Twitter Spam Detection based on Deep Learning

Shirish Kayastha (6129700)
Background

- Authored by Tingmin Wu, Shigang Liu, Jun Zhang and Yang Xiang
  - School of Information Technology, Deakin University, Australia
  - January 2017
  - Cited by 55 future works since
Introduction

- Twitter
  - A social media platform for microblogging
  - Approx each month, 42 million new accounts

- Goal
  - Develop a spam filter based on deep learning
  - Evaluate with other existing techniques
    - ML based
    - URL blacklisting
Introduction

- Compared against Email spam, users are more likely to click on spam links on Twitter instead
  - Twitter: 0.13%
  - Email: 0.0003% to 0.0006%
- On a normal Twitter dataset of 2 million tweets
  - 8% of it is flagged as spam
Introduction

Previous techniques for spam filtering were

○ Tweet content-based classifier
  ■ Couldn't generate comparable results

○ Machine learning-based binary classifiers
  ■ Easily fabricated features, hampering the accuracy

○ Blacklisting filtering is time-consuming
Introduction

- Proposed solution
  - Word2Vec to pre-process tweets
  - Binary detection model to detect spam and non-spam
  - Solve existing problems in other models
    - low speed
    - under-standard accuracy
    - characteristic extraction problem
Literature Review

● Syntax Analysis
  ○ Analyze tweet on a word level platform
  ○ Detect shortened URLs
    ■ Used by spammers to hide malicious URLs
    ■ Older models cannot handle redirected URLs
  ○ Extract tweet content, use deep learning to learn syntactic contexts and information
  ○ Naive Bayes classifier gave more efficient scores
Literature Review

● Feature Analysis
  ○ Extract tweets using Twitter’s Streaming API
  ○ Analyze the tweet’s features, like “retweet” and “like” counts.
  ○ Hashtag extraction
  ○ Etc...
Literature Review

- **Bayesian Model**
  - Learning model to detect spammers

- **SVM Model**
  - Detect both spam and spammers

- **Random Forest Model**
  - Obtain features from spam profiles
  - Trained by Decorate and LogitBoost algorithms
Literature Review

● Issues with Feature Analysis
  ○ Spam drift issue
    ■ Can be solved by fuzzy-based redistribution and asymmetric sampling
  ○ Feature fabrication in data collection
    ■ Social graph to expose robust features
    ■ Understand Twitter profile behaviours
Literature Review

- Blacklist Techniques
  - Blocking malicious websites
  - Time-consuming
  - Manual labelling
Methodology

- Understand and analyse text using a deep neural network with multiple layers
- Word Vector for language analysis
- Text-base Vector for linguistic analysis
Methodology

- Apply Word2Vec to map each word into a multidimensional vector
- 2-level neural network using Huffman technique
- Hierarchical softmax
- Improves efficiency in training
  - High-frequency words can be processed fast
- Stochastic gradient descent by backpropagation
- Optimal vectors are extracted for each word by CBOW or Skip-gram
Methodology

- **Doc2Vec training**
  - Represent one vector for every tweet using Paragraph Vector modelling

- **Word2Vec training**
  - Tweet-length with combination of word vectors and unique document vector per record

- **Input features for Random Forest or Neural Network**
  - Get document representation
  - Form training dataset
  - Form test dataset
Word2Vec

- 2-layer neural net that processes text by "vectorizing" words
- Input is a text corpus and its output is a set of vectors
- Text into a numerical form that deep neural networks can understand.
- Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.
Word2Vec
Doc2Vec

- Doc2vec is an NLP tool
- Represent documents as a vector and is a generalizing of the word2vec method.
Doc2Vec

Distributed Memory Model

Distributed Bag of Words Model
Methodology

Figure 2: New Twitter classification workflow based on deep learning

Figure 3: The procedure of learning document vector, where N represents the number of the words in a document.
Methodology

\[ \vec{D} = \{d_1, d_2, \ldots, d_M\}, \]
where \( M \) is the dimension amount of the document vector, \( d \) is the value for each level of it.

\[ \vec{t} = (\vec{D}, \text{label}), \]
where \( \vec{t} \) represents the concatenate vector, and label is the tweet flag of spam or non-spam.

\[ T = (\vec{t_1}, \vec{t_2}, \ldots, \vec{t_N}), \]

\[ \vec{L} = (l_1, l_2, \ldots, l_n) = C(\vec{D_1}, \vec{D_2}, \ldots, \vec{D_n}), \]
where \( n \) is the tweets number of testing data.
Evaluation

- Gathered data for 10-days
  - contains 1,376,206 spam tweets and 673,836 non-spam messages
- 4 sub-datasets
  - Dataset 1 and 3: 1:1 spam to non-spam
  - Dataset 2 and 4: 1:19 spam to non-spam

<table>
<thead>
<tr>
<th>Dataset No</th>
<th>Dataset Type</th>
<th>Spam : Non-spam</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuous</td>
<td>5k : 5k</td>
</tr>
<tr>
<td>2</td>
<td>Continuous</td>
<td>5k : 95k</td>
</tr>
<tr>
<td>3</td>
<td>Random</td>
<td>5k : 5k</td>
</tr>
<tr>
<td>4</td>
<td>Random</td>
<td>5k : 95k</td>
</tr>
</tbody>
</table>
Evaluation

- Basic Setup (Java)
  - KNIME Analytics Platform
  - Windows 10, I7 CPU, 12GB RAM
- First Layer: Doc2Vec with 2% learning rate and size of 200
- Second Layer: Rotating traditional machine learning models
- Looped 100 times, Calculate mean of each performance metric
- 60-40 train test split
Evaluation

- Recall, Precision, F-Measure and Accuracy metric for each classifier
- Confusion Matrix
  - TP: number of spam tweets classified correctly
  - FP: number of non-spam tweets classified wrongly
  - TN: number of non-spam tweets classified correctly
  - FN: number of spam tweets classified wrongly

<table>
<thead>
<tr>
<th></th>
<th>Spam</th>
<th>Non-spam</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Spam</strong></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td><strong>Non-spam</strong></td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>
Evaluation

- Text-based using Deep Learning
  - Random Forest
    - process the word representation trained by WordVector Technique
  - Neural Network
    - MLP with input extracted by WordVector
  - Decision Tree
    - Greedy splitting method for tree building
Evaluation

- Traditional Text-based (Vertical Comparison)
  - Palladian
    - Ngrams text classifier
  - Naive Bayes
    - Detect words distribution in documents
  - Naive Bayes (Frequencies)
    - Term frequency
Evaluation

- Feature-based Supported by Machine Learning (Horizontal Comparison)
  - Naive Bayes
    - 2-layered, label of spam/non-spam and the other for set of features
  - Random Forest
  - Decision Tree
    - C4.5, traditional machine learning
Evaluation

- Deep Learning vs Syntax-based
  - MLP performs better in Recall, F-measure and Accuracy
  - 25% higher precision
  - Outperforms the rest

- Deep Learning vs Feature-based
  - F-Measure is 30% higher than Random Forest
  - 9 times higher than Naive Bayes
Figure 4: Performance Value of our detection method based on deep learning based on 4 sampled datasets. (A) Recall; (B) Precision; (C) F-measure; (D) Accuracy
Figure 5: Vertical Comparison of performance values between our technique and traditional text-based detection approaches based on 4 sampled datasets. (A) Recall; (B) Precision; (C) F-measure; (D) Accuracy
Evaluation - Feature SVM (HC)

Figure 6: Horizontal Comparison of performance values between our technique and feature-based methods based on 4 sampled datasets. (A) Recall; (B) Precision; (C) F-measure; (D) Accuracy
Discussion

- **Spam Ratio**
  - Performance stays the same
  - Achieves better recall of 2.45%
  - F-measure of Naive Bayes
    - 60% in 1:1 dataset
    - 12% in 1:19 dataset

![Table 4: Impact of the Spam Ratio by Dataset 1 and 2 using MLP](image)
Discussion

- **Dataset Dissection**
  - Performance is stable
  - Continuous dataset performs better than random dataset

Table 5: Impact on Sample Dataset Discretisation of Dataset 1 and 3 using MLP

<table>
<thead>
<tr>
<th>Unit: %</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>93.48</td>
<td>95.04</td>
<td>94.25</td>
<td>94.30</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>91.48</td>
<td>94.23</td>
<td>92.83</td>
<td>92.94</td>
</tr>
</tbody>
</table>
Conclusion

- Explored issues around spam detection on Twitter data
- Proposed a new classification method using DL
- Utilized WordVector techniques for pre-processing
- Computation with high-multidimensional vectors
Conclusion - Future Works

- Explore theoretical studies on the deep learning framework
- Compare against other classifiers outside from the ones mentioned
- Collect more real data from other social media platforms (Facebook)
- Study the feedback of works that cited this paper
References

● Twitter Spam Detection based on Deep Learning
  ○ https://dl.acm.org/doi/10.1145/3014812.3014815

● Word2Vec Explained

● Doc2Vec Explained
  ○ https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e
The End