Harvard Capstone Project

Social media marketing: unleash the rising influencers

May 11, 2017

TF: Patrick Ohiomoba

Client: Ryan Lee, Christian Junge

Members: Chin Hui Chew, Ernest Chiew, Xinyi Ma, Leonard Loo
Business Motivation: Influencer marketing

$8 bn market

18.2% CAGR

47% Of online consumers use ad-block

92% Trust influencers over brands

74% Rely on social networks to guide purchase
Business Motivation: Our goal

- Budget
- Influence
- Scam
- Our goal
Client: Tribe Dynamics

Focuses on influencer marketing on Instagram for cosmetic brands
Data and Deliverables

Data

@-tags
user a, user b, @-tags count

#-mentions
user, month, brand, #-mentions count, Earned Media Value (EMV)

Deliverables

Prediction
20-page report

Rising Star
2 APIs

839 unique users
666 unique brands
12 months of 2016
Baseline Model

Method:
- Build directed \textit{weighted} influence graph
- Run PageRank and find \textit{top k nodes with highest scores}

Results:
- Nodes chosen \textit{consistent} with highest EMV per post

Problems:
- Does not model \textit{real world influence} which is \textit{stochastic}
- Top k nodes are \textit{not necessarily best} to seed given budget constraint
Preprocessing - Brand Segmentation

**Goal:** To decrease computational time and reduce overfitting

**Method:** Group brands based on user behavior then select a brand from each group.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Optimal Number of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical Clustering</td>
<td>6</td>
</tr>
<tr>
<td>K-means Clustering</td>
<td>7</td>
</tr>
<tr>
<td>Gaussian Mixture Model</td>
<td>8</td>
</tr>
</tbody>
</table>

**Literature Review: Linear Threshold Model**

**Algorithm**

<table>
<thead>
<tr>
<th>Optimal Number of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

**Cluster Dendrogram of Brands**

**Literature Review: Gap Statistics**

Matching

Prediction API

Rising stars API
Matching: Probability Transition Matrix, $P$

$$\text{# post}_t \times \text{Probability matrix} = \text{# post}_{t+1}$$

Closed-form solution:

$$P = (B_t^T B_t)^{-1} B_t^T B_{t+1}$$
Matching: Linear Threshold Model (LTM)

Activation: $\sum w_{v,u_{\text{active}}} \geq U(0,1)$

Convergence: No new activations between iterations

Literature Review: Linear Threshold Model
Maximizing the spread of influence through a social network by D. Kempe, et. al
Matching: Linear Threshold Model (LTM)

Activation: \( \sum_{u \in U} w_{u, u_{active}} \geq U(0, 1) \)

Convergence: No new activations between iterations

Literature Review: Linear Threshold Model
Maximizing the spread of influence through a social network by D. Kempe, et. al
Matching: **Dynamic Threshold Model (DTM)**

Dynamic Activation: \( \sum w_{u,v}u_{active} \geq (U(0, 1) - \text{normalized column sum of } P) \)

Convergence: No new activations between iterations

---

**Literature Review: Linear Threshold Model**

*Maximizing the spread of influence through a social network by D. Kempe, et. al*
Matching: Fast DTM with matrix ops

for u in ... :  
   for v in ... :

\[ A \cdot P^T[u] \]

Open Source implementations  
Our implementation

<table>
<thead>
<tr>
<th>Time per Iteration in seconds</th>
<th>800 nodes</th>
<th>5000 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Loops</td>
<td>1.212</td>
<td>50.215</td>
</tr>
<tr>
<td>Matrix Ops</td>
<td>0.005</td>
<td>0.214</td>
</tr>
</tbody>
</table>

We are 250x faster!
## Matching: Setup

<table>
<thead>
<tr>
<th>Reach Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reach: Number of activated nodes in a month</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of iterations to run for DTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 iterations = ~9.8 signal-to-noise (Mean / SE)</td>
</tr>
</tbody>
</table>
Matching: DTM + P

Our model **matches** the observed trends:
Matching

Prediction API

Rising stars API
Prediction: Algorithm

Step 1: Predict using past \( P \)

\[ P_{t-2,t-1} = (B_{t-2}^T B_{t-2})^{-1} B_{t-2}^T B_{t-1} \]

Step 2: Scale predictions with factor learned from past ratios

\[ \sum_{i=1}^{t-1} \frac{\text{actual}_i}{\text{prediction}_i} \]

\[ \frac{t-1}{t-1} \]
Prediction: API

**Input**
- mention csv file, @-tag csv file, brand ID

**API**
- `python prediction.py mention.csv tag.csv brand_id`

**Output**
- Predicted reach
Rising Star: Quality Reach

PageRank Score
Calculated from @-tag counts among users

Quality Reach
Summed PageRank scores of activated nodes (reach) in a month
Rising Star: Single Rising Star

DTM over first 6 months for each user
Rising Star: Single Rising Star

DTM over **first 6 months** for **each user**

Model gradient with **Linear Regression**

Top, median and bottom influencers in terms of slope of Reach

Reach is similar
Rising Star: Single Rising Star

DTM over **first 6 months** for each user

Model gradient with **Linear Regression**

Reach is similar

Quality Reach is different
Rising Star: How to choose Rising-k

Can’t just choose top 5 rising stars

Optimal choice is $^{839}C_5 = 3.42 \times 10^{12}$
Computationally painful!
Rising Star: Rising-k with Greedy Hill Climb

Algorithm

Step 1: Choose the fastest rising influencer

Step 2: Choose the next influencer such that it is the best pair with the first

Step 3: Perform Step 2 until k influencers are chosen

Literature Review: Proven to approximate 63% of optimal choice

Maximizing the spread of influence through a social network by D. Kempe, et. al
Rising Star: Rising 5 vs Top 5

We chose 5 rising influencers using our algorithm. We chose top 5 all-time influencers based on EMV/post data.

Comparison:
Rising Star: Rising 5 vs Top 5

We chose 5 rising influencers using our algorithm. We chose top 5 all-time influencers based on EMV/post data.

Comparison:

Rising stars caught up in quality reach after 3 months!
Rising Star: API

Input
mention csv file, @-tag csv file, brand ID, number of rising stars

API
```
python rising_star.py mention.csv tag.csv brand_id num_rising_star
```

Output
IDs of rising stars: [216, 475, 484, 490, 728]
Matching

- Prediction API
- Rising stars API
Prediction API: So what?

Output

Plot of predicted reach versus actual reach

Business Value

- Marketing impact assessment
- Sales forecast
- Predict even with limited past data
Rising Star API: So what?

Output

IDs of the k rising stars identified [216, 475, 484, 490, 728]

Business Value

- Cost savings
- Competitive advantage
- Capture exclusive contracts with rising influencers early
List of a few of the stuff attempted

1. Finding P by gradient descent
   
   **Good:** Faster computation time  
   **Bad:** Not as accurate matching

   \[
   \mathcal{L} = (B_{t+1} - B'_{t+1})^\top (B_{t+1} - B'_{t+1}) \\
   = (B_{t+1} - B_t P)^\top (B_{t+1} - B_t P) \\
   \frac{d\mathcal{L}}{dP} = -2(B_{t+1} - B_t P)B_t^\top \\
   P' \leftarrow P - \eta \times \frac{d\mathcal{L}}{dP}
   \]

2. Clustering to find k rising influencers
   
   No meaningful clusters

<table>
<thead>
<tr>
<th>Clustering algorithm</th>
<th>Counts per cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Nearest Neighbor with k=3</td>
<td>787, 45, 7</td>
</tr>
<tr>
<td>GMM with 3 components</td>
<td>751, 71, 17</td>
</tr>
<tr>
<td>Affinity Propagation</td>
<td>839</td>
</tr>
</tbody>
</table>
Thank You!

Any Questions?