

# 多元话题竞争传播建模和预测-Report

---

517030910348 聂圣昊

## 多元话题竞争传播建模和预测-Report

- 1.Backgroud&Traditional Model
  - 1.1 Backgroud
  - 1.2 Traditional Model
    - 1.2.1 SH Model
    - 1.2.2 IMM Model
- 2.Related Concepts
  - 2.1Strong Ties
  - 2.2User Competition Matrix
- 3.Model building
  - 3.1 Text Classification
    - 3.1.1 Word2vec
    - 3.1.2 CNN Classification
  - 3.2 Probe User
  - 3.3 SH&Strong Ties Model
  - 3.4 Competetion Model
- 4.Result
  - 4.1 SH&SH+Strong Ties Model
  - 4.2 Competition Model
  - 4.3 Contrast
- 5. Summary
- 6. Reference

## 1.Backgroud&Traditional Model

---

### 1.1 Backgroud

Multi topic competition communication is of great help to the government, enterprises, etc. For example, government can conduct preventive public opinion control over emergencies and hot events and enterprises can predict hot news ahead of time ,understand user focus and achieve a better business model of social networking sites and news apps. However, the traditional models are most prediction model of single topic heat.

---

### 1.2 Traditional Model

#### 1.2.1 SH Model

According to the SH model, there is a strong linear correlation between the logarithm of early heat and that of late heat. The formula is:

$$\ln N_s(t_2) = \ln r(t_1, t_2) + \ln N_s(t_1) + \epsilon_s(t_1, t_2)$$

Where  $N(t)$  is the network heat at time  $t$ , the first term on the right side of the equation is the intercept of linear function, and the last term is the error.

---

### 1.2.2 IMM Model

For the first time, the IMM model takes into account the category characteristics behind the topic content and applies them to information dissemination. They take into account the interaction between categories and get a matrix of the interaction between categories.

First, assume that the probability of users being infected by microblog  $x$  under  $K$  microblogs. This probability can be transformed by Bayesian formula. So we can get a new formula.

$$P(X|\{Y_k\}_{k=1}^K) = \frac{P(X)P(\{Y_k\}_{k=1}^K|X)}{P(\{Y_k\}_{k=1}^K)} = \frac{\prod_{k=1}^K P(X|Y_k)}{P(X)^{K-1}}$$

$$\therefore P(X|Y_k) \approx P(X) + \Delta_{count}(X, Y_k)$$

Where  $p(x|y)$  represents the probability of forwarding  $X$  when the content of  $Y$  is seen. This probability is approximately equal to the probability of forwarding  $x$  plus the interaction between  $X$  and  $Y$ . The interaction between  $X$  and  $y$  can be written as:

$$\Delta_{count}(X, Y_k) = \sum_t \sum_s M_X[t] * \Delta_{cluser}(C_t, C_s) * M_{Y_k}[s]$$

The two multipliers on the right side of the equation indicate the probability that Weibo belongs to  $s, t$  respectively. The middle multiplier is the category competition matrix of category  $t$  and category  $s$ .

Therefore, the loss function of the competition matrix can be obtained first, and then the competition matrix can be trained by the method of random gradient descent. After learning the competition matrix, we can use the competition matrix to predict the heat.

---

## 2. Related Concepts

---

### 2.1 Strong Ties

The most frequent contact is with their relatives, classmates, friends, colleagues, etc. This is a very stable but limited social cognition, which is a strong ties phenomenon.

The scope of weak ties is wider. It is possible for students, friends, relatives and friends, etc., that is, there are less opportunities for communication and interaction. For example, friends who don't often play together, relatives who don't often play together, etc.

The strong ties usually means that the actors have a high degree of interaction with each other, and they are close in some existing interaction patterns. Therefore, the messages generated through the strong ties are usually repetitive and easy to form a closed system.

Compared with the strong ties, the weak ties are more able to deliver non repetitive messages among different groups, so that members of the network can increase the opportunity to modify their original views.

For example, a blogger who joins a microblog has published several microblogs, among which the users who forward and comment on one microblog basically pay attention to the blogger's fans (these users are strong connected users). The other users who forward and comment on Weibo not only have fans of the user, but also many other users (these users are also weak connected users). Then it can be preliminarily determined that the second microblog has stronger competitiveness than the first one.

---

## 2.2 User Competition Matrix

The competition matrix of topic category interaction is proposed in imm. But in the same way, users are also of great significance to the competitive communication of topics. Therefore, the competition matrix of user's role on topic is proposed.

To get the user competition matrix, we need to introduce a new concept which calls probe user concept: probe user means stable and active user. For example, in all users of microblog, the users with the top 300 comments are used as probe users to avoid errors caused by zombie powder and inactive users.

Users' attention ability is limited (which is also the basis of topic competition communication), so when a large number of users comment on topics that do not belong to their preferred category, it indicates that the topic has strong competitiveness (user preference category vector needs to be extracted).

## 3. Model building

---

### 3.1 Text Classification

Word vectors are generated by word2vec, and then classified by CNN text classification. Then get the topic category vector.

#### 3.1.1 Word2vec

Word2vec has two structures: CBOW and Skip-gram. Skip-gram structure is used here.

```
from gensim.models.word2vec import Word2Vec
def word2vec1():
    file = open('sentence.txt') sss=[]
    while True:
        ss=file.readline().replace('\n','').rstrip()
        if ss=='': break
        s1=ss.split(" ")
        sss.append(s1) file.close()
    model = Word2Vec(sss, size=200, workers=5)
    build_vocab(sss) model.train(sss, total_examples = model.corpus_count, epochs
    = model.iter) model.save('/weibo2vec_model')
```

#### 3.1.2 CNN Classification

```
def get_data(fold):
    dax = []
    day = []
    for fname in os.listdir(fold):
        fpath = os.path.join(fold, fname)
        split_file_path = os.path.join(fold, fname)
        with open(split_file_path, "rb") as f:
            n1 = 0
```

```

        for line in f:
            if(nl>6000):
                break
            nl += 1
            dax.append(line)
            day.append(int(fname[:-4]))
    datalen = len(dax)
    train_len = int(datalen*0.8)
    day = to_categorical(day, nb_classes=None)
    temperper = np.random.permutation(datalen)
    trx = np.array([dax[i] for i in temperper[:train_len]])
    try1 = np.array([day[i] for i in temperper[:train_len]])
    tex = np.array([dax[i] for i in temperper[train_len:]])
    tey = np.array([day[i] for i in temperper[train_len:]])
    return ((trx, try1),(tex, tey))

def nsh_tra(maxlen=400,
            max_features = 50000,
            embedding_dims = 50,
            nb_filter = 250,
            batch_size = 32,
            filter_length = [3,4,5],
            hidden_dims = 250,
            nb_epoch = 2):
    (xtr, y_train), (xte, y_test) = get_data("../split_weibo")
    token = Tokenizer(nb_words=max_features, filters=base_filter(),
                      lower=True, split=" ")
    token.fit_on_texts(xtr)
    cPickle.dump(token, open("tokenfile", "wb"))
    # make up zero if it's not long enough, cut off if it's too long
    xtr = token.texts_to_sequences(xtr)
    xte = token.texts_to_sequences(xte)
    xtr = sequence.pad_sequences(xtr, maxlen=maxlen)
    xte = sequence.pad_sequences(xte, maxlen=maxlen)
    model = Sequential()
    minput = Input(shape=(maxlen, ), name='minput')
    embedding = Embedding(max_features,
                          embedding_dims,
                          input_length=maxlen,
                          dropout=0.2)(minput)
    convs = []
    for i in filter_length:
        conv = Convolution1D(nb_filter=nb_filter,
                             filter_length=i,
                             border_mode='valid',
                             activation='relu',
                             subsample_length=1)(embedding)
        pool_layer = MaxPooling1D()(conv)
        flatten = Flatten()(pool_layer)
        convs.append(flatten)
    out = Merge(mode='concat', name="merge_name")(convs)
    # add a vanilla hidden layer:
    dense1 = Dense(hidden_dims)(out)
    dropout1 = Dropout(0.2)(dense1)
    activation1 = Activation('relu')(dropout1)
    # project onto a single unit output layer, and squash it with a sigmoid:
    dense2 = Dense(15)(activation1)
    activation2 = Activation('softmax')(dense2)

```

```

model = Model(input=mininput, output=activation2)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
plot(model, to_file='model.png', show_shapes=True)
model.fit(xtr, y_train,
        batch_size=batch_size,
        nb_epoch=nb_epoch,
        validation_data=(xte, y_test))
json_string = model.to_json()
fs = open("weibo_model.json", "w")
fs.write(json_string)
fs.close()
model.save_weights('weibo_weights.h5')
return model

def nsh_pre(filepath='weibo_model.json', maxlen=400):
    fs = open("weibo_model.json", "r")
    json_string = fs.read()
    fs.close()
    model = model_from_json(json_string)
    model.load_weights('weibo_weights.h5')
    token = cPickle.load(open("tokenfile", 'rb'))
    title = draw_data()
    title_item = title.get_title_data('2020-04-22', '2020-4-28', 0)
    id2text = []
    for ti in title_item:
        id2text.append([i['_id']], list(jieba.cut(ti['title_text'])),
[ti['title_content']]))
    arrayid2text = np.array(id2text)
    predictData =
sequence.pad_sequences(token.texts_to_sequences(arrayid2text[:, 1]),
maxlen=maxlen)
    result = model.predict(predictData)
    result_dict = dict()
    for i in range(id2text):
        print(id2text[i][2] + str(result[i]))
        result_dict[id2text[i][0]] = result[i]
    np.savez("classify.dict", result_dict)

```

## 3.2 Probe User

Each user in Weibo has its own ID. for example, a certain percentage of users with the first total comments in a certain period of time (4.22-4.28) are taken as probe users.

```

def prouser(prousernum, dicttoid, percent):
    import operator
    prouser = {}
    if percent == '%':
        templist = []
        templist = sorted(dicttoid.iteritems(), key =
operator.itemgetter(1), reverse=True)
        changepercent = int(len(dicttoid) * prousernum // 100)
        for i in range(changepercent):
            prouser[templist[i][0]] = None
    else:

```

```

    for i in dicttoid.keys():
        if dicttoid[i]>=prousernum:
            prouser[i] = None
    return prouser

```

### 3.3 SH&Strong Ties Model

```

def regression():
    each_item = np.load("SHitem.npz")['arr_0'][(0)]
    xda = []
    yda = []
    save_id = each_item.keys();
    for i in save_id:
        if len(each_item[i]) == 4:
            xda.append(each_item[i][0:-1])
            yda.append(each_item[i][-1])
    n_sample = len(xda)
    sidx = np.random.permutation(n_sample)
    n_train = int(np.round(n_sample * 0.3))
    texset = np.log10(np.array([xda[s] for s in sidx[:n_train]]+1))
    teyset = np.array([yda[s] for s in sidx[:n_train]])
    test_id = [save_id[s] for s in sidx[:n_train]]
    trxset = np.log10(np.array([xda[s] for s in sidx])+1)
    tryset = np.log10(np.array([yda[s] for s in sidx]))
    train_id = [save_id[s] for s in sidx]
    rg = linear_model.LinearRegression()
    rg.fit(trxset, tryset)
    prey = np.power(10, rg.predict(texset))
    trypre = rg.predict(trxset)
    evs = explained_variance_score(teyset, prey)
    rmse = np.sqrt(mean_squared_error(teyset, prey))
    logging.info("rmse:"+str(rmse)+" evs:"+str(evs))
    logging.info("mape:"+str(mape(prey, teyset))+ "%")
    draw_same(teyset, prey)
    dict_id_pred = dict()
    for i in range(len(train_id)):
        dict_id_pred[train_id[i]] = [trypre[i], tryset[i]]
    for i in range(len(test_id)):
        dict_id_pred[test_id[i]] = [prey[i], teyset[i]]
    np.savez("SH_pred", dict_id_pred)
    logging.info(u", done!")

```

### 3.4 Competetion Model

```

def targetIn(bitem, titem):
    temp = titem * 0.2
    less_line = titem - temp
    up_line = titem + temp
    if bitem >= less_line and bitem <= up_line:
        return 1
    else:
        return 0

```

```

def train_data(queue, birds, level=3000):
    allexample = []
    bestsum = 0.0
    for listi in queue:
        for idnew in listi:
            if dict_id_pred[idnew][0] > level:
                allexample.append([idnew, dict_id_pred[idnew][0],
dict_id_pred[idnew][1]])
                bestsum += (abs(dict_id_pred[idnew][0] - dict_id_pred[idnew]
[1])/dict_id_pred[idnew][1])
            if len(allexample) > 0:
                pso = PSO(fitFunc=fitFunc, birdNum=1500, c1=3, c2=2, solutionSpace=100,
extend=allexample, birds=birds)
                lBestPosition, birds, lBestFit = pso.solve(300)
                return lBestPosition, birds, lBestFit, bestsum/len(allexample)
    return None, birds, 0, 0

def predict_data(position, predict_example, level=3000, true_predict = []):
    #print position
    basesum = 0.0
    predsum = 0.0
    for i in predict_example:
        if dict_id_pred[i][0] > level:
            new_temp = classify_dict[i].reshape(1, 15)
            pred_new = dict_id_pred[i][0] + np.dot(np.dot(new_temp, position),
new_temp.T)
            true_predict.append([pred_new[0][0], dict_id_pred[i][0],
dict_id_pred[i][1]])
            predsum += (abs(pred_new[0][0] - dict_id_pred[i][1])/dict_id_pred[i]
[1])
        else:
            true_predict.append([dict_id_pred[i][0], dict_id_pred[i][0],
dict_id_pred[i][1]])
            predsum += (abs(dict_id_pred[i][0] - dict_id_pred[i]
[1])/dict_id_pred[i][1])
            basesum += (abs(dict_id_pred[i][0] - dict_id_pred[i][1])/dict_id_pred[i]
[1])

    if len(predict_example)>0:
        return true_predict

def drawPosition(dayposition, daytime, level, k):

    topic_name = [u'汽车',u'财经',u'科技',u'健康',u'体育',
u'旅游',u'教育',u'文化',u'军事',u'社会',
u'国内',u'国际',u'游戏',u'娱乐',u'时尚']

    plt.matshow(dayposition)
    np.savetxt("interaction "+str(daytime+datetime.timedelta(days = (-k+1)))+
to "+str(daytime+datetime.timedelta(days = 1))+" level="+str(level)+".csv",
dayposition,
delimiter=', ')
    plt.title("interaction "+str(daytime+datetime.timedelta(days = (-k+1)))+
"+str(daytime+datetime.timedelta(days = 1))+" level="+str(level), fontsize=18)
    plt.xticks(np.arange(15), topic_name, fontsize=16, rotation=30)
    plt.yticks(np.arange(15), topic_name, fontsize=16, rotation=30)
    v = np.linspace(-1000,1000, endpoint=True)
    plt.colorbar(ticks=v)
    # plt.show()

```

```

plt.savefig(str(daytime)+" level="+str(level)+".png", dpi=200)

def plot_3d(dateX, travel):

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    X = mdates.date2num(np.array(dateX))
    Y = np.arange(0, 15, 1)

    xnew = np.linspace(X.min(), X.max(), 6*len(X))
    ynew = np.linspace(Y.min(), Y.max(), 3*len(Y))

    z = travel.T
    Z = np.zeros(shape=(0,6*len(X)))
    for i in range(z.shape[0]):
        z_smooth = spline(X, z[i], xnew)
        Z = np.insert(Z, Z.shape[0], values=z_smooth, axis=0)
    znew = np.zeros(shape=(3*len(Y),0))
    for i in range(Z.shape[1]):
        z_smooth = spline(Y, Z[:,i], ynew)
        znew = np.insert(znew, znew.shape[1], values=z_smooth, axis=1)
    xnew, ynew = np.meshgrid(xnew, ynew)

    topic_name = [u'汽车',u'财经',u'科技',u'健康',u'体育',
                  u'旅游',u'教育',u'文化',u'军事',u'社会',
                  u'国内',u'国际',u'游戏',u'娱乐',u'时尚']
    yearsFmt = mdates.DateFormatter('%Y-%m-%d')
    ax.xaxis.set_major_formatter(yearsFmt)
    surf = ax.plot_surface(xnew, ynew, znew, rstride=1, cstride=1,
                           cmap=plt.cm.jet)

    plt.yticks(np.arange(15), topic_name, fontsize=15, rotation=40)

    fig.colorbar(surf)

    plt.show()

def get_data():
    draw_mysql = draw_data()
    rmses = []
    mapes = []
    for level in range(500, 3500, 300):
        allposition = np.zeros(shape=(15, 15))
        begin_time = datetime.datetime.strptime('2020-04-22', '%Y-%m-%d')
        end_time = datetime.datetime.strptime('2020-04-28', '%Y-%m-%d')
        queue = []
        for i in range(3):
            print begin_time
            temp = draw_mysql.get_title_data(str(begin_time), str(end_time), 0)
            today_day = []
            for ti in temp:
                if ti['_id'] in dict_id_pred and ti['_id'] in classify_dict:
                    today_day.append(ti['_id'])
            queue.append(today_day)
            begin_time = end_time

```



```

        end_time = end_time + datetime.timedelta(days = 1)
birds = None
pred_position = None
true_predict = []
lbestFitList = []
baseFitList = []
xdate = []
travel = np.zeros(shape=(0, 15))
for i in range(15):#87
    xdate.append(begin_time+datetime.timedelta(days = -1))
    temp = draw_mysql.get_title_data(str(begin_time), str(end_time), 0)
    today_day = []
    for ti in temp:
        if ti['_id'] in dict_id_pred and ti['_id'] in classify_dict:
            today_day.append(ti['_id'])
    birds = None
    temp_position, birds, lbestfit, basefit = train_data(queue, birds,
level)
        if temp_position != None:
            lbestFitList.append(lbestfit)
            baseFitList.append(basefit)
            pred_position = temp_position
            travel = np.insert(travel, travel.shape[0],
values=temp_position[9], axis=0)
            allposition = allposition + pred_position
            print begin_time+datetime.timedelta(days = -1)
            # drawPosition(pred_position, begin_time+datetime.timedelta(days
= -1), level, 3)
            true_predict = predict_data(pred_position, today_day, level,
true_predict)
            queue.pop(0)
            queue.append(today_day)
            begin_time = end_time
            end_time = end_time + datetime.timedelta(days = 1)
            np.savetxt("allposition"+str(level)+".csv",
allposition/len(xdate),
delimiter=', ')
            drawPosition(allposition/len(xdate), begin_time+datetime.timedelta(days
= -1), level, 0)
            mape = 0.0
            rmse = 0.0
            for i in true_predict:
                # mape += targetIn(i[0], i[2])
                mape += abs(i[0] - i[2])/i[2]
                rmse += (i[0]-i[2])**2
            mape /= len(true_predict)
            rmse /= len(true_predict)
            rmse = np.sqrt(rmse)
            rmse.append(rmse)
            mape.append(mape)
            basemape = 0.0
            basermse = 0.0
            for i in true_predict:
                # basemape += targetIn(i[1], i[2])
                basemape += abs(i[1] - i[2])/i[2]
                rmse += (i[1]-i[2])**2
            basemape /= len(true_predict)
            basermse /= len(true_predict)

```

```

basermse = np.sqrt(basermse)
plt.plot(range(1000, 4000, 300), mapes, label="mape")
plt.plot([1000, 4200], [basemape, basemape], label="basemape")
np.savetxt("rmse.csv", rmases, delimiter=', ')
np.savetxt("mape.csv", mapes, delimiter=', ')
plt.show()

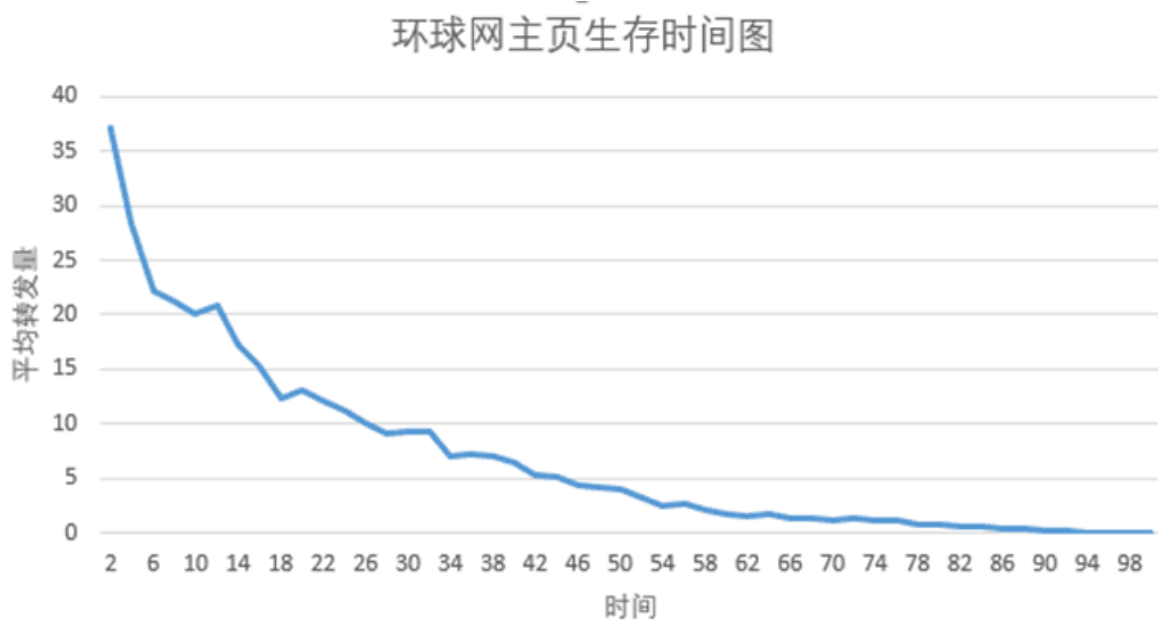
```

## 4.Result

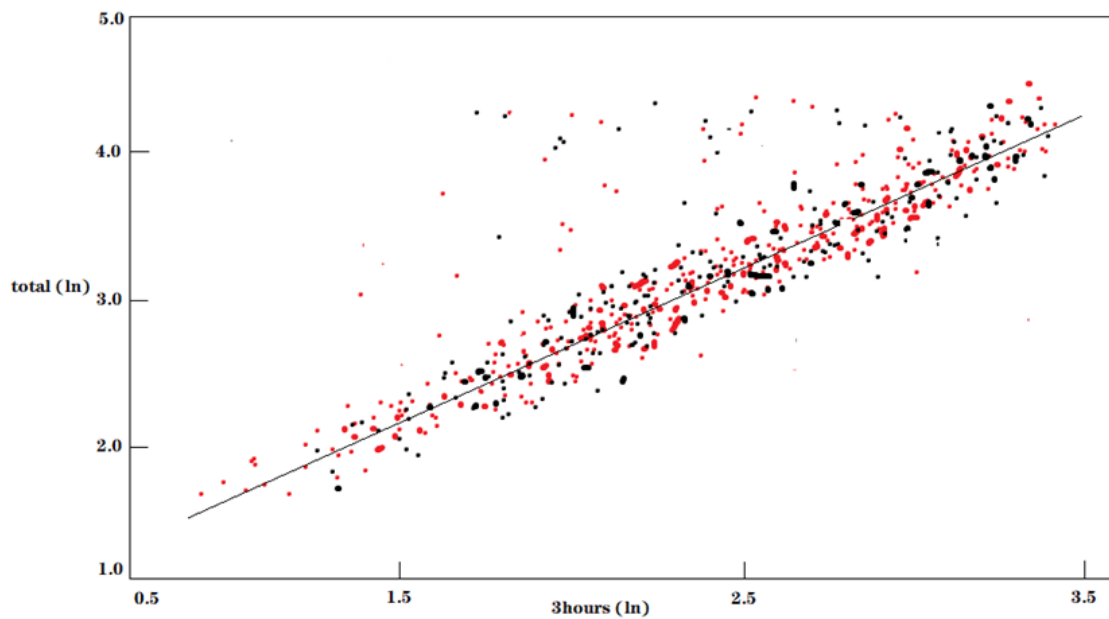
I crawled through all the tweets posted by seven bloggers in the Weibo dataset between April 22 and April 28. These seven bloggers have more than 10 million fans, and their microblogs usually belong to different sectors.

### 4.1 SH&SH+Strong Ties Model

First, we can get the average survival time of each microblog.

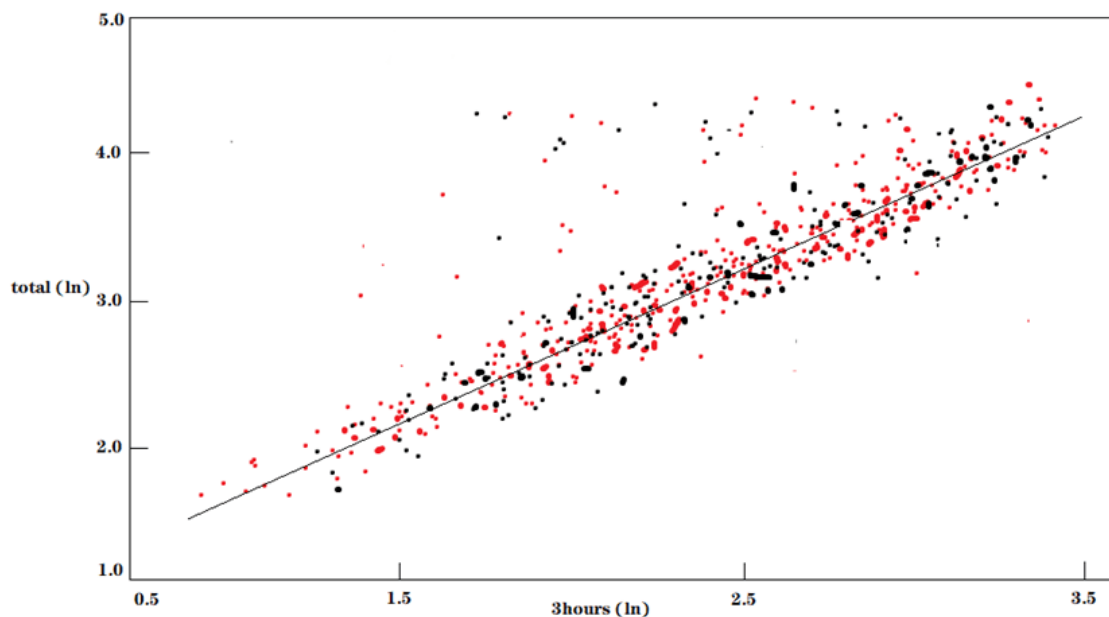


We can find that the average survival time of each microblog is about 100 hours. Therefore, we can define the first three hours as the initial time, and use sh model to predict the heat degree. The result is as shown in the figure (the red dot is the training set and the black dot is the test set):



It can be seen from the figure that these points are basically linear. At the same time, some points can be found, which are special cases. At the initial stage of microblogging, there was no attention, but after a period of time, the popularity will skyrocket.

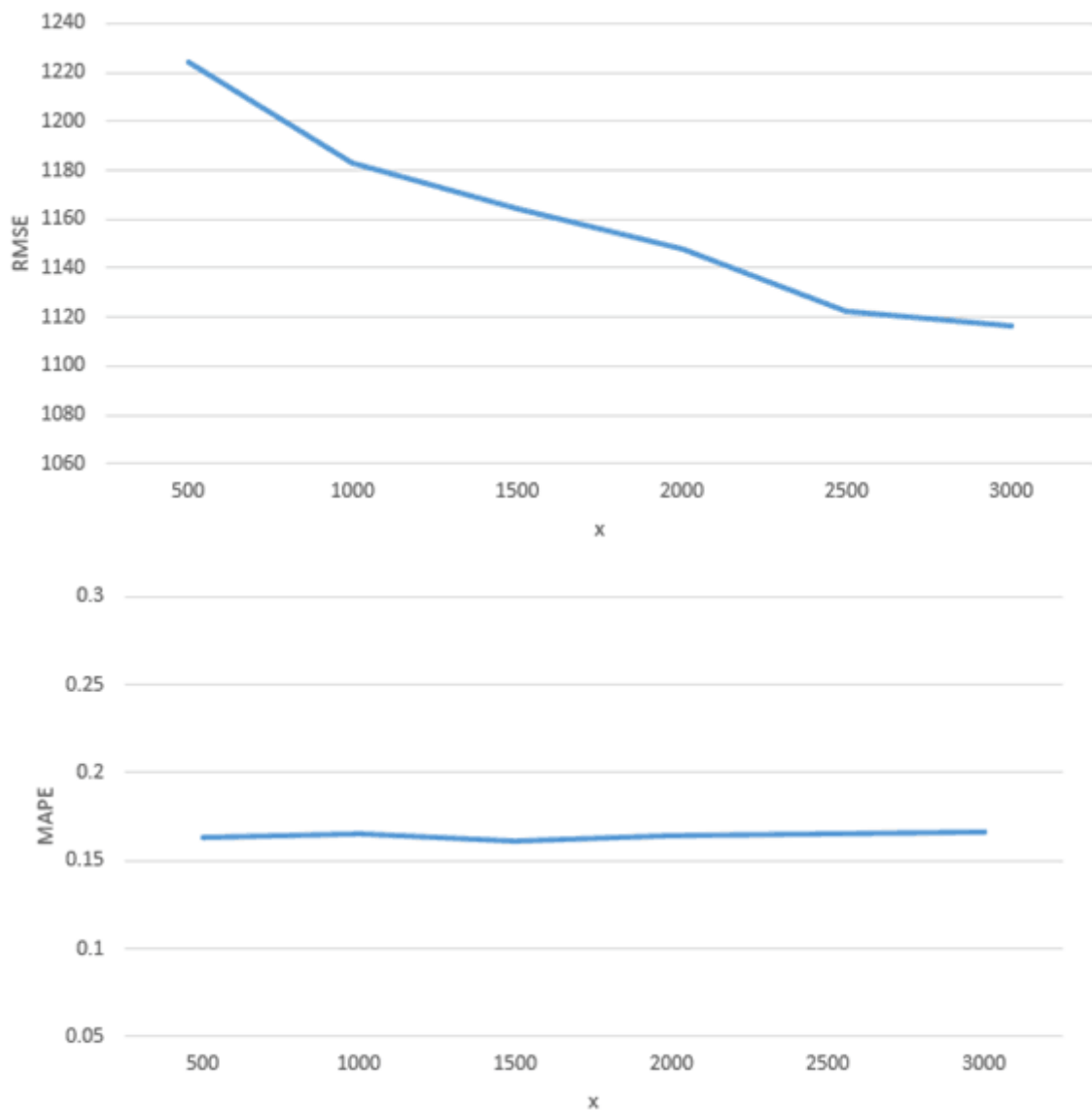
Now we can get a new graph by adding strong ties.



As can be seen from the figure, the obvious point aggregation is better. Therefore, it can be preliminarily judged that strong connection is helpful for the improvement of the model.

## 4.2 Competition Model

Select microblogs with more than  $x$  comments as training samples. Because the competition usually takes place between popular microblogs. The hot topics (micro blog with small comments) are relatively stable, and the hot topics are not easy to be seized, nor other hot topics. Therefore, it will increase the error. This is also why RMSE decreases with the increase of  $X$ , while MAPE (mean absolute percentage error) is basically unchanged.



## 4.3 Contrast

Model	RMSE	MAPE
SH	2693.015	39.482%
SH+强连接	1552.598	17.948%
竞争矩阵 (X=1000)	1183.103	16.520%
竞争矩阵 (X=2000)	1147.690	16.423%
竞争矩阵 (X=3000)	1116.248	16.632%

It can be seen from the table that the competition matrix (user impact on category) model is better than the SH + strong ties model. And SH + strong ties model is better than SH model.

## 5. Summary

This assignment let me learn how to predict the topic heat and how to predict the multiple topic competition and communication heat. But unfortunately, because I have only one person, although I tried to put forward my own method, I failed. So I can only make some improvement on the existing model. Thank you, teacher Fu and the TA for this course. teacher Fu and the elder and elder students who have explained all kinds of knowledge to us are very responsible. At the

same time, they are very patient, which makes me feel a lot of more interesting things in the computer field.

---

## 6. Reference

---

- [1] Xu T, Xu M, Ding H. BBS Topic's Hotness Forecast Based on Back-Propagation Neural Network[C]//Web Information Systems and Mining (WISM), 2010 International Conference on. IEEE, 2010, 1: 57-61.
- [2] Bakshy E, Hofman J M, Mason W A, et al. Everyone's an influencer: quantifying influence on twitter[C]//Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011: 65-74.
- [3] Leskovec J, McGlohon M, Faloutsos C, et al. Patterns of cascading behavior in large blog graphs[C]//Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007: 551-556.
- [4] Granovetter M. Threshold models of collective behavior[J]. American journal of sociology, 1978, 83(6): 1420-1443.
- [5] Schelling T. Micromotives and macromotives[J]. 1978.
- [6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.
- [7] Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network[C]//Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003: 137-146.
- [8] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.
- [9] Hethcote H W. The mathematics of infectious diseases[J]. SIAM review, 2000, 42(4): 599-653.
- [10] May R M, Lloyd A L. Infection dynamics on scale-free networks[J]. Physical Review E, 2001, 64(6): 066112.
- [11] Yang J, Leskovec J. Modeling information diffusion in implicit networks[C]//Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010: 599-608.
- [12] The economy as an evolving complex system, III: Current perspectives and future directions[M]. Oxford University Press, 2005.
- [13] Kreindler G E, Young H P. Rapid innovation diffusion in social networks[J]. Proceedings of the National Academy of Sciences, 2014, 111(Supplement 3): 10881-10888.
- [14] Bi Y, Wu W, Zhu Y. Csi: Charged system influence model for human behavior prediction[C]//Data Mining (ICDM), 2013 IEEE 13th International Conference on. IEEE, 2013: 31-40.
- [15] Weng L, Flammini A, Vespignani A, et al. Competition among memes in a world with limited attention[J]. Scientific reports, 2012, 2.
- [16] Beutel A, Prakash B A, Rosenfeld R, et al. Interacting viruses in networks: can both survive? [C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012: 426-434.
- [17] Karrer B, Newman M E J. Competing epidemics on complex networks[J]. Physical Review E, 2011, 84(3): 036106.

- [18] Myers S A, Leskovec J. Clash of the contagions: Cooperation and competition in information diffusion[C]//Data Mining (ICDM), 2012 IEEE 12th International Conference on. IEEE, 2012: 539-548.
- [19] Song X, Lin C Y, Tseng B L, et al. Modeling and predicting personal information dissemination behavior[C]//Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005: 479-488.
- [20] Jamali S, Rangwala H. Digging digg: Comment mining, popularity prediction, and social network analysis[C]//Web Information Systems and Mining, 2009. WISM 2009. International Conference on. IEEE, 2009: 32-38.
- [21] Chen J, Yu J, Shen Y. Towards topic trend prediction on a topic evolution model with social connection[C]//Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society, 2012: 153-157.
- [22] XuanYu Modeling and prediction of multi topic competitive communication 2017-6
- [23] Zhang H, Zhao B, Zhong H. Hot trend prediction of network forum topic based on wavelet multi-resolution analysis[J]. Computer Technology and Development, 2009, 4: 021. [24] Zhao K, Zhang Y, Li B, et al. Repost Number Prediction of Micro-blog on Sina Weibo Using Time Series Fitting and Regression Analysis[C]//Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015 International Conference on. IEEE, 2015: 66-69.
- [25] Szabo G, Huberman B A. Predicting the popularity of online content[J]. Communications of the ACM, 2010, 53(8): 80-88.
- [26] Gómez V, Kaltenbrunner A, López V. Statistical analysis of the social network and discussion threads in slashdot[C]//Proceedings of the 17th international conference on World Wide Web. ACM, 2008: 645-654.
- [27] Cheng H, Liu Y. An online public opinion forecast model based on time series[J]. Journal of Internet Technology, 2008, 9(5): 429-432.
- [28] Granovetter M S. The strength of weak ties[J]. American journal of sociology, 1973, 78(6): 1360-1380. [29] Yakubovich V. Weak ties, information, and influence: How workers find jobs in a local Russian labor market[J]. American sociological review, 2005, 70(3): 408-421.
- [30] Montgomery J D. Job search and network composition: Implications of the strength-of-weak-ties hypothesis[J]. American Sociological Review, 1992: 586-596.
- [31] Henning C, Lieberg M. Strong ties or weak ties? Neighbourhood networks in a new perspective[J]. Scandinavian Housing and Planning Research, 1996, 13(1): 3-26.
- [32] Granovetter M. The strength of weak ties: A network theory revisited[J]. Sociological theory, 1983: 201-233. [33] Kavanaugh A L, Reese D D, Carroll J M, et al. Weak ties in networked communities[J]. The Information Society, 2005, 21(2): 119-131. [34] Hansen M T. The search-transfer problem: The role of weak ties in sharing knowledge across organization subunits[J]. Administrative science quarterly, 1999, 44(1): 82-111.
- [35] Friedkin N E. Information flow through strong and weak ties in intraorganizational social networks[J]. Social networks, 1982, 3(4): 273-285.
- [36] Weimann G. The strength of weak conversational ties in the flow of information and influence[J]. Social Networks, 1983, 5(3): 245-267.
- [37] Centola D, Macy M. Complex contagions and the weakness of long ties 1[J]. American journal of Sociology, 2007, 113(3): 702-734.

[38] Bakshy E, Rosenn I, Marlow C, et al. The role of social networks in information diffusion[C]//Proceedings of the 21st international conference on World Wide Web. ACM, 2012: 519-528.

[39] Pan R K, Saramäki J. The strength of strong ties in scientific collaboration networks[J]. EPL (Europhysics Letters), 2012, 97(1): 18007.

[40] Onnela J P, Saramäki J, Hyvönen J, et al. Structure and tie strengths in mobile communication networks[J]. Proceedings of the National Academy of Sciences, 2007, 104(18): 7332-7336. [41] By Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, Alessandro Provetti. On Facebook, Most Ties Are Weak. Communications of the ACM, Vol. 57 No. 11, Pages 78-84