TwiSent: A Multi-Stage System for Analyzing Sentiment in Twitter

Subhabrata Mukherjee, Akshat Malu, Balamurali A.R. and Pushpak Bhattacharyya
Dept. of Computer Science and Engineering, IIT Bombay

21st ACM Conference on Information and Knowledge Management
CIKM 2012,
Hawai, Oct 29 - Nov 2, 2012
Had Hella fun today with the team. Y’all are hilarious! &Yes, i do need more black homies......
Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily

Had Hella fun today with the team. Y’all are hilarious! & Yes, i do need more black homies......
Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily.
- This information content could be leveraged to create applications that have a social as well as an economic value.

Had Hella fun today with the team. Y’all are hilarious! & Yes, i do need more black homies......
Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily
- This information content could be leveraged to create applications that have a social as well as an economic value
- Text limit of 140 characters per tweet makes Twitter a noisy medium
  - Tweets have a poor syntactic and semantic structure
  - Problems like slangs, ellipses, nonstandard vocabulary etc.

Had Hella fun today with the team. Y’all are hilarious! &Yes, i do need more black homies......
Social Media Analysis

- Social media sites, like Twitter, generate around 250 million tweets daily

- This information content could be leveraged to create applications that have a social as well as an economic value

- Text limit of 140 characters per tweet makes Twitter a noisy medium
  - Tweets have a poor syntactic and semantic structure
  - Problems like slangs, ellipses, nonstandard vocabulary etc.

- Problem is compounded by increasing number of spams in Twitter
  - Promotional tweets, bot-generated tweets, random links to websites etc.
  - In fact Twitter contains around 40% tweets as pointless babble

*Had Hella fun today with the team. Y’all are hilarious! &Yes, i do need more black homies......*
TwiSent: Multi-Stage System Architecture

Tweets
Tweet Fetcher
Spam Filter
Spell Checker
Dependency Extractor
Polarity Detector
Pragmatics Handler
Opinion
Spam Categorization and Features

- Re-tweets
- Promotional tweets for some entity
- Tweets containing links to some other websites
- Tweets in languages other than English
- Tweets with incomplete text
- Automatically generated tweets by bots
- Tweets built primarily for search engines or tweets with excessive off-topic keywords
- Multiple tweets offering substantially the same content

<table>
<thead>
<tr>
<th>Number of Words per Tweet</th>
<th>Frequency of Foreign Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Word Length</td>
<td>Validity of First Word</td>
</tr>
<tr>
<td>Frequency of “?” and “!”</td>
<td>Presence / Absence of links</td>
</tr>
<tr>
<td>Frequency of Numeral Characters</td>
<td>Frequency of POS Tags</td>
</tr>
<tr>
<td>Frequency of hashtags</td>
<td>Strength of Character Elongation</td>
</tr>
<tr>
<td>Frequency of @users</td>
<td>Frequency of Slang Words</td>
</tr>
<tr>
<td>Extent of Capitalization</td>
<td>Average Positive and Negative Sentiment of Tweets</td>
</tr>
<tr>
<td>Frequency of the First POS Tag</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm for Spam Filter

Input: Build an initial naive bayes classifier NB-C, using the tweet sets $M$ (mixed unlabeled set containing spams and non-spams) and $P$ (labeled non-spam set)

1: Loop while classifier parameters change
2:   for each tweet $t_i \in M$ do
3:     Compute $\Pr[c_1 | t_i]$, $\Pr[c_2 | t_i]$ using the current NB // $c_1$ - non-spam class, $c_2$ - spam class
4:     $\Pr[c_2 | t_i] = 1 - \Pr[c_1 | t_i]$
5:     Update $\Pr[f_{i,k}|c_1]$ and $\Pr[c_1]$ given the
6:         probabilistically assigned class for all $t_i$ ($\Pr[c_1|t_i]$).
7:     (a new NB-C is being built in the process)
8:   end for
9: end loop

$$\Pr[c_j | t_i] = \frac{\Pr[c_j] \prod_k \Pr[f_{i,k}|c_j]}{\sum_r \Pr[c_r] \prod_k P(f_{i,k}|c_r)}$$
Categorization of Noisy Text

- Dropping of Vowels - *btfl* (*beautiful*), *lvng* (*loving*)
- Vowel Exchange - *good* vs. *gud* (*o,u*)
- Mis-spelt words - *redicule* (*ridicule*), *magnificant* (*magnificent*)
- Text Compression - *shok* (*shock*), *terorism* (*terrorism*)
- Phonetic Transformation - *be8r* (*better*), *gud* (*good*), *fy9* (*fine*), *gr8* (*great*)
- Normalization and Pragmatics - *hapyyyyy* (*happy*), *guuuuud* (*good*)
- Segmentation with Punctuation - *beautiful, (beautiful)*
- Segmentation with Compound Words - *breathtaking* (*breath-taking*), *eyecatching* (*eye-catching*), *good-looking* (*good looking*)
- Hashtags and Segmentation - #notevenkidding, #worthawatch
- Combination of all - #awsummm (*awesome*), gr88888 (*great*), amzng,btfl (*amazing, beautiful*)
Spell-Checker Algorithm
Spell-Checker Algorithm

- Heuristically driven to resolve the *identified errors* with a *minimum edit distance based spell checker*
Spell-Checker Algorithm

- Heuristically driven to resolve the identified errors with a minimum edit distance based spell checker

- A normalize function takes care of Pragmatics and Number Homophones
  - Replaces happyyyyy with hapy, ‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’
Spell-Checker Algorithm

- Heuristically driven to resolve the identified errors with a minimum edit distance based spell checker

- A normalize function takes care of Pragmatics and Number Homophones
  - Replaces happpyyyyy with hapy, ‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’

- A vowel_dropped function takes care of the vowel dropping phenomenon
Spell-Checker Algorithm

- Heuristically driven to resolve the identified errors with a minimum edit distance based spell checker

- A *normalize* function takes care of Pragmatics and Number Homophones
  - Replaces *happyyyy* with *hapy*, ‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’

- A *vowel_dropped* function takes care of the vowel dropping phenomenon

- The parameters *offset* and *adv* are determined empirically
Spell-Checker Algorithm

- Heuristically driven to resolve the identified errors with a minimum edit distance based spell checker

- A normalize function takes care of Pragmatics and Number Homophones
  - Replaces happpyyyy with hapy, ‘2’ with ‘to’, ‘8’ with ‘eat’, ‘9’ with ‘ine’

- A vowel_dropped function takes care of the vowel dropping phenomenon

- The parameters offset and adv are determined empirically

- Words are marked during normalization, to preserve their pragmatics
  happppyyyyy, normalized to hapy and thereafter spell-corrected to happy, is marked so as to not lose its pragmatic content
Spell-Checker Algorithm

- **Input:** For string s, let S be the set of words in the lexicon starting with the initial letter of s.

/* Module Spell Checker */

for each word \( w \in S \) do

\[ w' = \text{vowel\_dropped}(w) \]

\[ s' = \text{normalize}(s) \]

/*\text{diff}(s, w)\ gives\ difference\ of\ length\ between\ s\ and\ w*/

if \( \text{diff}(s', w') < \text{offset} \) then

\[ \text{score}[w] = \min(\text{edit\_distance}(s, w), \text{edit\_distance}(s, w'), \text{edit\_distance}(s', w)) \]

else

\[ \text{score}[w] = \max\_\text{centinel} \]

end if

end for
Sort score of each \( w \) in the Lexicon and retain the top \( m \) entries in suggestions(s) for the original string \( s \)

\[
\text{for each } t \text{ in suggestions(s) do}
\]

\[
\text{edit}_1 = \text{edit} \text{distance}(s', s)
\]

/* \( t.\text{replace}(char1, char2) \) replaces all occurrences of \( char1 \) in the string \( t \) with \( char2 \)*/

\[
\text{edit}_2 = \text{edit} \text{distance}(t.\text{replace}(a, e), s')
\]

\[
\text{edit}_3 = \text{edit} \text{distance}(t.\text{replace}(e, a), s')
\]

\[
\text{edit}_4 = \text{edit} \text{distance}(t.\text{replace}(o, u), s')
\]

\[
\text{edit}_5 = \text{edit} \text{distance}(t.\text{replace}(u, o), s')
\]

\[
\text{edit}_6 = \text{edit} \text{distance}(t.\text{replace}(i, e), s')
\]

\[
\text{edit}_7 = \text{edit} \text{distance}(t.\text{replace}(e, i), s')
\]

\[
\text{count} = \text{overlapping} \text{characters}(t, s')
\]

\[
\text{min} \_\text{edit} =
\]

\[
\text{min} (\text{edit}_1, \text{edit}_2, \text{edit}_3, \text{edit}_4, \text{edit}_5, \text{edit}_6, \text{edit}_7)
\]

\[
\text{if } (\text{min} \_\text{edit} == 0 \text{ or score}[s] == 0) \text{ then}
\]

\[
\text{adv} = -2 /* \text{ for exact match assign advantage score */}
\]

\[
\text{else}
\]

\[
\text{adv} = 0
\]

\[
\text{end if}
\]

\[
\text{final} \_\text{score}[t] = \text{min} \_\text{edit} + \text{adv} + \text{score}[w] - \text{count};
\]

\[
\text{end for}
\]

\[
\text{return } t \text{ with minimum final} \_\text{score};
\]
I have an ipod and it is a great buy but I'm probably the only person that dislikes the iTunes software.

Here the sentiment w.r.t ipod is positive whereas that respect to software is negative.
Opinion Extraction Hypothesis

“More closely related words come together to express an opinion about a feature”
“I want to use Samsung which is a great product but am not so sure about using Nokia”.

Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

Here “great” and “product” are more related to Samsung than they are to Nokia

Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
“I want to use Samsung which is a great product but am not so sure about using Nokia”.

Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

Here “great” and “product” are more related to Samsung than they are to Nokia.

Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”.
“I want to use Samsung which is a great product but am not so sure about using Nokia.”

- Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.
- *Here “great” and “product” are more related to Samsung than they are to Nokia*
- Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
Hypothesis Example

“I want to use Samsung which is a great product but am not so sure about using Nokia.”

Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

Here “great” and “product” are more related to Samsung than they are to Nokia.

Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
"I want to use Samsung which is a great product but am not so sure about using Nokia."

Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

Here “great” and “product” are more related to Samsung than they are to Nokia

Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
Hypothesis Example

“I want to use Samsung which is a great product but am not so sure about using Nokia”.

- Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

- Here “great” and “product” are more related to Samsung than they are to Nokia

- Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
“I want to use Samsung which is a great product but am not so sure about using Nokia.”

Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.

**Here “great” and “product” are more related to Samsung than they are to Nokia**

**Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”**
Hypothesis Example

“I want to use Samsung which is a great product but am not so sure about using Nokia.”

- Here “great” and “product” are related by an adjective modifier relation, “product” and “Samsung” are related by a relative clause modifier relation. Thus “great” and “Samsung” are transitively related.
- Here “great” and “product” are more related to Samsung than they are to Nokia
- Hence “great” and “product” come together to express an opinion about the entity “Samsung” than about the entity “Nokia”
Example of a Review

- I have an ipod and it is a great buy but I'm probably the only person that dislikes the iTunes software.
Example of a Review

- I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the **iTunes software**.
I have an ipod and it is a great buy but I'm probably the only person that dislikes the iTunes software.
Example of a Review

- *I have an ipod* and it is a great buy but I'm probably the only person that dislikes the *iTunes software*. 
Example of a Review

I have an **ipod** and it is a great buy but I'm probably the only person that dislikes the **iTunes** **software**.
Feature Extraction : Domain Info
Not Available
Initially, all the Nouns are treated as features and added to the feature list $F$. 
Initially, all the Nouns are treated as features and added to the feature list $F$.

$F = \{ \text{ipod, buy, person, software} \}$
Initially, all the Nouns are treated as features and added to the feature list $F$.

$F = \{ \text{ipod, buy, person, software} \}$

Pruning the feature set
- Merge 2 features if they are strongly related
Initially, all the Nouns are treated as features and added to the feature list $F$.

$F = \{ \text{ipod}, \text{buy}, \text{person}, \text{software} \}$

Pruning the feature set
- Merge 2 features if they are strongly related

“buy” merged with “ipod”, when target feature = “ipod”,
- “person, software” will be ignored.
Initially, all the Nouns are treated as features and added to the *feature list* $F$.

$F = \{ \text{ipod, buy, person, software} \}$

- Pruning the feature set
  - Merge 2 features if they are *strongly related*

- “buy” merged with “ipod”, when target feature = “ipod”,
  - “person, software” will be ignored.

- “person” merged with “software”, when target feature = “software”
  - “ipod, buy” will be ignored.
Relations

- Direct Neighbor Relation
  - Capture **short range dependencies**
  - Any 2 consecutive words (such that none of them is a StopWord) are directly related
  - Consider a sentence $S$ and 2 consecutive words $w_i, w_{i+1}$.
  - If $w_i, w_{i+1} \notin \text{Stopwords}$, then they are directly related.

- Dependency Relation
  - Capture **long range dependencies**
  - Let $\text{Dependency\_Relation}$ be the list of significant relations.
  - Any 2 words $w_i$ and $w_j$ in $S$ are directly related, if
    $\exists D_i \text{ s.t. } D_i(w_i, w_j) \in \text{Dependency\_Relation}$
Given a sentence $S$, let $W$ be the set of all words in the sentence $S$. A Graph $G(W, E)$ is constructed such that any $w_i, w_j \in W$ are directly connected by $e_k \in E$, if $\exists R_l$ s.t. $R_l(w_i, w_j) \in R$. 
Algorithm

i. Initialize $n$ clusters $C_i \ \forall i = 1..n$

ii. Make each $f_i \in F$ the clusterhead of $C_i$. The target feature $f_t$ is the clusterhead of $C_t$. Initially, each cluster consists only of the clusterhead.
iii. Assign each word $w_j \in S$ to cluster $C_k$

$$s.t. \quad k = \arg \min_{i \in n} \text{dist}(w_j, f_i),$$

Where $\text{dist}(w_j, f_i)$ gives the number of edges, in the shortest path, connecting $w_j$ and $f_i$ in $G$. 
iv. Merge any cluster $C_i$ with $C_t$ if,
\[ \text{dist}(f_i, f_t) < \theta, \]
Where $\theta$ is some threshold distance.

v. Finally the set of words $w_i \in C_t$ gives the opinion expression regarding the target feature $f_t$.  

Clustering
Clustering
Clustering
Clustering
Clustering
Clustering
Clustering
Clustering
Pragmatics
Elongation of a word, repeating alphabets multiple times - Example: happppyyyyyy, gooooooood. More weightage is given by repeating them twice
Pragmatics

- **Elongation of a word, repeating alphabets multiple times** - Example: `happppyyyyyy, goooooood`. More weightage is given by repeating them twice.

- **Use of Hashtags** - `#overrated, #worthawatch`. More weightage is given by repeating them thrice.
Pragmatics

- *Elongation of a word, repeating alphabets multiple times* - Example: happppyyyyyy, goooooood. More weightage is given by repeating them twice

- *Use of Hashtags* - #overrated, #worthawatch. More weightage is given by repeating them thrice

- *Use of Emoticons* - 😊 (happy), ☹️ (sad)
Elongation of a word, repeating alphabets multiple times - Example: happpyyyyyyy, gooooooood. More weightage is given by repeating them twice

Use of Hashtags - #overrated, #worthawatch. More weightage is given by repeating them thrice

Use of Emoticons - 😊 (happy), ☹️ (sad)

Use of Capitalization - where words are written in capital letters to express intensity of user sentiments
  - Full Caps - Example: I HATED that movie. More weightage is given by repeating them thrice
  - Partial Caps - Example: She is a Loving mom. More weightage is given by repeating them twice
## Spam Filter Evaluation

### 2-Class Classification

<table>
<thead>
<tr>
<th>Tweets</th>
<th>Total Tweets</th>
<th>Correctly Classified</th>
<th>Misclassified</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>7007</td>
<td>3815</td>
<td>3192</td>
<td>54.45</td>
<td>55.24</td>
</tr>
<tr>
<td>Only spam</td>
<td>1993</td>
<td>1838</td>
<td>155</td>
<td>92.22</td>
<td>92.22</td>
</tr>
<tr>
<td>Only non-spam</td>
<td>5014</td>
<td>2259</td>
<td>2755</td>
<td>45.05</td>
<td>-</td>
</tr>
</tbody>
</table>

### 4-Class Classification

<table>
<thead>
<tr>
<th>Tweets</th>
<th>Total Tweets</th>
<th>Correctly Classified</th>
<th>Misclassified</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>7007</td>
<td>5010</td>
<td>1997</td>
<td>71.50</td>
<td>54.29</td>
</tr>
<tr>
<td>Only spam</td>
<td>1993</td>
<td>1604</td>
<td>389</td>
<td>80.48</td>
<td>80.48</td>
</tr>
<tr>
<td>Only non-spam</td>
<td>5014</td>
<td>4227</td>
<td>787</td>
<td>84.30</td>
<td>-</td>
</tr>
</tbody>
</table>
TwiSent Evaluation
TwiSent Evaluation

Lexicon-based Classification
TwiSent Evaluation

Lexicon-based Classification

![Bar Chart]

- 2-Class Classification Dataset
- 3-Class Classification Dataset
- 2-Class Classification Dataset

1. C-Feel-It
2. TwiSent
TwiSent Evaluation

Lexicon-based Classification

Supervised Classification
TwiSent Evaluation

Lexicon-based Classification

Supervised Classification

<table>
<thead>
<tr>
<th>System</th>
<th>2-class Accuracy</th>
<th>Precision/Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Feel-It</td>
<td>50.8</td>
<td>53.16/72.96</td>
</tr>
<tr>
<td>TwiSent</td>
<td>68.19</td>
<td>64.92/69.37</td>
</tr>
</tbody>
</table>
TwiSent Evaluation

<table>
<thead>
<tr>
<th>System</th>
<th>2-class Accuracy</th>
<th>Precision/Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Feel-It</td>
<td>50.8</td>
<td>53.16/72.96</td>
</tr>
<tr>
<td>TwiSent</td>
<td>68.19</td>
<td>64.92/69.37</td>
</tr>
</tbody>
</table>
TwiSent Evaluation

Lexicon-based Classification

Ablation Test

<table>
<thead>
<tr>
<th>System</th>
<th>2-class Accuracy</th>
<th>Precision/Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Feel-It</td>
<td>50.8</td>
<td>53.16/72.96</td>
</tr>
<tr>
<td>TwiSent</td>
<td>68.19</td>
<td>64.92/69.37</td>
</tr>
</tbody>
</table>
TwiSent Evaluation

Lexicon-based Classification

Ablation Test

<table>
<thead>
<tr>
<th>Module Removed</th>
<th>Accuracy</th>
<th>Statistical Significance Confidence (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity-Specificity</td>
<td>65.14</td>
<td>95</td>
</tr>
<tr>
<td>Spell-Checker</td>
<td>64.2</td>
<td>99</td>
</tr>
<tr>
<td>Pragmatics Handler</td>
<td>63.51</td>
<td>99</td>
</tr>
<tr>
<td>Complete System</td>
<td>66.69</td>
<td>-</td>
</tr>
</tbody>
</table>