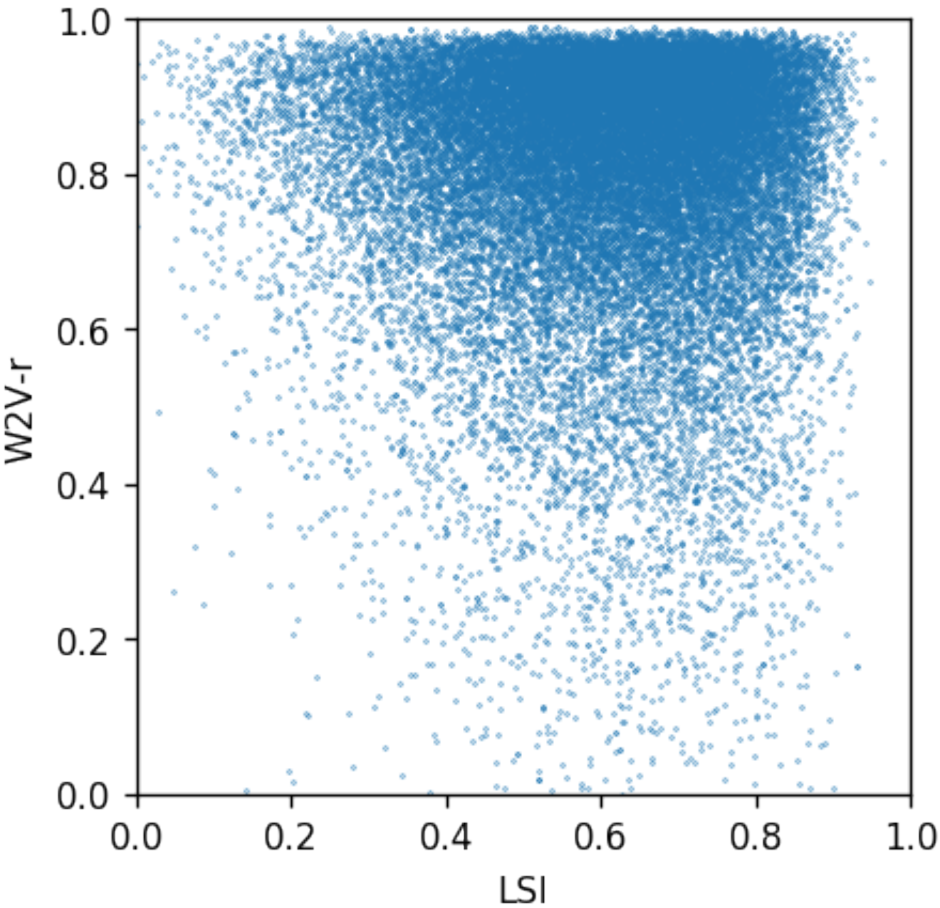
Out[82]: (0, 1)



LSI/W2V_r Similarity

In [86]:
```
sim_actual_pred = []
for i, row in df_lsi_sims.iterrows():
    u_a, u_b = row.user_id, row.lsi_sim_user
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.lsi_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)

plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('LSI')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

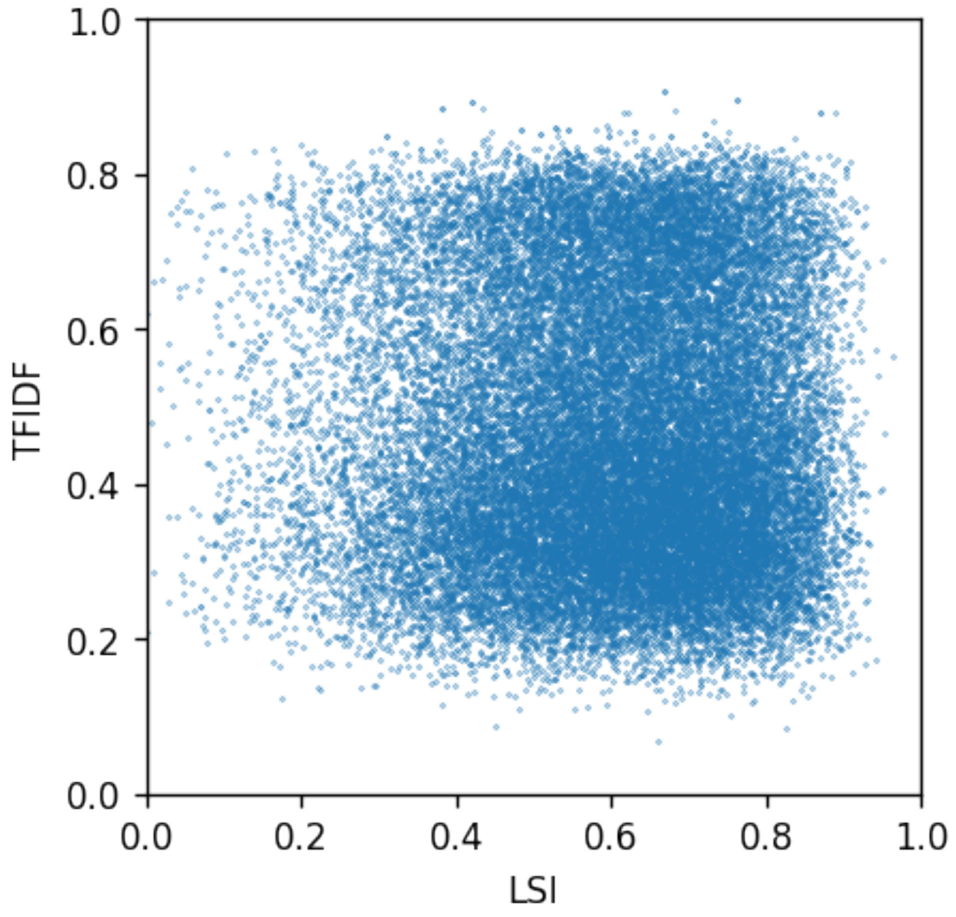
Out[86]: (0, 1)



LSI/TFIDF Similarity

In [88]:
```
sim_actual_pred = []
for i, row in df_lsi_sims.iterrows():
    u_a, u_b = row.user_id, row.lsi_sim_user
    tfidf_cos_sim = df_tfidf_sims[df_tfidf_sims.user_id==u_a].tfidf
_sim
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.lsi_sim, tfidf_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)

plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('LSI')
plt.ylabel('TFIDF')
plt.xlim(0,1)
plt.ylim(0,1)
```

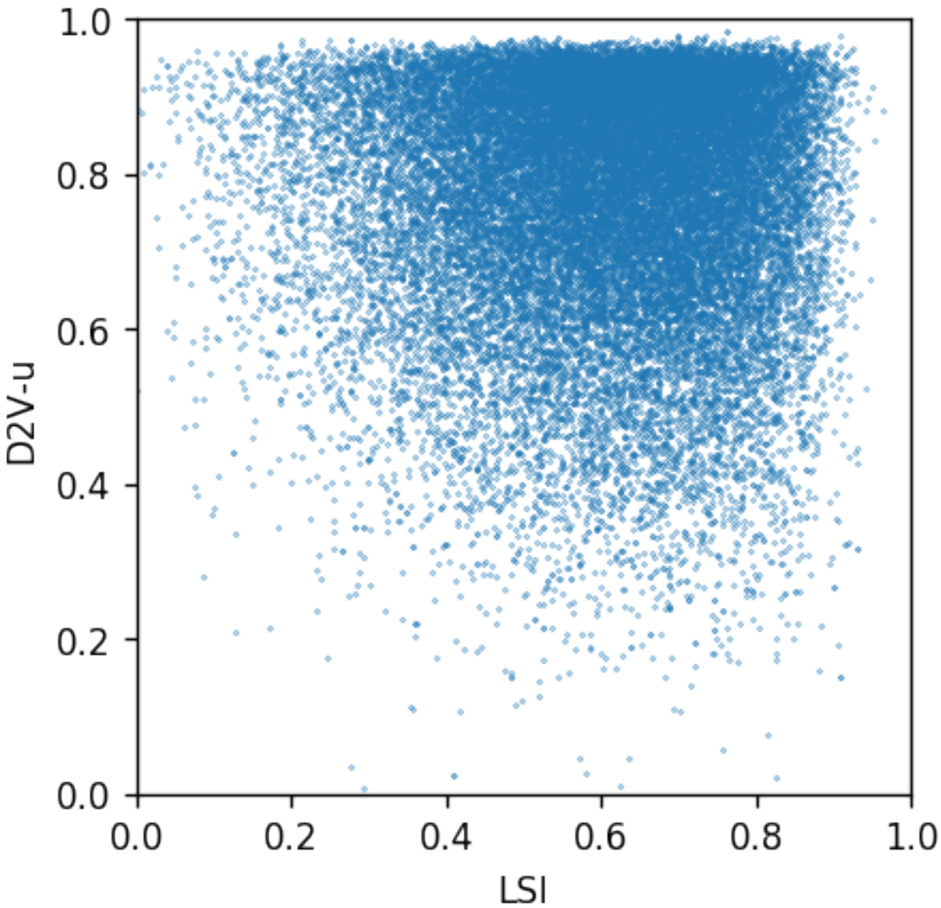
Out[88]: (0, 1)



LSI/D2V

In [89]:
```
sim_actual_pred = []
for i, row in df_lsi_sims.iterrows():
    u_a, u_b = row.user_id, row.lsi_sim_user
    dv_cos_sim = dv_u_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([row.lsi_sim, dv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)

plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('LSI')
plt.ylabel('D2V-u')
plt.xlim(0,1)
plt.ylim(0,1)
```

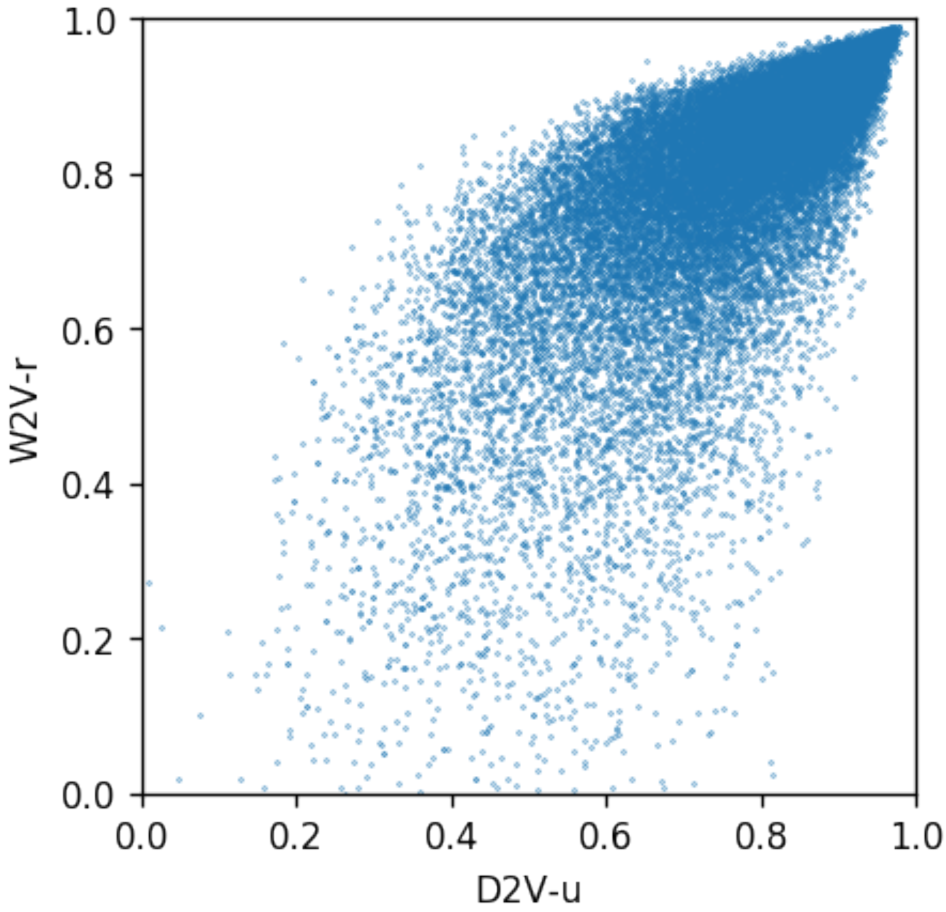
Out[89]: (0, 1)



D2V/W2V_r Similarity

In [90]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    dv_cos_sim = dv_u_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    sim_actual_pred.append([dv_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('D2V-u')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

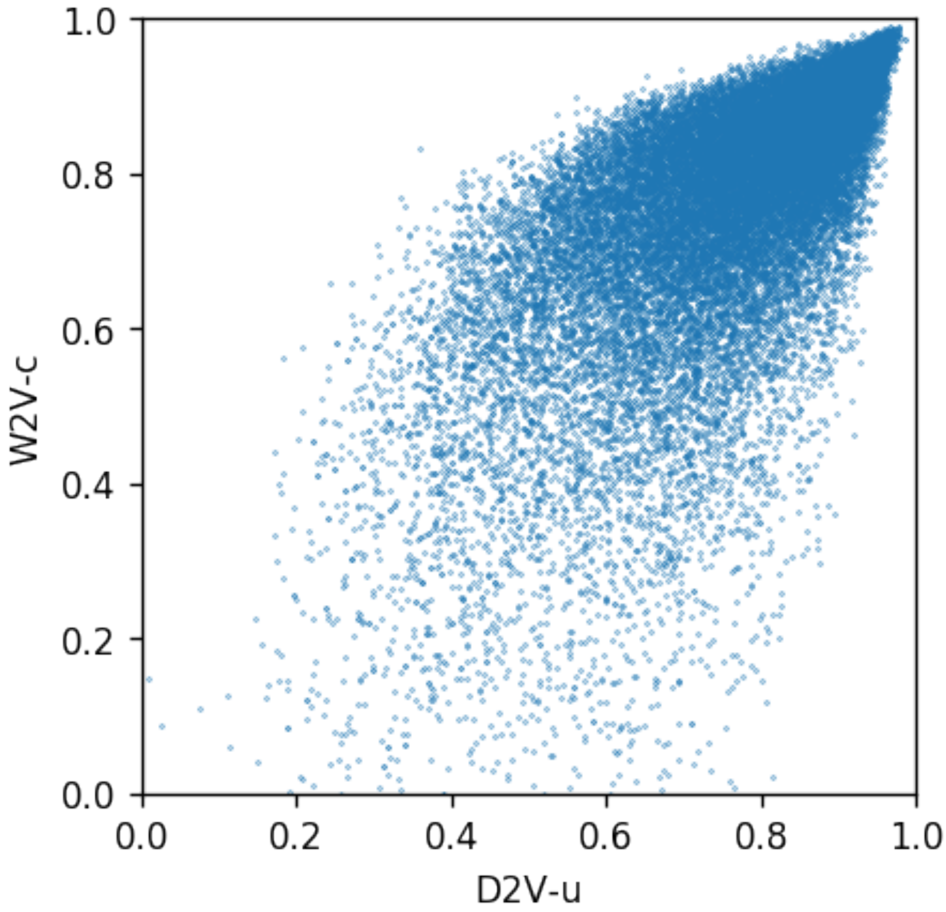
Out[90]: (0, 1)



D2V/W2V_c Similarity

In [91]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_c_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    dv_cos_sim = dv_u_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    sim_actual_pred.append([dv_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('D2V-u')
plt.ylabel('W2V-c')
plt.xlim(0,1)
plt.ylim(0,1)
```

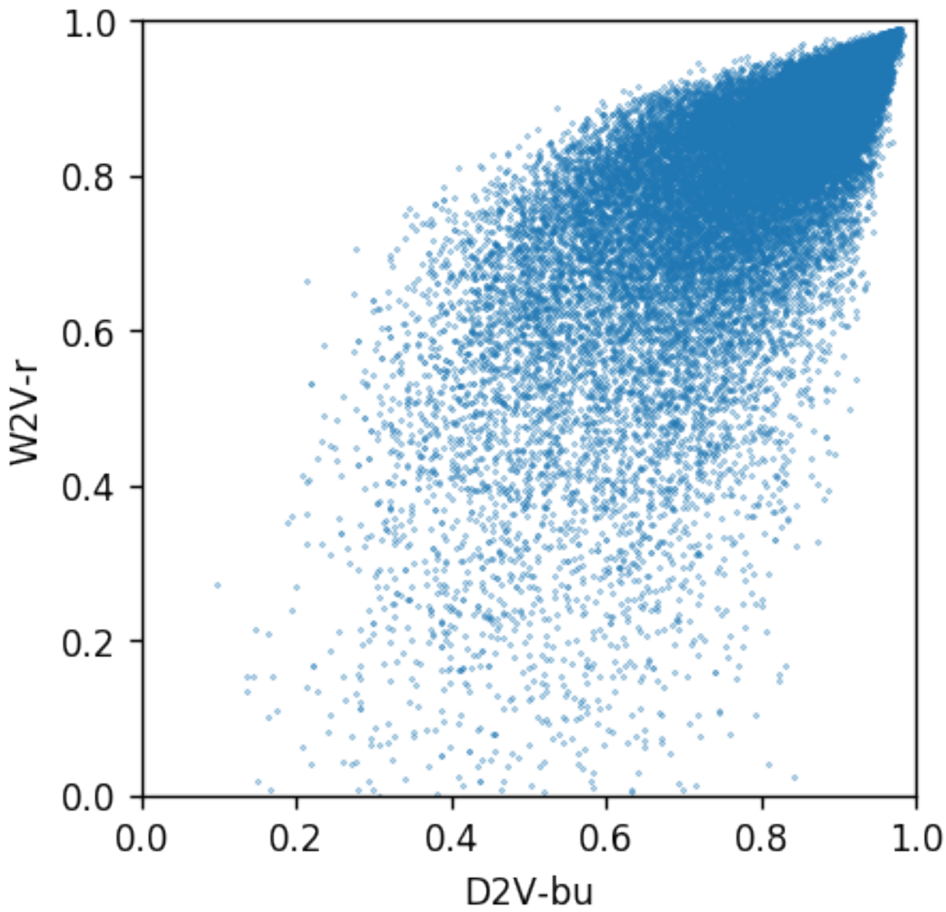
Out[91]: (0, 1)



D2V_bu/W2V_r Similarity

In [92]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_r_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    dv_cos_sim = dv_bu_sim_300[user_idx_inv[u_a]].flatten()[user_id
x_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([dv_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('D2V-bu')
plt.ylabel('W2V-r')
plt.xlim(0,1)
plt.ylim(0,1)
```

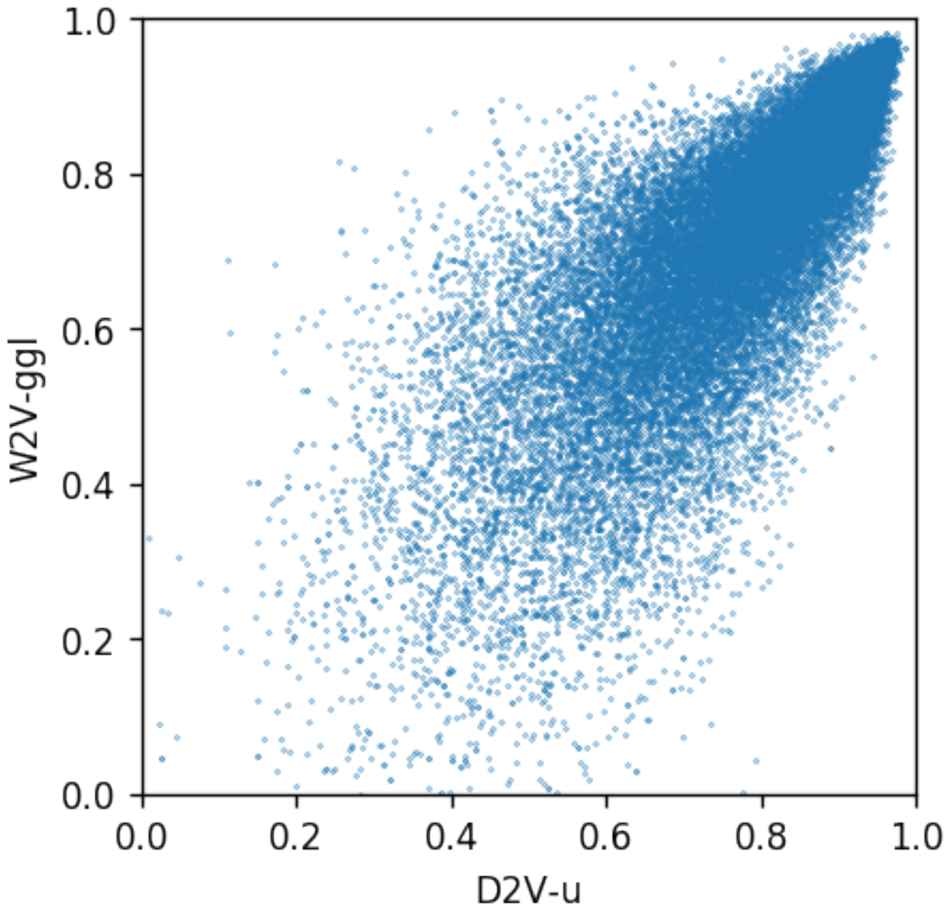
Out[92]: (0, 1)



D2V/W2V_ggl Similarity

In [94]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    wv_cos_sim = wv_ggl_sim[user_idx_inv[u_a]].flatten()[user_idx_i
nv[u_b]]
    dv_cos_sim = dv_u_sim_300[user_idx_inv[u_a]].flatten()[user_idx
_inv[u_b]]
    sim_actual_pred.append([dv_cos_sim, wv_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('D2V-u')
plt.ylabel('W2V-ggl')
plt.xlim(0,1)
plt.ylim(0,1)
```

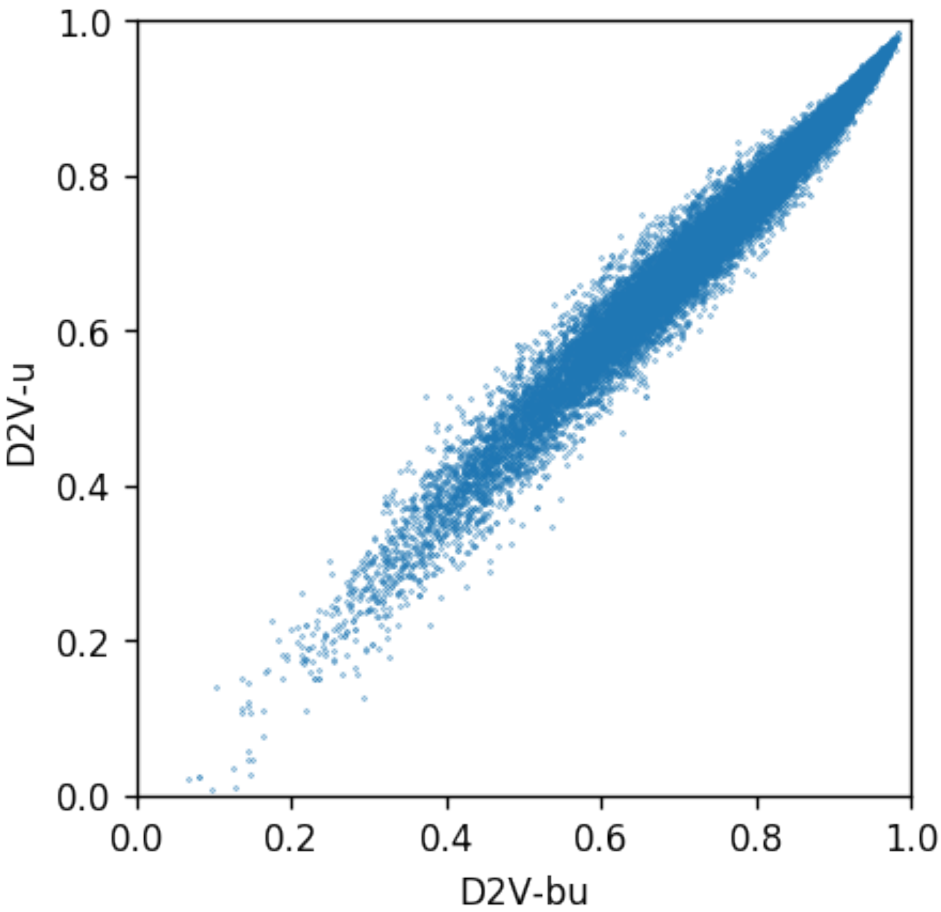
Out[94]: (0, 1)



D2Vbu/D2Vu

In [95]:
```
sim_actual_pred = []
for i, row in df_tfidf_sims.iterrows():
    u_a, u_b = row.user_id, row.tfidf_sim_user
    dvu_cos_sim = dv_u_sim_300[user_idx_inv[u_a]].flatten()[user_id
x_inv[u_b]]
    dv_cos_sim = dv_bu_sim_300[user_idx_inv[u_a]].flatten()[user_id
x_inv[u_b]]
    #print(i, u_a, u_b, row.sim, wv_cos_sim)
    sim_actual_pred.append([dv_cos_sim, dvu_cos_sim])
sim_actual_pred = np.array(sim_actual_pred)


plt.plot(sim_actual_pred[:,0], sim_actual_pred[:,1], marker='o', lw
=0, markersize=0.3)
plt.axes().set_aspect('equal')
plt.xlabel('D2V-bu')
plt.ylabel('D2V-u')
plt.xlim(0,1)
plt.ylim(0,1)
```

Out[95]: (0, 1)



TSNE

Analyze users with beer categories

In [96]:
```
%%time

#Get beer categories from the database
eng = sa.create_engine("sqlite:////Users/cagatay/Desktop/cagatay_th
esis/beeradvocate.db")
sql = "select beer_id, beer_style from beers group by beer_id, beer
_style"
beers = pd.read_sql(sql,eng)

#Add beer style into df (to be run only once, then gives errors)
df = df.set_index('beer_id').join(beers.set_index('beer_id'))
df['beer_id']= df.index
```

```
CPU times: user 846 ms, sys: 118 ms, total: 964 ms
Wall time: 1.08 s
```

In [97]:
```
#Get only top beer categories
top_styles = [u'american ipa', u'american double/imperial ipa',
              u'american pale ale (apa)', u'russian imperial stout'
, u'american double/imperial stout']
df_topstyles = df[df.beer_style.isin(top_styles)]

#change df_topstyles to df if you wanna change back to all categori
es
beer_doc_vectors = [beer_models[300].docvecs[index] for index in df
_topstyles.beer_id]
```

In [98]:
```
df_topstyles2 = df_topstyles.sample(frac=0.084, replace=True)
beer_doc_vectors = [beer_models[300].docvecs[index] for index in df
_topstyles2.beer_id]
len(df_topstyles2)
```

Out[98]:
```
30224
```

In [99]:
```
%%time

from sklearn.manifold import TSNE
tsne_model = TSNE(n_components=2, verbose=1, random_state=0, init='
pca')

tsne_d2v_beer = tsne_model.fit_transform(beer_doc_vectors)
```

```
[t-SNE] Computing pairwise distances...
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Computed conditional probabilities for sample 1000 / 30224
[t-SNE] Computed conditional probabilities for sample 2000 / 30224
[t-SNE] Computed conditional probabilities for sample 3000 / 30224
[t-SNE] Computed conditional probabilities for sample 4000 / 30224
```

```
[t-SNE] Computed conditional probabilities for sample 5000 / 30224
[t-SNE] Computed conditional probabilities for sample 6000 / 30224
[t-SNE] Computed conditional probabilities for sample 7000 / 30224
[t-SNE] Computed conditional probabilities for sample 8000 / 30224
[t-SNE] Computed conditional probabilities for sample 9000 / 30224
[t-SNE] Computed conditional probabilities for sample 10000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 11000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 12000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 13000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 14000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 15000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 16000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 17000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 18000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 19000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 20000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 21000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 22000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 23000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 24000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 25000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 26000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 27000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 28000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 29000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 30000 / 3022
4
[t-SNE] Computed conditional probabilities for sample 30224 / 3022
4
[t-SNE] Mean sigma: 0.000000
[t-SNE] KL divergence after 100 iterations with early exaggeration
: 0.510597
[t-SNE] Error after 375 iterations: 0.510597
```

```
CPU times: user 27min 44s, sys: 9min 36s, total: 37min 20s
Wall time: 28min 58s
```

In [101]:
```
#See beer category distribution
#df.groupby('beer_style').agg('count').sort_values('beer_id', ascending=False).head(5).index.values
df.groupby('beer_style').agg('count').sort_values('beer_id', ascending=False).text[0:10]
```

Out[101]:
```
beer_style
american ipa                       113124
american double/imperial ipa        85091
american pale ale (apa)             58059
russian imperial stout              53406
american double/imperial stout      50130
american porter                     46633
american amber/red ale              41721
belgian strong dark ale             37482
fruit/vegetable beer                31976
american strong ale                 31340
Name: text, dtype: int64
```

TSNE of beers with beer categories

In [103]:
```
import warnings
warnings.filterwarnings('ignore')

import bokeh.plotting as bp
from bokeh.models import HoverTool, BoxSelectTool
from bokeh.plotting import figure, show, output_notebook

colormap = {u'american ipa': '#00afb8', u'american double/imperial
ipa': '#110797', u'american pale ale (apa)': '#5f009c', u'american
double/imperial stout': '#ffff00', u'russian imperial stout': '#ffa
500'}
colors = [colormap[x] for x in df_topstyles2['beer_style']]

plot_d2v = bp.figure(plot_width=900, plot_height=700, title="D2V Be
er Models by Beer Category",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

from bokeh.models.mappers import LinearColorMapper
plot_d2v.scatter(x=tsne_d2v_beer[:,0], y=tsne_d2v_beer[:,1],
                    color=colors,
                    source=bp.ColumnDataSource({
                        "review": df_topstyles2.text,
                        "beer_id": df_topstyles2.beer_id,
                        "cat": df_topstyles2.beer_style
        }))
hover = plot_d2v.select(dict(type=HoverTool))
hover.tooltips=[("review", "@review"), ("beer_id", "@beer_id"), ("c
ategory", "@cat")]
output_notebook()
show(plot_d2v)
```

(http://bokeh.pydata.org) BokehJS 0.12.5 successfully loaded.

In [141]:
```
#df.groupby('beer_id').nunique().sort_values('text', ascending=Fals
e)['text'][0:20]
np.unique(df[df.beer_id.isin(top_beers) & df.beer_style.isin(top_st
yles)][['beer_id', 'beer_style']])
```

Out[141]:
```
array([u'1005', u'1013', u'1093', u'1160', u'11757', u'17112', u'1
904',
       u'2093', u'276', u'4083', u'412', u'6108', u'680', u'7971',
u'88',
       u'american double/imperial ipa', u'american double/imperial
stout',
       u'american ipa', u'american pale ale (apa)',
       u'russian imperial stout'], dtype=object)
```

In [144]:
```
import warnings
warnings.filterwarnings('ignore')

import bokeh.plotting as bp
from bokeh.models import HoverTool, BoxSelectTool
from bokeh.plotting import figure, show, output_notebook

top_beers = [u'1005', u'1013', u'1093', u'1160', u'11757', u'17112'
, u'1904', u'2093', u'276', u'4083', u'412', u'6108', u'680', u'797
1', u'88']
colormap = {'4083': '#0066ff', '17112': '#0000ff', '7971': '#08457e
', '2093': '#180367', '11757': '#98e698', '1904': '#ff8787', '88':
'#ffc0cb', '1093': '#ff69b4', '6108': '#ffd3d3', '1005': '#e3d8ea',
'276': '#323232', '1160': '#ffe0a5 ', '412': '#ffa500 ', '680': '#b
d8110 ', '1013': '#ffe9c1 '}
colors = [colormap[x] for x in df_topstyles2['beer_id'] if x in top
_beers]

plot_d2v = bp.figure(plot_width=900, plot_height=700, title="D2V Be
er Models by Beer Category",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

from bokeh.models.mappers import LinearColorMapper
plot_d2v.scatter(x=tsne_d2v_beer[:,0], y=tsne_d2v_beer[:,1],
                    color=colors,
                    source=bp.ColumnDataSource({
                        "review": df_topstyles2.text,
                        "beer_id": df_topstyles2.beer_id,
                        "cat": df_topstyles2.beer_style
        }))
hover = plot_d2v.select(dict(type=HoverTool))
hover.tooltips=[("review", "@review"), ("beer_id", "@beer_id"), ("c
ategory", "@cat")]
output_notebook()
show(plot_d2v)
```

(http://bokeh.pydata.org) BokehJS 0.12.5 successfully loaded.

TSNE of top 95 users

In [ ]:
```
%%time

#Get only top users
top_users = list(df.user_id.value_counts()[0:95].index)
df_topusers = df[df.user_id.isin(top_users)]

#change df_topusers to df if you wanna change back to all users
user_doc_vectors = [user_models[300].docvecs[index] for index in df
_topusers.user_id]

from sklearn.manifold import TSNE
tsne_model = TSNE(n_components=2, verbose=1, random_state=0)

tsne_d2v_user = tsne_model.fit_transform(user_doc_vectors)
```

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')

import bokeh.plotting as bp
from bokeh.models import HoverTool, BoxSelectTool
from bokeh.plotting import figure, show, output_notebook

colormap = {u'northyorksammy': '#FFF633', u'masterski': '#FAEBD7',
u'barleywinefiend': '#00FFFF', u'mikesgroove': '#7FFFD4', u'ccrida'
: '#F0FFFF', u'buckeyenation': '#F5F5DC', u'reddiamond': '#FFE4C4',
u'beerchitect': '#ffe0a5', u'chaingangguy': '#FFEBCD', u'jason': '#
0000FF', u'oberon': '#8A2BE2', u'smcolw': '#A52A2A', u'brentk56': '
#DEB887', u'wasatch': '#5F9EA0', u'tone': '#7FFF00', u'gavage': '#D
2691E', u'emerge077': '#FF7F50', u'brent': '#6495ED', u'kegatron':
'#FFF8DC', u'thorpe429': '#DC143C', u'vancer': '#00FFFF', u'russpow
ell': '#00008B', u'merlin48': '#008B8B', u'jamess': '#B8860B', u'vi
ggo': '#A9A9A9', u'georgiabeer': '#A9A9A9', u'wl0307': '#006400', u
'cyberkedi': '#BDB76B', u'jdhilt': '#8B008B', u'unclejimbo': '#556B
2F', u'agentzero': '#FF8C00', u'bluejacket74': '#9932CC', u'weswes'
: '#8B0000', u'taez555': '#E9967A', u'dogbrick': '#8FBC8F', u'zeff8
0': '#483D8B', u'brewdlyhooked13': '#2F4F4F', u'derek': '#2F4F4F',
u'slatetank': '#00CED1', u'ibunit63': '#9400D3', u'womencantsail':
'#FF1493', u'beertunes': '#00BFFF', u'akorsak': '#696969', u'phyl21
ca': '#696969', u'feloniousmonk': '#1E90FF', u'scruffwhor': '#B2222
2', u'glid02': '#FFFAF0', u'beerandraiderfan': '#228B22', u'royalt'
: '#FF00FF', u'jays2629': '#DCDCDC', u'jwc215': '#F8F8FF', u'tmoney
2591': '#FFD700', u'jrallen34': '#DAA520', u'tpd975': '#808080', u'
mora2000': '#808080', u'drabmuh': '#008000', u'tempest': '#ADFF2F',
u'adamette': '#F0FFF0', u'nerofiddled': '#FF69B4', u'weeare138': '#
CD5C5C', u'stcules': '#4B0082', u'hopdog': '#FFFFF0', u'clvand0': '
#F0E68C', u'beeradvocate': '#E6E6FA', u'reagan1984': '#FFF0F5', u'b
lackhaddock': '#7CFC00', u'u2carew': '#FFFACD', u'bark': '#ADD8E6',
u'bung': '#F08080', u'cresant': '#E0FFFF', u'rhoadsrage': '#FAFAD2'
, u'thagr81us': '#D3D3D3', u'jpm30': '#D3D3D3', u'beer2day': '#90EE
90', u'bierman9': '#FFB6C1', u'lilbeerdoctor': '#FFA07A', u'mcallis
ter': '#20B2AA', u'epicac': '#87CEFA', u'gratefulbeerguy': '#778899
', u'pootz': '#778899', u'jlindros': '#B0C4DE', u'wvbeergeek': '#FF
```

```
FFE0', u'tilley4': '#00FF00', u'gusler': '#32CD32', u'nrbw23': '#FA
F0E6', u'mynie': '#FF00FF', u'pencible': '#800000', u'thelongbeachb
um': '#66CDAA', u'tmoneyba': '#0000CD', u'redrover': '#BA55D3', u't
urdfurgison': '#9370DB', u'johngalt1': '#3CB371', u'lacqueredmouse'
: '#7B68EE', u'rblwthacoz': '#00FA9A', u'bierguy5': '#48D1CC'}
colors = [colormap[x] for x in df_topusers['user_id']]

plot_d2v = bp.figure(plot_width=900, plot_height=700, title="D2V Be
er Reviews by User",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

from bokeh.models.mappers import LinearColorMapper
plot_d2v.scatter(x=tsne_d2v_user[:,0], y=tsne_d2v_user[:,1],
                    color=colors,
                    source=bp.ColumnDataSource({
                        "review": df_topusers.text,
                        "beer_id": df_topusers.beer_id,
                        "user_id": df_topusers.user_id,
                        "beer_cat": df_topusers.beer_style
        }))
hover = plot_d2v.select(dict(type=HoverTool))
hover.tooltips=[("review", "@review"), ("beer_id", "@beer_id"), ("u
ser_id", "@user_id"), ("beer_cat", "@beer_cat")]
output_notebook()
show(plot_d2v)
```

In [ ]:

```
df.user_id.value_counts()[0:10]
```

TSNE of Mid 95 Users

In [ ]:

```
%%time

#Get only top users
top_users = list(df.user_id.value_counts()[2401:2495].index)
df_topusers = df[df.user_id.isin(top_users)]

#change df_topusers to df if you wanna change back to all users
user_doc_vectors = [user_models[300].docvecs[index] for index in df
_topusers.user_id]

from sklearn.manifold import TSNE
tsne_model = TSNE(n_components=2, verbose=1, random_state=0)

tsne_d2v_user = tsne_model.fit_transform(user_doc_vectors)
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')

import bokeh.plotting as bp
```

```
from bokeh.models import HoverTool, BoxSelectTool
from bokeh.plotting import figure, show, output_notebook

colormap = {u'waltersrj': '#F0F8FF', u'slentz': '#FAEBD7', u'gregor
yvii': '#00FFFF', u'gonzaloyanna': '#7FFFD4', u'xylophonica': '#F0F
FFF', u'specialk088': '#F5F5DC', u'pecorasc': '#FFE4C4', u'redbaron
': '#000000', u's1ckboy': '#FFEBCD', u'crotor': '#0000FF', u'bootle
gger1929': '#8A2BE2', u'skutra': '#A52A2A', u'gabrielsyme': '#DEB88
7', u'djmichaelk': '#5F9EA0', u'kevman527': '#7FFF00', u'kjburrows'
: '#D2691E', u'roan22': '#FF7F50', u'sisupride': '#6495ED', u'johnw
': '#FFF8DC', u'jwaks': '#DC143C', u'dogfooddog': '#00FFFF', u'pixr
ob': '#00008B', u'cylinsier': '#008B8B', u'mcfunkyj': '#B8860B', u'
baconbourbon': '#A9A9A9', u'elizabethcraig': '#A9A9A9', u'daehpoh':
'#006400', u'slaintemhor': '#BDB76B', u'rmac111': '#8B008B', u'ajst
ank': '#556B2F', u'drumminbrewer': '#FF8C00', u'f2brewers': '#9932C
C', u'icetrey9785': '#8B0000', u'bigjim5021': '#E9967A', u'ahking':
'#8FBC8F', u'khumbard': '#483D8B', u'kraken': '#2F4F4F', u'wiseguy1
42': '#2F4F4F', u'jlh8643': '#00CED1', u'ipogios': '#9400D3', u'cva
ypai': '#FF1493', u'tippebrewcrew2': '#00BFFF', u'betsona': '#69696
9', u'raninator84': '#696969', u'nickname': '#1E90FF', u'freshhawk'
: '#B22222', u'jneiswender': '#FFFAF0', u'smittysguinness': '#228B2
2', u'dbexpert': '#FF00FF', u'beerisfoodllc': '#DCDCDC', u'theduder
ules': '#F8F8FF', u'awalk1227': '#FFD700', u'vw73182': '#DAA520', u
'bonsreeb85': '#808080', u'golden2wenty1': '#808080', u'miketd': '#
008000', u'trevorjn06': '#ADFF2F', u'gregmurer': '#F0FFF0', u'mcowg
ill': '#FF69B4', u'beeronthebrain': '#CD5C5C', u'paxchristi': '#4B0
082', u'solomon420': '#FFFFF0', u'flux': '#F0E68C', u'homerbag': '#
E6E6FA', u'atowle25': '#FFF0F5', u'claytri': '#7CFC00', u'schroeder
m': '#FFFACD', u'mrhurmateeowish': '#ADD8E6', u'translucent': '#F08
080', u'kmcg': '#E0FFFF', u'steeltown71': '#FAFAD2', u'vande': '#D3
D3D3', u'vanrassel': '#D3D3D3', u'ojiikun': '#90EE90', u'olieipa':
'#FFB6C1', u'ericj551': '#FFA07A', u'scalffd': '#20B2AA', u'alewife
': '#87CEFA', u'drunkensailor': '#778899', u'gregblatz': '#778899',
u'unclederby': '#B0C4DE', u'brewmaven': '#FFFFE0', u'kingspank': '#
00FF00', u'hoboagogo': '#32CD32', u'bosox941827': '#FAF0E6', u'stja
ernstoft': '#FF00FF', u'eweneek83': '#800000', u'jushoppy2beer': '#
66CDAA', u'codsmith': '#0000CD', u'iceball585': '#BA55D3', u'bjrhom
ebrew': '#9370DB', u'beerisheaven': '#3CB371', u'box299': '#7B68EE'
, u'kelp': '#00FA9A', u'bierguy5': '#48D1CC'}
colors = [colormap[x] for x in df_topusers['user_id']]

plot_d2v = bp.figure(plot_width=900, plot_height=700, title="D2V Be
er Reviews by User",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

from bokeh.models.mappers import LinearColorMapper
plot_d2v.scatter(x=tsne_d2v_user[:,0], y=tsne_d2v_user[:,1],
                    color=colors,
                    source=bp.ColumnDataSource({
                        "review": df_topusers.text,
                        "beer_id": df_topusers.beer_id,
                        "user_id": df_topusers.user_id,
```

```
                              "beer_cat": df_topusers.beer_style
          }))
hover = plot_d2v.select(dict(type=HoverTool))
hover.tooltips=[("review", "@review"), ("beer_id", "@beer_id"), ("u
ser_id", "@user_id"), ("beer_cat", "@beer_cat")]
output_notebook()
show(plot_d2v)
```

TSNE of low 95 users

In [ ]:
```
%%time

#Get only top users
top_users = list(df.user_id.value_counts()[4851:4945].index)
df_topusers = df[df.user_id.isin(top_users)]

#change df_topusers to df if you wanna change back to all users
user_doc_vectors = [user_models[300].docvecs[index] for index in df
_topusers.user_id]

from sklearn.manifold import TSNE
tsne_model = TSNE(n_components=2, verbose=1, random_state=0)

tsne_d2v_user = tsne_model.fit_transform(user_doc_vectors)
```

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')

import bokeh.plotting as bp
from bokeh.models import HoverTool, BoxSelectTool
from bokeh.plotting import figure, show, output_notebook

colormap = {u'fank2788': '#F0F8FF', u'fabric8r': '#FAEBD7', u'cityb
oy1986': '#00FFFF', u'bigchris1313': '#7FFFD4', u'davey': '#F0FFFF'
, u'agbarr': '#F5F5DC', u'davidpgibbons': '#FFE4C4', u'px2grafx': '
#000000', u'glotz': '#FFEBCD', u'only23': '#0000FF', u'crewfan16':
'#8A2BE2', u'taelec': '#A52A2A', u'wethorseblanket': '#DEB887', u'v
alevapor': '#5F9EA0', u'skarod': '#7FFF00', u'crushedvol': '#D2691E
', u'jeffbob': '#FF7F50', u'jerryc123': '#6495ED', u'akajimmyd': '#
FFF8DC', u'seh410': '#DC143C', u'beercon5': '#00FFFF', u'wjvii': '#
00008B', u'fillpott': '#008B8B', u'therickguy78': '#B8860B', u'1whi
skey': '#A9A9A9', u'ukipiper': '#A9A9A9', u'wegomlegging': '#006400
', u'hunter': '#BDB76B', u'fatherskull': '#8B008B', u'meadhead': '#
556B2F', u'haveaduff2': '#FF8C00', u'fernans': '#9932CC', u'etienne
': '#8B0000', u'sethmeister': '#E9967A', u'bamabrew22': '#8FBC8F',
u'latackbeer': '#483D8B', u'vicsju1991': '#2F4F4F', u'rivalrome': '
#2F4F4F', u'andreix': '#00CED1', u'rodrjf': '#9400D3', u'johnpputz'
: '#FF1493', u'swamper': '#00BFFF', u'pjbear05': '#696969', u'tende
rstone': '#696969', u'finewinemike': '#1E90FF', u'mcduhamel': '#B22
222', u'lordeche': '#FFFAF0', u'cyn1c': '#228B22', u'stevecee2003':
```

```
'#FF00FF', u'yuenglingfan101': '#DCDCDC', u'andrecrompton': '#F8F8F
F', u'boospackage': '#FFD700', u'taddyporter': '#DAA520', u'wolfint
hemirror': '#808080', u'israel': '#808080', u'hindbender': '#008000
', u'doppelmax': '#ADFF2F', u'robbrandes': '#F0FFF0', u'watchnerd':
'#FF69B4', u'mrretardedmonkey': '#CD5C5C', u'jlebowski': '#4B0082',
u'dcurfman': '#FFFFF0', u'tenesmus': '#F0E68C', u'thepeskydingo': '
#E6E6FA', u'baldiegooner': '#FFF0F5', u'cdrummer': '#7CFC00', u'rym
ar9': '#FFFACD', u'slayer3604': '#ADD8E6', u'bucksbrew': '#F08080',
u'fo5s': '#E0FFFF', u'andrewinski1': '#FAFAD2', u'rutgersbeerguy':
'#D3D3D3', u'route66pubs': '#D3D3D3', u'aeminter': '#90EE90', u'bea
rcobb': '#FFB6C1', u'taxwarrior': '#FFA07A', u'derty1': '#20B2AA',
u'brewman13': '#87CEFA', u'myrtlebeachbums': '#778899', u'jbogan':
'#778899', u'cstrong821': '#B0C4DE', u'bigaizsosexy': '#FFFFE0', u'
popkas': '#00FF00', u'webbcreative': '#32CD32', u'banjaxed': '#FAF0
E6', u'belenita': '#FF00FF', u'flipper2gv': '#800000', u'claybeer':
'#66CDAA', u'jjboesen': '#0000CD', u'patm1986': '#BA55D3', u'season
six': '#9370DB', u'thekuz17': '#3CB371', u'css82420': '#7B68EE', u'
lethalbadger': '#00FA9A', u'bierguy5': '#48D1CC'}
colors = [colormap[x] for x in df_topusers['user_id']]

plot_d2v = bp.figure(plot_width=900, plot_height=700, title="D2V Be
er Reviews by User",
    tools="pan,wheel_zoom,box_zoom,reset,hover,previewsave",
    x_axis_type=None, y_axis_type=None, min_border=1)

from bokeh.models.mappers import LinearColorMapper
plot_d2v.scatter(x=tsne_d2v_user[:,0], y=tsne_d2v_user[:,1],
                    color=colors,
                    source=bp.ColumnDataSource({
                        "review": df_topusers.text,
                        "beer_id": df_topusers.beer_id,
                        "user_id": df_topusers.user_id,
                        "beer_cat": df_topusers.beer_style
        }))
hover = plot_d2v.select(dict(type=HoverTool))
hover.tooltips=[("review", "@review"), ("beer_id", "@beer_id"), ("u
ser_id", "@user_id"), ("beer_cat", "@beer_cat")]
output_notebook()
show(plot_d2v)
```