Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning

Jing Ma¹, Wei Gao²*, Kam-Fai Wong¹,³

¹The Chinese University of Hong Kong
²Victoria University of Wellington, New Zealand
³MoE Key Laboratory of High Confidence Software Technologies, China

July 31- August 2, 2017 – ACL 2017 @ Vancouver, Canada

*Work done when Wei Gao was in QCRI
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Introduction

A story or statement whose truth value is **unverified** or deliberately **false**

Eric Tucker, a 35-year-old co-founder of a marketing company in Austin, Tex., had just about 40 Twitter followers. But his recent tweet about paid protesters being bused to demonstrations against President-elect Donald J. Trump fueled a nationwide conspiracy theory — one that Mr. Trump joined in promoting.

Mr. Tucker’s post was shared at least 16,000 times on Twitter and more than 350,000 times on Facebook. The problem is that Mr. Tucker got it wrong. There were no such buses packed with paid protesters.

But that didn’t matter.

While some fake news is produced purposefully by **teenagers in the Balkans** or **entrepreneurs in the United States** seeking to make money from advertising, false information can also arise from misinformed social media posts by regular people that are seized on and spread through a hyperpartisan blogosphere.

Here, The New York Times deconstructs how Mr. Tucker’s now-deleted declaration on Twitter the night after the election turned into a fake-news phenomenon. It is an example of how, in an ever-connected world where speed often takes precedence over truth, an observation by a private citizen can quickly become a talking point, even as it is being proved false.
The fake news went viral

** indicates the level of influence.

Start from a grass-roots users, promoted by some influential accounts, widely spread.
**Motivation**

- We generally are not good at distinguishing rumors

- It is crucial to track and debunk rumors early to minimize their harmful effects.

- Online fact-checking services have limited topical coverage and long delay.

- Existing models use feature engineering – over simplistic; or recently deep neural networks – ignore propagation structures.
Contributions

- Represent information spread on Twitter with propagation tree, formed by harvesting user’s interactions, to capture high-order propagation patterns of rumors.

- Propose a kernel-based data-driven method to generate relevant features automatically for estimating the similarity between two propagation trees.

- Enhance the proposed model by considering propagation paths from source tweet to subtrees to capture the context of transmission.

- Release two real-world twitter datasets with finer-grained ground truth labels.
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Related Work

- Systems based on common sense and investigative journalism, e.g.,
  - snopes.com
  - factcheck.org

- Learning-based models for rumor detection
  - Information credibility: Castillo et al. (2011), Yang et al. (2012)
  - Using cue terms: Zhao et al. (2015)
  - Using recurrent neural networks: Ma et al. (2016)

- Kernel-based works
  - Tree kernel: syntactic parsing (Collins and Duffy, 2001)
  - Question-answering (Moschitti, 2006)
  - Semantic analysis (Moschitti, 2004)
  - Relation extraction (Zhang et al., 2008)
  - Machine translation (Sun et al., 2010)
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Problem Statement

- Given a set of microblog posts \( R = \{r\} \), model each source tweet as a tree structure \( T(r) = < V, E > \), where each node \( v = (u_v, c_v, t_v) \) provide the creator of the post, the text content and post time. And \( E \) is directed edges corresponding to response relation.

- Task 1 – finer-grained classification for each source post
  
  *false rumor, true rumor, non-rumor, unverified rumor*

- Task 2 – detect rumor as early as possible
**O**: Walmart donates $10,000 to support Darren Wilson and the ongoing racist police murders

**U1**: You don't honestly believe that, do you?

**U2**: I honestly do

**U3**: ...Sam Walton gave 300k to Obama's campaign? *THINK.*

**U4**: Sam Walton was dead before #Obama was born. He have wired campaign donation from heavens.

**U5**: where is the credible link?

**U6**: Need proof of this—can't find any...

**U7**: not sure...sorry I see a meme trending but no proof...perhaps if we had real journalists?

**U8**: I'm pretty good at research—I think this is not true—plenty of other reasons to boycott WalMart. ;)

**Viva La Revolución**

@70torinoman · 18 Oct 2014

Walmart donates $10,000 to support Darren Wilson and the ongoing racist police murders 

#Ferguson #BoycottWalmart
Content-based signals (e.g., stance) (Zhao et al., 2015)

Network-based signals (e.g., relative influence) and temporal traits (Kwon et al., 2017)

Our hypothesis: high-order patterns needs to/could be captured using kernel method
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Traditional Tree Kernel (TK)

- TK computes the syntactic similarity between two sentences by counting the common subtrees.

- Kernel Function: $\sum_{v_i \in V_1} \sum_{v_j \in V_2} \Delta(v_i, v_j)$
  - $\Delta(v_i, v_j)$: common subtrees rooted at $v_i$ and $v_j$.
Why PTK?
- Existing tree kernel cannot apply here, since in our case (1) node is a vector of continuous numerical values; (2) similarity needs to be softly defined between two trees instead of hardly counting on identical nodes.

Similarity Definition
- User Similarity: \( \mathcal{E}(u_i, u_j) = \| u_i - u_j \| \)
- Content Similarity: \( J(c_i, c_j) = \frac{|Ngram(c_i) \cap Ngram(c_j)|}{|Ngram(c_i) \cup Ngram(c_j)|} \)
- Node Similarity:
  \[
  f(v_i, v_j) = e^{-t}(\alpha \mathcal{E}(u_i, u_j) + (1-\alpha)J(c_i, c_j))
  \]
Propagation Tree Kernel

- Given two trees $T_1 = \langle V_1, E_1 \rangle$ and $T_2 = \langle V_2, E_2 \rangle$, PTK computes similarity between them by enumerating all similar subtrees.

- Kernel Function: $\sum_{v_i \in V_1} \Delta(v_i, v'_i) + \sum_{v_j \in V_2} \Delta(v'_j, v_j)$
  - $v_i$ and $v'_i$ are similar node pairs from $V_1$ and $V_2$ respectively
    $$v'_i = \arg \max_{v_j \in V_2} f(v_i, v_j)$$
  - $\Delta(v, v')$: similarity of two subtrees rooted at $v$ and $v'$

- Kernel algorithm
  1) if $v$ or $v'$ are leaf nodes, then
    $$\Delta(v, v') = f(v, v');$$
  2) else
    $$\Delta(v, v') = f(v, v') \prod_{k=1}^{\min(nc(v), nc(v'))} (1 + \Delta(ch(v, k), ch(v', k)));$$
Context-Sensitive Extension of PTK

- Consider propagation paths from root node to the subtree

- Why cPTK?
  - PTK ignores the clues outside the subtrees and the route embed how the propagation happens.
  - Similar intuition to context-sensitive tree kernel (Zhou et al., 2007)

- Kernel Function: \[ \sum_{v_i \in V_1} \sum_{x=0}^{L_{v_i} - 1} \Delta_x(v_i, v_i') + \sum_{v_j \in V_2} \sum_{x=0}^{L_{v_j} - 1} \Delta_x(v_j', v_j) \]
  - \( L_{v_i} \): the length of propagation path from root \( r \) to \( v \).
  - \( v[x] \): the x-th ancestor of \( v \).
  - \( \Delta_x(v, v') \): similarity of subtrees rooted at \( v[x] \) and \( v'[x] \).

- Kernel Algorithm
  1) if \( v[x] \) and \( v'[x] \) are the x-th ancestor nodes of \( v \) and \( v' \), then
     \[ \Delta_x(v, v') = f(v[x], v'[x]); \]
  2) else
     \[ \Delta_x(v, v') = \Delta(v, v') \); i.e., PTK)
• Incorporate the proposed tree kernel functions (i.e., PTK or cPTK) into a supervised learning framework, for which we utilize a kernel-based SVM classifier.

• Avoid feature engineering – the kernel function can explore an implicit feature space when calculating the similarity between two objects.

• For multi-class task, perform One vs. all, i.e., building $K$ (# of classes) basic binary classifiers so as to separate one class from all the others.
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Data Collection

- Construct our propagation tree datasets based on two reference Twitter datasets:
  - Twitter15 (Liu et al, 2015)
  - Twitter16 (Ma et al, 2016)

Extract popular source tweets

Collect propagation threads

Convert event label: binary -> quarternary

revised labels

(Source tweet: highly retweeted or replied) (retweets: Twrench.com)
(replies: Web crawler)
## Statistics of Data Collection

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Twitter15</th>
<th>Twitter16</th>
</tr>
</thead>
<tbody>
<tr>
<td># of users</td>
<td>276,663</td>
<td>173,487</td>
</tr>
<tr>
<td># of source tweets</td>
<td>1,490</td>
<td>818</td>
</tr>
<tr>
<td># of threads</td>
<td>331,612</td>
<td>204,820</td>
</tr>
<tr>
<td># of non-rumors</td>
<td>374</td>
<td>205</td>
</tr>
<tr>
<td># of false rumors</td>
<td>370</td>
<td>205</td>
</tr>
<tr>
<td># of true rumors</td>
<td>372</td>
<td>205</td>
</tr>
<tr>
<td># of unverified rumors</td>
<td>374</td>
<td>203</td>
</tr>
<tr>
<td>Avg. time length / tree</td>
<td>1,337 Hours</td>
<td>848 Hours</td>
</tr>
<tr>
<td>Avg. # of posts / tree</td>
<td>223</td>
<td>251</td>
</tr>
<tr>
<td>Max # of posts / tree</td>
<td>1,768</td>
<td>2,765</td>
</tr>
<tr>
<td>Min # of posts / tree</td>
<td>55</td>
<td>81</td>
</tr>
</tbody>
</table>

URL of the datasets:
https://www.dropbox.com/s/0jhsfwep3ywvpca/rumdetect2017.zip?dl=0
Approaches to compare with

- **DTR**: Decision tree-based ranking model using enquiry phrases to identify trending rumors (Zhao et al., 2015)
- **DTC and SVM-RBF**: Twitter information credibility model using Decision Tree Classifier (Castillo et al., 2011); SVM-based model with RBF kernel (Yang et al., 2012)
- **RFC**: Random Forest Classifier using three parameters to fit the temporal tweets volume curve (Kwon et al., 2013)
- **SVM-TS**: Linear SVM classifier using time-series structures to model the variation of social context features. (Ma et al., 2015)
- **GRU**: The RNN-based rumor detection model. (Ma et al., 2016)
- **BOW**: linear SVM classifier using bag-of-words.
- **Ours (PTK and cPTK)**: Our kernel based model
- **PTK- and cPTK-**: Our kernel based model with subset node features.
### Results on Twitter

NR: Non-Rumor; FR: False Rumor; TR: True Rumor; UR: Unverified Rumor;

<table>
<thead>
<tr>
<th>Method</th>
<th>Accu.</th>
<th>NR F1</th>
<th>FR F1</th>
<th>TR F1</th>
<th>UR F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTR</td>
<td>0.409</td>
<td>0.501</td>
<td>0.311</td>
<td>0.364</td>
<td>0.473</td>
</tr>
<tr>
<td>SVM-RBF</td>
<td>0.318</td>
<td>0.455</td>
<td>0.037</td>
<td>0.218</td>
<td>0.225</td>
</tr>
<tr>
<td>DTC</td>
<td>0.454</td>
<td>0.733</td>
<td>0.355</td>
<td>0.317</td>
<td>0.415</td>
</tr>
<tr>
<td>SVM-TS</td>
<td>0.544</td>
<td>0.796</td>
<td>0.472</td>
<td>0.404</td>
<td>0.483</td>
</tr>
<tr>
<td>RFC</td>
<td>0.565</td>
<td>0.810</td>
<td>0.422</td>
<td>0.401</td>
<td>0.543</td>
</tr>
<tr>
<td>GRU</td>
<td>0.646</td>
<td>0.792</td>
<td>0.574</td>
<td>0.608</td>
<td>0.592</td>
</tr>
<tr>
<td>BOW</td>
<td>0.548</td>
<td>0.564</td>
<td>0.524</td>
<td>0.582</td>
<td>0.512</td>
</tr>
<tr>
<td>PTK-</td>
<td>0.657</td>
<td>0.734</td>
<td>0.624</td>
<td>0.673</td>
<td>0.612</td>
</tr>
<tr>
<td>cPTK-</td>
<td>0.697</td>
<td>0.760</td>
<td>0.645</td>
<td>0.696</td>
<td>0.689</td>
</tr>
<tr>
<td>PTK</td>
<td>0.710</td>
<td>0.825</td>
<td>0.685</td>
<td>0.688</td>
<td>0.647</td>
</tr>
<tr>
<td>cPTK</td>
<td>0.750</td>
<td>0.804</td>
<td>0.698</td>
<td>0.765</td>
<td>0.733</td>
</tr>
</tbody>
</table>
Results on Twitter16

NR: Non-Rumor; FR: False Rumor;
TR: True Rumor; UR: Unverified Rumor;

<table>
<thead>
<tr>
<th>Method</th>
<th>Accu.</th>
<th>NR</th>
<th>FR</th>
<th>TR</th>
<th>UR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
</tr>
<tr>
<td>DTR</td>
<td>0.414</td>
<td>0.394</td>
<td>0.273</td>
<td>0.630</td>
<td>0.344</td>
</tr>
<tr>
<td>SVM-RBF</td>
<td>0.321</td>
<td>0.423</td>
<td>0.085</td>
<td>0.419</td>
<td>0.037</td>
</tr>
<tr>
<td>DTC</td>
<td>0.465</td>
<td>0.643</td>
<td>0.393</td>
<td>0.419</td>
<td>0.403</td>
</tr>
<tr>
<td>SVM-TS</td>
<td>0.574</td>
<td>0.755</td>
<td>0.420</td>
<td>0.571</td>
<td>0.526</td>
</tr>
<tr>
<td>RFC</td>
<td>0.585</td>
<td>0.752</td>
<td>0.415</td>
<td>0.547</td>
<td>0.563</td>
</tr>
<tr>
<td>GRU</td>
<td>0.633</td>
<td>0.772</td>
<td>0.489</td>
<td>0.686</td>
<td>0.593</td>
</tr>
<tr>
<td>BOW</td>
<td>0.585</td>
<td>0.553</td>
<td>0.556</td>
<td>0.655</td>
<td>0.578</td>
</tr>
<tr>
<td>PTK-</td>
<td>0.653</td>
<td>0.673</td>
<td>0.640</td>
<td>0.722</td>
<td>0.567</td>
</tr>
<tr>
<td>cPTK-</td>
<td>0.702</td>
<td>0.711</td>
<td>0.664</td>
<td>0.816</td>
<td>0.608</td>
</tr>
<tr>
<td>PTK</td>
<td>0.722</td>
<td>0.784</td>
<td>0.690</td>
<td>0.786</td>
<td>0.644</td>
</tr>
<tr>
<td>cPTK</td>
<td>0.732</td>
<td>0.740</td>
<td>0.709</td>
<td>0.836</td>
<td>0.686</td>
</tr>
</tbody>
</table>
Results on Early Detection

- In the first few hours, the accuracy of the kernel-based methods climbs more rapidly and stabilize more quickly.
- cPTK can detect rumors with 72% accuracy for Twitter15 and 69.0% for Twitter16 within 12 hours, which is much earlier than the baselines and the mean official report times.
Example subtree of a rumor captured by the algorithm at early stage of propagation

[∗]: #Walmart donates $10,000 to #DarrenWilson fund to continue police racial profiling, brutality, murder of black ppl

[∗∗]: Judging by the way #Walmart pays & treats its employees this is no surprise.

[∗∗∗]: That's Wal-Mart always doing good for the overlords of society.

[∗]: RT

[∗]: lol

[∗∗∗]: RT

[∗]: RT

[∗]: RT

[∗∗]: NEED SOURCE, have a feeling this is just hearsay or they donated to backstoppers or something tangential.

[∗]: I agree. I have been hearing this all day but no source

[∗∗]: Exactly, i don't think Wal-Mart would let everyone know this if they did!!

[∗]: RT

Influential users boost its propagation, unpopular-to-popular information flow, Textual signals (underlined)
Outline

- Introduction
- Related Work
- Tweets Propagation
- Kernel Modeling
- Evaluation
- Conclusion and Future Work
Conclusion and future work

- Apply kernel learning method for rumor debunking by utilizing the propagation tree structures.
- Propagation tree encodes the spread of a source tweet with complex structured patterns and flat information regarding content, user and time associated with the tree nodes.
- Our kernel are combined under supervised framework for identifying rumors of finer-grained levels by directly measuring the similarity among propagation trees.

Future work:
- Explore network representation method to improve the rumor detection task.
- Develop unsupervised models due to massive unlabeled data from social media.
Thank You!