Real-Time Top-R Topic Detection on Twitter with Topic Hijack Filtering

Kohei Hayashi (NII & ERATO)

August 3, 2015

Joint work with
- Takanori Maehara (Shizuoka Univ)
- Masashi Toyoda (Univ Tokyo)
- Ken-ichi Kawarabayashi (NII & ERATO)
SNS with short messages (tweets)

Big data 41M users, 1.4B interactions

Diversity Covering any topics: news, politics, TV, ...

Rapid 1 tweet \( \leq \) 140 chars

\( \Rightarrow \) Low latency
Automatic Trend Detection on Twitter

• Find word clusters by word co-occurrence
• May discover breaking news and events even faster than news media
Automatic Trend Detection on Twitter

A promising data resource for topic detection

- Find word clusters by word co-occurrence
- May discover breaking news and events even faster than news media
Two Challenges

- Topic Detection in Real-time
- Noise Filtering
An ultimate goal of topic detection on Twitter

- Have to deal with 0.27M tweets/min
- Words rarely co-occur
  ⇒ Severely degrade the quality of topics
Noise Filtering

Many spam tweets generated by not human

- e.g. “tweet buttons”

Exaggerate co-occurrence and “hijack” important topics
Contributions

A streaming topic detection algorithm based on non-negative matrix factorization (NMF)

1. Highly scalable:
   Able to deal with a $20M \times 1M$ sparse matrix/sec
2. Automatic topic hijacking detection & elimination
Contributions

A streaming topic detection algorithm based on non-negative matrix factorization (NMF)

1. Highly scalable:
   Able to deal with a $20M \times 1M$ sparse matrix/sec

2. Automatic topic hijacking detection & elimination

Technical Points

1. Reformulate NMF in a stochastic manner
   - Stochastic gradient descent (SGD) updates with $O(\text{NNZ})$ time

2. Use of statistical testing
   - Assume normal topics follow power law
Streaming NMF
Consider to obtain $R$ topics from all past tweets
Consider to obtain $R$ topics from all past tweets
Consider to obtain $R$ topics from all past tweets

\[ X \approx \sum_{r=1}^{R} u_r v_r^\top \]
Problem

\[
\min_{U \in \mathbb{R}^{I \times R}_+, V \in \mathbb{R}^{J \times R}_+} f_\lambda(X; U, V),
\]

\[
f_\lambda(X; U, V) = \frac{1}{2} \| X - UV^\top \|_F^2 + \frac{\lambda}{2} (\|U\|_F^2 + \|V\|_F^2)
\]

Batch Algorithm

Repeat until convergence:

1. \( U \leftarrow [U - \eta \nabla_U f_\lambda(X; U, V)]_+ \)
2. \( V \leftarrow [V - \eta \nabla_V f_\lambda(X; U, V)]_+ \)

Guaranteed converging to stationary points
Stochastic Formulation

- Now we observe $X^{(t)}$ for each time $t$
- Keep track to $\bar{X}^{(t)} = \frac{1}{t} \sum_{s=1}^{t} X^{(s)}$
  - Efficient updates from $\bar{X}^{(t)}$ to $\bar{X}^{(t+1)}$?
Stochastic Formulation

• Now we observe $X(t)$ for each time $t$
• Keep track to $\bar{X}(t) = \frac{1}{t} \sum_{s=1}^{t} X(s)$
  • Efficient updates from $\bar{X}(t)$ to $\bar{X}(t+1)$?

Key idea: decompose $f_\lambda$ for each $t$

\[
\| \bar{X}(t) - UV^\top \|_F^2 = \frac{1}{t} \sum_s \| X(s) - UV^\top \|_F^2 + \text{const.}
\]
Stochastic Formulation

- Now we observe $X^{(t)}$ for each time $t$
- Keep track to $\bar{X}^{(t)} = \frac{1}{t} \sum_{s=1}^{t} X^{(s)}$
  - Efficient updates from $\bar{X}^{(t)}$ to $\bar{X}^{(t+1)}$?

Key idea: decompose $f_\lambda$ for each $t$

\[
\|\bar{X}^{(t)} - UV^\top\|_F^2 = \frac{1}{t} \sum_s \|X^{(s)} - UV^\top\|_F^2 + \text{const.}
\]

If $X^{(t)}$ is i.i.d. random variable,

\[
\|E[X^{(t)}] - UV^\top\|_F^2 = E\|X^{(t)} - UV^\top\|_F^2 + \text{var}[X^{(t)}]
\]

Now we can use stochastic optimization
Streaming Algorithm

For $t = 1, \ldots, T$:

1. $A_U \leftarrow \nabla_U \nabla_U f_\lambda, \quad A_V \leftarrow \nabla_V \nabla_V f_\lambda$ (metrics)
2. $U^{(t)} \leftarrow [U^{(t-1)} - \eta_t \nabla_U f_\lambda(X^{(t)}; U, V^{(t-1)})A_U^{-1}]^+$
3. $V^{(t)} \leftarrow [V^{(t-1)} - \eta_t \nabla_V f_\lambda(X^{(t)}; U^{(t)}, V)A_V^{-1}]^+$

Guaranteed converging to the stationary points of $f_\lambda(\bar{X}^{(t)}; U, V)$ for some $\eta_n$ and i.i.d. $\{X^{(t)}\}$
Comparing with the Batch NMF ...

Much faster

- Update cost: depends on $\text{NNZ}(X^{(t)}) \ll \text{NNZ}(\bar{X}^{(t)})$

Memory efficient

- Able to discard $X^{(1)}, \ldots, X^{(t-1)}$
Comparing with the Batch NMF ...

Much faster

- Update cost: depends on $\text{NNZ}(\mathbf{X}^{(t)}) \ll \text{NNZ}((\mathbf{X})^{(t)})$

Memory efficient

- Able to discard $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}$

Smoothing effect

$$
\mathbf{U}^{(t)} = [(1 - \eta_t)\mathbf{U}^{(t-1)} + \eta_t\mathbf{X}^{(t)}\mathbf{V}^{(t-1)}\mathbf{A}_U^{-1}] + \\
= [(1 - \eta_t)\mathbf{U}^{(t-1)} + \eta_t \arg\min_{\mathbf{U}} f_\lambda(\mathbf{X}^{(t)}; \mathbf{U}, \mathbf{V}^{(t-1)})] +
$$

- A weighted average of
  the prev solution and the NMF solution of $\mathbf{X}^{(t)}$

- Mitigates the sparsity of $\mathbf{X}^{(t)}$
Topic Hijacking Detection
Problem Setting

**Goal:** Find hijacked topics

**Idea:** Check word distributions

- The word dist of a *hijacked topic* should be different from the word dist of a *normal topic*
Problem Setting

**Goal:** Find hijacked topics

**Idea:** Check word distributions

- The word dist of a *hijacked topic* should be different from the word dist of a *normal topic*
- NMF estimates topic-specific word dists as $V$

\[
X_{ij} \propto p(\text{user}_i, \text{word}_j) \\
\propto \sum_r p(\text{topic}_r)p(\text{user}_i|\text{topic}_r)p(\text{word}_j|\text{topic}_r) \\
\propto \sum_r u_{ir}v_{jr}
\]
Defining Normal/Hijacked Topics

Normal topics:

- Many users involve
  - Mixing many different vocabularies
  - Heavy-tailed (Zipf’s law)
Defining Normal/Hijacked Topics

Normal topics:
- Many users involve
  ⇒ Mixing many different vocabularies
  ⇒ Heavy-tailed (Zipf’s law)

Definition: A Normal Topic

Topic \( r \) is normal if

\[
p(\text{rank(word)} \mid \text{topic}_r) = \text{power} (\alpha)
\]
Hijacked topics:

- Few users involve
  - The same vocabulary is repeatedly used
  - Uniform probs & (almost) no tail
Defining Normal/Hijacked Topics (Cont’d)

Hijacked topics:

- Few users involve
  - The same vocabulary is repeatedly used
  - Uniform probs & (almost) no tail

Definition: A Hijacked Topic

Topic \( r \) is hijacked if

\[
p(\text{rank(word)} \mid \text{topic}_r) = \text{step}(\text{rank}_{\text{min}})
\]
Log-likelihood Ratio Test

\[ \mathcal{L}(\text{rank}_{\text{min}}) = \sum_j \log \frac{\text{step}(\text{rank}_j \mid \text{rank}_{\text{min}})}{\text{power}(\text{rank}_j \mid \hat{\alpha})} \]

**Theorem (Asymptotic normality [Vuong'89])**

Let \( N \) be \# of observed words. Then, \( \mathcal{L}(\text{rank}_{\text{min}})/\sqrt{N} \) converges in distribution to \( N(0, \sigma^2) \) where

\[ \sigma^2 = \frac{1}{N} \sum_{j=1}^{N} \left( \log \frac{\text{step}(\text{rank}_j \mid \text{rank}_{\text{min}})}{\text{power}(\text{rank}_j \mid \hat{\alpha})} \right)^2 - \left( \frac{1}{N} \mathcal{L}(\text{rank}_{\text{min}}) \right)^2. \]
Log-likelihood Ratio Test

\[ \mathcal{L}(\text{rank}_{\text{min}}) = \sum_j \log \frac{\text{step}(\text{rank}_j \mid \text{rank}_{\text{min}})}{\text{power}(\text{rank}_j \mid \hat{\alpha})} \]

**Theorem (Asymptotic normality [Vuong'89])**

Let \( N \) be \# of observed words. Then, \( \mathcal{L}(\text{rank}_{\text{min}})/\sqrt{N} \) converges in distribution to \( N(0, \sigma^2) \) where

\[ \sigma^2 = \frac{1}{N} \sum_{j=1}^{N} \left( \log \frac{\text{step}(\text{rank}_j \mid \text{rank}_{\text{min}})}{\text{power}(\text{rank}_j \mid \hat{\alpha})} \right)^2 - \left( \frac{1}{N} \mathcal{L}(\text{rank}_{\text{min}}) \right)^2. \]

For \( r = 1, \ldots, R \):

- Estimate \( \hat{\alpha} = \text{argmax}_\alpha \log \text{power}(\text{rank}(u_k) \mid \alpha) \)
- For \( \text{rank}_{\text{min}} = 1, \ldots, 140 \):
  - Compute \( \mathcal{L}(\text{rank}_{\text{min}}) \)
  - Topic \( r \) is hijacked if \( p\text{-val} < 0.05 \)
Experiments
Data

Japanese Twitter stream
- April 15–16, 2013
  - 417K users
  - 1.98M words
  - 15.3M tweets
  - 69.4M co-occurrences
- Generated $X^{(t)}$ for each 10K co-occurrences
## Runtime

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th>1%</th>
<th>10%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Batch</strong></td>
<td></td>
<td>27.0m</td>
<td>1.9h</td>
<td>16.4h</td>
</tr>
<tr>
<td><strong>Online</strong></td>
<td>[Cao+ 07]</td>
<td>5.6h</td>
<td>9.2h</td>
<td>17.8h</td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td>[Saha+ 12]</td>
<td>16.7h</td>
<td>&gt;24h</td>
<td>&gt;24h</td>
</tr>
<tr>
<td><strong>Streaming</strong></td>
<td>[proposed]</td>
<td>4.0m</td>
<td>21.7m</td>
<td>3.6h</td>
</tr>
<tr>
<td><strong>w/ Filter</strong></td>
<td>[proposed]</td>
<td>5.3m</td>
<td>24.1m</td>
<td>3.8h</td>
</tr>
</tbody>
</table>
Runtime

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th>1%</th>
<th>10%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>27.0m</td>
<td>1.9h</td>
<td>16.4h</td>
<td></td>
</tr>
<tr>
<td>Online [Cao+ 07]</td>
<td>5.6h</td>
<td>9.2h</td>
<td>17.8h</td>
<td></td>
</tr>
<tr>
<td>Dynamic [Saha+ 12]</td>
<td>16.7h</td>
<td>&gt;24h</td>
<td>&gt;24h</td>
<td></td>
</tr>
<tr>
<td>Streaming</td>
<td>4.0m</td>
<td>21.7m</td>
<td>3.6h</td>
<td></td>
</tr>
<tr>
<td>w/ Filter [proposed]</td>
<td>5.3m</td>
<td>24.1m</td>
<td>3.8h</td>
<td></td>
</tr>
</tbody>
</table>

Streaming NMF: ×5–250 faster!
- 67K tweets/m
  ⇒ The real-time speed of all jp tweets!

Filtering cost is ignorable
## Perplexity

- Similarity between a topic dist and a target dist
  - Used Y! headlines\(^1\) for the target term dist
- Lower is better

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th>1%</th>
<th>10%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>9.01E+09</td>
<td>6.32E+06</td>
<td>4.51E+04</td>
<td></td>
</tr>
<tr>
<td>Online [Cao+ 07]</td>
<td>1.06E+04</td>
<td>3.25E+04</td>
<td>1.83E+04</td>
<td></td>
</tr>
<tr>
<td>Dynamic [Saha+ 12]</td>
<td>6.53E+04</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Streaming [proposed]</td>
<td>5.65E+07</td>
<td>3.25E+04</td>
<td>8.71E+03</td>
<td></td>
</tr>
<tr>
<td>w/ Filter [proposed]</td>
<td>5.25E+09</td>
<td>2.40E+04</td>
<td>7.90E+03</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)http://news.yahoo.co.jp/list
Perplexity

- Similarity between a topic dist and a target dist
  - Used Y! headlines\(^1\) for the target term dist
- Lower is better

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th>1%</th>
<th>10%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>9.01E+09</td>
<td>6.32E+06</td>
<td>4.51E+04</td>
<td></td>
</tr>
<tr>
<td>Online [Cao+ 07]</td>
<td>1.06E+04</td>
<td>3.25E+04</td>
<td>1.83E+04</td>
<td></td>
</tr>
<tr>
<td>Dynamic [Saha+ 12]</td>
<td>6.53E+04</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Streaming [proposed]</td>
<td>5.65E+07</td>
<td>3.25E+04</td>
<td>8.71E+03</td>
<td></td>
</tr>
<tr>
<td>w/ Filter [proposed]</td>
<td>5.25E+09</td>
<td>2.40E+04</td>
<td>7.90E+03</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)http://news.yahoo.co.jp/list

- Streaming NMF: best at 100% data
- Topic hijacking filter improves perplexity
ボストンマラソン爆破事件のトピックを確認
Detected Hijacking Phrases

- シネマトゥデイ 妖精トム 映画 ハリウッド 戦闘 クルーズ 主演
- 発売 2013 情報 開催 参加 商品 入荷
- 限定 情報 開催 イベント 好評 商品 @******
- リプライ 富士山 樋口 フェスティバル 興味 早稲田 開催 現在 不問 募集 スタッフ 大

- リフォロー アカウント 19836 フォロワー Only 希望 支援 696382 交流
- 無料 日本 拡散 希望 2013/04 リプ

- 特徴 燃料 格闘 操縦 射撃 装甲 整備 評価 機動
- 所有 City ポイント 前回 Tweet アクセス 獲得 Intel
- 自動 だれ 宣伝 AutoTweet オートツイート Twitter 定期 設定 サイト

- とノω。
- う▂▂▂▂▂▂▂▂▂ˈ お ●∩ く⊂ よ。）「」。
- *・*・*・*・*・*・*・*・*・*・*'。

広告系
拡散系
ボット系
顔文字系
Summary

Proposed the streaming algorithm for Twitter topic detection

- Works in real time
  (would handle all jp tweets in theory)
- Automatically filters spam topics
Summary

Proposed the streaming algorithm for Twitter topic detection

- Works in real time
  (would handle all jp tweets in theory)
- Automatically filters spam topics

Thank you!
Integrated Twitter Topic Detection System

For $t = 1, \ldots, T$:

1. Generate $X^{(t)}$ from tweets $\not\in$ Blacklist \hspace{2cm} $O(N_t)$
2. Update $U$, $V$ by SGD \hspace{2cm} $O(N_t R^2 + R^3)$
3. With some intervals,
   - Detect Topic Hijacking and update Blacklist \hspace{2cm} $O(J)$