Hw2 report
成員:
r01922164 李揚 r03922032 王珮恂 r02922062 郭冠呈

Notable Results:

Method	F1	MAP
common_neighbor	0.072296	0.103799
Approximate Katz	0.076751	0.115088
Learning(RF)	0.059909	0.078108
adamic	0.076740	0.113445
adamic + Approximate katz	0.792000 /0.120050 (after postprocessing)	0.120050
根據使用者目的的推薦 (based on adar + Approximate katz)	0.80878 /0.121770 (after postprocessing)	0.119371
MF optimizes AUC for (girls or boys)	0.047372	0.057914
Model ensemble for F1	**0.123287**	X
Model ensemble for MAP	X	**0.120903**

Summary of the notable results:
開賽前幾天，我們嘗試了許多baseline，最後發現跟common neighbor有關的baseline最好，之後發現Adar,katz又是其中佼佼者。之後用了learning，可惜效果不彰。
之後公布column名稱之後，我們發現異性link佔超過60%，我們又衍生出一些比較不一樣的想法。
如:

1. 根據使用者目的的推薦
2. 對男生推薦女生，對女生推薦男生

最後因為我們擁有眾多prediction，我對他們做了一些合理的組合，並且照data distribution去切F1 score該切的點
第一點稍有進步，但第二點沒這麼有用。以下我們會按照這樣的脈絡做各方法的簡單介紹。
如果有興趣知道我們所有的方法的話，以下次我們全部傳的quota和相對應的介紹:
https://docs.google.com/document/d/1H0b4AUzNdl2d_u1haK8OgQQ-kOHSRMOHc1rZOp3K8io/edit (同資料夾下quota.pdf)

baseline

在這部分，我們挑我們最好的兩個結果介紹，分別是common neighbor和adar。

1. Common_Neighbor:

去找節點的共同朋友數，然後把成績最高的前三十名照順序變成最後上傳的檔案。
根據我們測試(嘗試過shortest path,page rank, MF)，這個方法是其中次好的baseline。
但他也以下缺點:

Cold start nodes:
在這張圖裡面，有不少767個節點是不在training edges 裡面的。這些節點我們根本就無法去找common neighbor，甚至還有一些節點往外連的點並不多，也難以去找common neighbor。而這種情形，會發生在所有edges based methods，所以我們之後一直嘗試著用node based methods來彌補。

同分:
依照這common neighbor的比較方式，會有非常多節點擁有同樣的common neighbor數，我們無法把這些先後順序分出來。為了解決這個問題，我們之後也採去各種不同的精進方法。
ex:katz，從node file下手、community detection。

2. Adamic

跟common neighbor 很像的觀念，但我們改用老師教的Adamic演算法，結果有了進步。
MAP: 0.103799 => 0.113445
F1: 0.072296 => 0.076740

除此之外，以上兩種演算法如果遇到同分情形或cold start的情形，我們都用pre_nodes_profile的資訊來判斷。我們用以下章節做說明。

Feature Selection Using Random Forest:

由於一開始我們並不知道各個column的意義。我們要做feature selection。
首先，我們得先製造features，我們方法參考於一篇發表在2010年ACM recsys的論文

"RECON: a reciprocal recommender for online dating"。這篇論文有提到一個compatity的算法。我們用類似的概念，對於category的columns，我們看一樣的交集數，對於數值類的columns，我們用相減的絕對值，命名為feature X。

我們用Random forest去學，把學習時切割選擇features的值做統計，我們發現以下column最為重要。

1. feature 4 (0.641303)
2. feature 45 (0.091058)
3. feature 10 (0.048559)
4. feature 6 (0.044510)
5. feature 23 (0.040930)
6. feature 38 (0.038806)
7. feature 22 (0.022905)
8. feature 2 (0.020837)

所以我們之後對於cold start的nodes，都會用feature 4來比較。而在這裡，我們得到的RF的model效果並沒有很好，於是暫時不用。後面在製造一些更有用的features時，有重拾learning的手段，這點會在Learning的章節作介紹

Learning

1. 針對testing node的negative sampling:
   我們認為，既然要測試的就只有testing node。那其他的node加進來可能反而會造成noise。所以我們的第一個策略就是，只用各個testing node與他的鄰居的連結當作positive instance，之外，對於每一個testing node都sample與他鄰居同樣多的negative instance(為了平衡)，然後用這些資料去訓練我們的模型。

   我們用了Random Forest下去做訓練，造了以下的features
   1. 第二層Katz(common neighbor)
   2. 第三層Katz(下一章節會提到)
   3. column_4(這是我們針對pre_node_profile做feature selection發現最好的features，把他們的交集數當作features)
   4. 是否為異性
   5. 是否為都是0性
   6. 是否為都是1性

   我們先針對以上六個features去做訓練。

   難題:Prediction:
   我覺得這次作業最困難的就是prediction，因為我要去猜10萬* 1萬次的次數，再加上Katz要算很久，所以我之前把它存成sparse matrix(但sparse matrix 的access time也會是一個問題)，所以predict完會花很久的時間。所以要很有耐心。

   補充:為什麼我們用Random Forest?
   1. 解讀性高，我希望train 完之後我們可以發現各個model覺得重要的features有何不同，藉此做下次的改進
   2. Nonlinearity。我們想讓他學到Nonlinearity。以common neighbor為例，1跟2應該不會差很多，但0跟1就會差很多，我們想要用RF看著資料幫我們切這一刀。

結果:

	MAP	F1

all candidates	0.078108	0.059909
candidate has common neighbor	0.077783	0.059966

結論:learning model有點悽慘，個人猜測是因為我們sampling做得不好。

但在這裡，我們有一個意外的發現，就是第三層的katz在我們學習過程中其實是比common neighbor更為重要的，我們也為了這個疑點，在下面繼續做一些分析與研究。

Approximate Katz:

我們用了Katz score，但由於中間的反矩陣運算會花費N^3的時間，且在這裡N約等於十萬，所以我們不可能算出真正的Katz score。
所以我必須要挑我覺得比較重要的層數，因為某幾個特別的features(ex: column_46)，我們認為這個社群網路是一個"交友網站"，實際上我們也統計出了男女在建立連結的分布，如下(因為我不知道0或1代表男或女，所以我直接用0跟1表示):

00 (都是0)	220711
11 (都是1)	174855
opposite (異性)	628773

我們可以發現異性比同性高了1.6倍，而一般社群網路應該是同性會比較多才對。而很不幸的，common neighbor會推薦的幾乎是同性。(因為大部分人跟異性建立朋友，異性的異性會是同性)。
也因此，我們猜測，走三步的katz很重要(異性的異性的異性又會是異性)，而且跟自己有關連性，實際上我們有做這個上傳，也證實確實不差，甚至接近於common neighbor了，結果如下。

層數	MAP	F1
第二層(common neighbor)	0.103799	0.072296
第三層	0.0917603	0.065386

然後我們猜測混和第二層和第三層可以增加我們的prediction，於是我們挑不同的Betta，去算只算到第三層的katz score，回傳前30高的，結果如下:

Betta	MAP	F1
0.01	0.114906	0.0762189

0.02	0.115388	0.076580
0.1	0.111505	0.075040

補充:Katz的難題(記憶體配置):
Katz最麻煩的就是他在做矩陣乘法時需要給他一大塊記憶體，我在這邊用sparse matrix處理，並且留下mapping對應，但隨著層數增加，sparsity會降低，再度造成記憶體負擔。因此，我只留下testing nodes存在的row，並留下其mapping。

Approximate Katz + Adamic(exp_0.py):

經過前面的實驗，我們已經發現Katz的三層，可以提升原本只用第二層(common neighbor)。
於是我們嘗試是否可以利用Adamic加上Katz的第三層，來提升最後的結果。
最後結果是可以的:)
一樣我們給第三層的Katz不同的weight，把他跟adamic去做組合。

	MAP	F1
0.08	0.120072	0.079200
0.1	0.120050	0.079163

建立於使用者性別偏好的推薦(exp_1.py):

在上面的我們提到，其實我們猜測這是一個交往網站。而且我們發現異性邊比同性邊多1.6倍。我們繼續探勘後發現，有3578個人，異性link的比例高於0.7。也就是說，大約有36.78%的人都是來這裡想認識異性的。但也有些人認識異性的比例與同性差不多。這兩種人可能分開處理會得到比較好的結果，所以我們先做一個簡單的嘗試: 我們把每個人認識的性別比例算出來，然後照著這個比例去推薦相對應的男女比例。假設一個人在training set跟5個男生交朋友，1個女生交朋友，我就會在我推薦的30個人中，用25個男生，和5個女生。然後我們繼續維持上面Adamic + Katz第三層的框架，去找出最高的25個男生，和最高的5個女生。由於我沒有針對這一點做一個好的排序，純粹只是把這30個抓近來，照他的分數去做sort。於是我們的MAP沒有進步，但**F1有了進步。從原本的0.79200 進步到0.80878**。

接著，我們還有一個發現，以上我們都是用common neighbor based演算法，也確實得到相對比較好的結果，但common neighbor真的對異性和同性都是個好方法嗎?同性間可以很合理的想像，但因為異性間有common neighbor很有可能代表著三角戀的關係或其他不是這麼緊密的關係，所以不見得會是朋友。
於是我們去做了統計:

	link是common neighbor的比例	link間的平均common neighbor數	link有兩步neighbor的比例(第三層katz)	link的第三層katz的平均
同性	80.4%	4.246	98.13%	131.94
異性	54.5%	2.317	100%	205.37

做完這個統計後，我們有幾個發現:

1. common neighbor對同性非常有用，因為他們只要有link幾乎都是common neighbor，而且平均高達4.246個，非常有機會被前三十名挑中
2. 對於異性，common neighbor可能就沒這麼有用，反倒是第三層的katz可能比較有用，但在原本的model裡，我們給第三層katz的比重很小。

另外，我們針對之前的prediction做分析，果然，大部分都是猜同性，所以我們實際上只有解這個問題的40%，另外60%(異性邊)，還有機會有大幅的進步。

反應這份統計，我們做了兩個實驗:

1. 對異性推薦時用第三層katz，對同性用原本的方法
2. 男生女生的推薦式系統

雖然這兩種方法沒有導致直接的進步，但個人認為第二個方法是一個不錯的嘗試，只可惜比賽已接近尾聲，沒有時間繼續深入研究他。

嘗試:推薦式系統解決男女推薦問題(exp_3.py):

我們認為，比起共同朋友，跟你相似的人喜歡的異性，可能是再推薦異性時更好的依據。對於女生，我們把男生當作商品，對於男生，我們把女生當作商品，去做collaborative fitering。採用的模型是MF，但是是使用Matrix Factorization optimize AUC (參考 BPR: Bayesian personalized ranking from implicit feedback這篇在2009年發表在UAI的論文)。

雖然花了很多時間，但在這邊沒有得到太好的結果，我們猜測，我們還必須比social network的information加進去才行，像是老師上課所教的加social network 的資訊去提升推薦系統。

Postprocessing for F1 score:

取不同的數量的點進來，會大大影響到F1 score。所以我們開始思考如何在這方面做改良。

1.看著各點neighbor數的切割方法

由於我們知道助教在sample時的比例是80%:20%，然後是random去取樣。所以我們可以很合理的從training各個node中的edge數，去推估testing各個node的edge數。依照最合理的方法，我們根據每一個node的neighbor數，除以N(最合理應該是除四，因為sample

時4:1)，當作testing node數，然後用以上最好的方法，取作取樣。N理論上要離4很近，所以我們試了3,4,6，結果發現還是4最好。
另外，針對0個neighbor的nodes，我們還是會讓他猜一個，因為precision 為0就鐵定F1就鐵定是0了，還不如賭一個看看。針對超過30N個neighbor的nodes，我們因為限制也只猜30個。

N	MAP	F1
3	0.086754	0.115652
4	0.092213	0.115060
6	0.078792	0.1122589

藉由以上的處理，我們的F1 score從原本的0.076751進步到0.115652。
之後我們又把我們最好的結果去做這樣的切割，達到**F1 score在0.121770**的成績。

Ensemble the Prediction to Enhance the Performance(F1/MAP):
經過了一長串的實驗，我們產生了超過七十個的predictions。我們利用這些prediction，去做一些簡單的組合。
How we ensemble our model to enhance the F1 score?
對於F1 score，我們設計了三種方法去做組合，分別是combine, and,or，其中以or的效果最好。

1. combine: 去統計各個node在file出現的次數，之後再做一次sort，選剛剛所提到的training的1/4個
2. and:選N個file，他們中間的交集，當作submission
3. or: 選N個file，他們中間的聯集，當作submission

結果combine和and都沒有造成明顯的進步，但or有讓我們進步，從原本的 **0.121770進步到0.123287**

How we ensemble our model to enhance MAP?
我們發現，有幾個submission，F1較高，但MAP較低。有些則MAP較高，F1較低，所以我們合理推估: 當prediction A的F1比prediction B高，但MAP比B低。代表的是A在前段的prediction可能比B差，後段的prediction可能比B好。
我們利用這種方法去組合我們最好的結果，結果MAP可以從**0.120072進步到0.120903**。

另外，我們也嘗試過另外一種組合方法，把第一個位置當作30分，第二個位置當作29分...以此類推，去組合各個file，然後把每個點得到的分數總和記錄下來，最後再取出最高的30個節點。但這個方法並沒有導致太多的進步。

結論:

在這次比賽裡面，我們嘗試了很多方法，發現common neighbor based方法有比較好的結果(ex: katz, adamic)。我們也嘗試了learning，可惜並沒有學到更好的結果，但是我們利用learing的過程做feature selection，把重要的features用來猜cold start的使用者。
我覺得我們這次比賽最可惜的就是沒有利用到公佈name的這件事情，來大幅提升我們的model，雖然有很多想法，但並沒有造成確實的進步。
我們根據我們的prediction去做分析，我們在推薦異性的部分其實是比較糟糕的，或許，異性的連結比較難以共同朋友來詮釋，我們也在上面做了這樣的分析。
在大部分都是異性link的social network裡面，我們沒做到這點真的挺可惜的。

整理一下，我們這組作了以下的嘗試識大概如下:

1. Variety of Baselines taught in class (Adamic wins)
2. Approximate Katz (improve the adamic)
3. Learning (Fail … but provide some useful information)
4. BPRMF for Recommendation System (Fail.. but we think it's a nice try)
5. 根據使用者偏好的推薦 (Improve the F1 score, and provide variety of sources to ensemble)
6. Ensemble for F1 and MAP