

# Challenge SII Yelp Predictor

Andrea Iuliano - Valerio Cestarelli

# Problemi Riscontrati

- **Dataset contenente informazioni non consistenti**
  - **Matrice molto sparsa**
  - **Cold start problem**
- **Predizioni accurate richiedono tempi di calcolo onerosi**
- **Alcuni concetti teorici studiati, si sono rivelati inadatti al caso di studio**

# Modellazione

```
Utente{ String ID, int reviewCount, double averageStars,  
int countSameBusiness }
```

```
Business{ String ID, int reviewCount, double stars, int  
countSameUsers }
```

```
Review{ String userID, String businessID, int stars }
```

```
Matrix{ Map<userID, Map<businessID, rating>> }
```

# Classificazione

1. Parsing dei file .json
2. Popolamento del DataBase
3. Creazione degli Utenti e dei Business mancanti

# Predizione

## Strategia generale

```
if (user != null && business != null)
    return linearCombination(review, user, business)
else if (business != null)
    return approximate(business.getStars())
else if (user != null)
    return approximate(user.getAverageStars())
return AVERAGE_VALUE
```

# Predizione

## Linear Combination

$$\lambda P_u + (1 - \lambda) P_i$$

Effettuiamo sia una predizione **User-Based** che una **Item-Based**, calcolandone la **combinazione lineare**.

# Predizione User-Based

Dati un utente, un business ed il neighborhood, la predizione è effettuata tramite la seguente formula:

$$P_u(u, b, neigh) = avgstars_u + \frac{\sum_i^{neigh} commonvalue_i * (rating(i,b) - avgstars_i)}{\sum_i^{neigh} commonvalue_i}$$

# Predizione

## Item-Based

Dati un utente, un business ed il neighborhood, la predizione è effettuata tramite la seguente formula:

$$P_i(u, b, neigh) = avgstars_u + \frac{\sum_i^{neigh} commonvalue_i * (rating(u,i) - avgstars_u)}{\sum_i^{neigh} commonsvalue_i}$$

# Predizione

## Calcolo del Lambda

Il calcolo del lambda è effettuato dinamicamente, per ogni predizione, sulla base di due parametri estrapolati dal neighborhood: countSameUsers e countSameBusiness, rispettivamente per il neighborhood di business e user.

$$\lambda = \frac{\textit{user-value}}{(\textit{user-value} + \textit{business-value})}$$

# Predizione

## Calcolo del Neighborhood - ItemBased

tabelle di supporto

Review del business\_id  
di cui cerchiamo il neighborhood

Business*	Review
-----------	--------

Join tra la tabella precedente e la  
tabella utenti

Precedente	Utenti
------------	--------

Calcolo i business votati dall'  
utente corrente

User*	Review
-------	--------

# Predizione

## Calcolo del Neighborhood - ItemBased

Andiamo a raggruppare per utenti, e contiamo quanti user ha ogni business.  
Questi rappresentano gli user che sono hanno votato anche il business corrente.  
Ordiniamo poi i business, in base a quanti utenti in comune hanno con il business corrente.

Precedente	Business
------------	----------

```
select allTogetherB.business_id, COUNT(allTogetherB.user_id) as countSameUsers, avg(stars) as  
averageStarsUsers  
from allTogetherB, itemsThatAreVotedByHim where allTogetherB.business_id =  
itemsThatAreVotedByHim.business_id  
and allTogetherB.business_id != 'hzHyNCc3WijMVOzPcCuRjg'  
group by allTogetherB.business_id order by countSameUsers desc;
```