Event Detection with Convolutional Neural Networks for Forensic Investigation

Bo Yang[1,2], Ning Li[1,2(✉)], Zhigang Lu[1,2], and Jianguo Jiang[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{yangbo32,lining6}@iie.ac.cn
[2] Beijing Key Laboratory of Network Security Technology, Beijing 100093, China

Abstract. Traditional approaches rely on domain expertise to acquire complicated features. Meanwhile, existing Natural Language Processing (NLP) tools and techniques are not competent to extract information from digital artifacts collected for investigation. In this paper, we propose an improved framework based on a Convolutional neural network (CNN) to capture significant clues for event identification. The experiments show that our solution achieves excellent results.

Keywords: Event detection · Convolutional neural networks · Digital investigation

1 Introduction

In digital investigations, the investigator typically has to handle substantial digital artifacts for forensics analysis. Among them, most are in the form of unstructured textual data, such as emails, chat logs, etc. The investigator searches clues from these data in order to answer questions about what happened, who caused the events, when events occurred, where, with whom they communicated, and so on. A pervasive problem is the fact that unstructured data are recorded using natural language, which is hard to understand completely by computer. This problem impedes the automation of crucial incriminating information retrieval and information extraction processes when facing a mass of textual data. Although the investigator can rely on modern digital forensics tools, such as executing keyword searches, this is often a manual process [1]. Due to the fact that the same concept is typically expressed by different terms and language styles, using keyword searches is not always effective. The quality of an analysis usually varies with the investigative experience and the expertise of each investigator [2].

With the advancement of data mining and availability of the computational re-sources to improve algorithm performance, methods of text mining using natural language processing techniques have gradually become available. By means of text mining, it makes the investigator easy to conduct content analysis and extract clues from textual data. Furthermore, automated techniques avoid threats to the privacy issues. For the reason that digital artifacts can reveal interested events, such as illegal

Z. Shi et al. (Eds.): IIP 2016, IFIP AICT 486, pp. 97–107, 2016.
DOI: 10.1007/978-3-319-48390-0_11

activities at key times, communications between suspects and victims, and other clues to the investigation, obtaining valuable information from them is based on the event detection task, which involves identification of events from specific types in the artifacts. Unlike traditional event extraction task defined in Automatic Content Extraction (ACE) evaluation [3], the event detection task in forensics varies according to case type and the investigator focuses a few specific types in a specific investigation. For example, the investigator should be interested in contact events, movement events etc. other than business events in a case of murder investigation. Current research on event extraction does not apply to digital forensics.

There are two categories of features for textual analysis on event detection: lexical-level features and sentence-level features. Convolutional neural networks that model sequential data have been proved to capture important information at sentence-level [4]. However, at the lexical-level, the same word influenced by its contexts has different meaning. Consequently, it frequently makes classifiers confused and also causes the inefficiency of keywords searches. In this paper, we present an improved one-layer convolutional neural network, which we name Classification with Similarity Vector by CNN (CSV-CNN), to capture more significant clues for event identification. Associated with each specified event is an event trigger, which most clearly expresses an event occurrence. We establish an event trigger lookup table containing most representative terms for a specific event to obtain lexical-level features. Our method computes and averages cosine similarity between each word in a sentence and word in the trigger lookup table in order to obtain a similarity vector. On the other hand, given input sentences, the network uses convolutional layers to acquire distributed vector representations of inputs to learn sentence-level features. The proposed network learns a distributed vector representation and similarity vector for each event class.

2 Preliminary

In this section, we first discuss related work in event extraction, and then introduce challenges posed by textual evidence.

2.1 Related Work

Since 1987, a series of Message Understanding Conferences (MUC) which were initiated by DARPA showed considerable interest in event extraction and encouraged the development of new methods of in information extraction [5]. MUC participants required submit evaluations to compete in textual information tasks. These tasks covered a wide range of goals, such as extracting fleet operations, identifying terrorist activities, detecting joint ventures and leadership changes in business, etc. The Automatic Content Extraction [3] program addresses the same problems as the MUC program, which defines extraction tasks according to the target objects, such as entity, relation and event. ACE focuses on annotating 8 event types and 33 subtypes.

There are several instances of the implementation of text mining techniques for event extraction can be found in literature. Best et al. [6] make use of a combination of

entity extraction and a machine-learning technique for pattern-based event extraction. Li et al. [7] proposed the crossing-entity inference to improve the traditional sentence-level event extraction task. Li et al. [8] elaborate on a unified structure for extraction of entity mentions, relations and events in ACE task. These methods achieve relatively high performance on the basis of suitable choices of features selection. However, these methods suffer from complicated feature engineering and errors from existing NLP toolkits.

Over the past few years, deep learning methods have achieved remarkable success in machine learning, especially in computer vision and speech processing tasks [9, 10]. More recently, methods of handling NLP tasks using deep learning techniques started to overtake traditional sparse, linear models [11]. Word2Vec propose by Mikolov et al. [12], which could construct representation of words into a dense, low dimensional vector under a large corpus, has drawn great interests and is widely used as pre-trained word vectors. Convolutional neural networks (CNN), originally invented for computer vision, have subsequently realized significant performance in sentence classification. Nguyen et al. [13] used CNN for event detection that automatically learned feature form pre-trained Word2Vec embeddings, position embeddings, and entity type embeddings, with promising results.

2.2 Challenges Posed by Textual Evidence

Because of the efficiency and the convenience, the internet and computers are being used by criminals to facilitate their offenses. Most types of collected digital textual evidences are short texts, such as emails, chat logs, blogs, and tweets. In the past, these textual data have been studied for mining digital evidences, but all of them faced common challenges posed by noisy characteristics. First, these texts are short and have limited context. For example, emails seldom contain more than 500 words, and tweets can have less than 140 characters. Thus, statistical methods such as topic modeling do not apply to discover the themes of texts for insufficient words [14]. Second, most of them are more informal in style. The discourse can be regarded as written speech or spoken writing [15]. People do not always observe the grammatical rules and tend to make spelling mistakes. This means traditional NLP techniques are seldom appropriate for recorded conversations [16]. Third, people tend to use shortened terms, characters or punctuation symbols to convey or express more meaning or inside feelings or moods [17]. For example, the word “thanks” can be written as “thx”, and the emoticon “:D” represents feeling happy. However, these terms or symbols are not captured by general-purpose tokenisers and not recognized as single tokens.

3 Model Description

In this paper, we formalize the event detection problem as a multi-class classification problem. Given a sentence, we want to make a prediction whether it expresses some event in the pre-defined event set or not [8]? In Subsect. 3.1, we first introduce standard

one-layer CNN for sentence classification [4]. We then propose our augmentations in Subsect. 3.2, which exploit two stages: lexical-level and sentence-level feature extraction. Figure 1 describes the architecture of the proposed CSV-CNN.

3.1 Basic CNN

In this model, we first convert a tokenized sentence to a sentence matrix. Each token in a sentence is mapped into a low-dimensional vector representation, whether it is a word or not. It may be a fixed dimensional feature vector initialized at random and updated during training, or an output from pre-trained word2vec. We denote the dimensionality of the word vectors by n. In order to perform convolution on a sentence via non-linear filters, we expand the context to the maximum sentence length s by padding shorter sentences with special tokens. It is useful because it allows us to efficiently treat our data as an "image". Thus, the dimensionality of the sentence matrix is $s \times n$. In this matrix, let each row vector denote the corresponding token so that we can retain the inherent sequential structure of a sentence. We then denote a sentence x of length n by $x = \{x_1, x_2, \ldots, x_n\}$, and the sentence matrix is represented as A $\in \mathbb{R}^{s \times n}$.

The main idea behind a convolution and pooling computation is to apply non-linear filters over each instantiation of a k-word sliding window over a sentence. A filter $w \in \mathbb{R}^{k \times n}$ converts a window of k words into a fixed dimensional vector that achieves new features of the words in the window. We vary the dimensionality k of the filter to acquire different filters or use multiple filters for the same k to learn complementary features. In order to obtain a feature c_i, the filter is applied on a sub-part of A:

$$c_i = f(w \cdot x_{i:i+k-1} + b)$$

Where $x_{i:i+k-1}$ refers to the concatenation of tokens $x_i, x_{i-1}, \ldots, x_{i+k-1}$, $b \in \mathbb{R}$ is bias term, and f is a non-linear activation function that is applied element-wise. We execute convolution operations for each possible sub-matrix of A, which is $\{x_{1:k}, x_{2:k+1}, \ldots, x_{n-k+1:n}\}$, to produce a feature map:

$$\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-k+1}]$$

Where $\boldsymbol{c} \in \mathbb{R}^{n-k+1}$. A max-pooling function is thus applied to the feature map to obtain the most salient information. Multiple filters are implemented and the outputs are concatenated into a "top-level" feature vector. Finally, this feature vector is passed to a fully connected softmax layer for classification.

Weights can be regularized in two ways. One is Dropout, in which we set values in the weight vector with a portion p at random during forward-backpropagation, the other is $l2$ norm constraint, for which we set a threshold λ for weight vectors during training; when it exceeds the threshold, we rescale the vector accordingly.

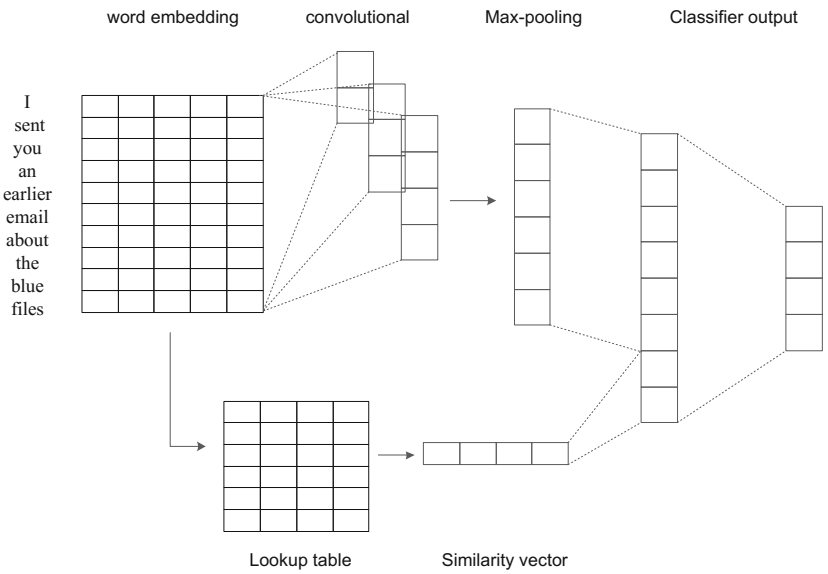


Fig. 1. Illustration of CSV-CNN

3.2 CSV-CNN

We propose an augmentation of CNN so that the model can learn more important features at the lexical-level. And the features at the sentence-level are left to learn by basic CNN model. We first select the most important words of each event type from data set. $TF - IDF$ [18] is one of the most popular algorithms used in the fields of in-formation retrieval and text mining. It provides a numerical statistic to reflect the extent of importance of a word in a collection. The occurrences of a word are directly proportional to the importance, but are inversely proportional to the frequency of the word in the collection. The $TF - IDF$ weight of word j is computed as follows:

$$TF - IDF\left(word_j\right) = TF\left(word_j\right) \times IDF\left(word_j\right)$$

Where TF represents the times a given word occurs in a specific document, IDF, which represents Inverse Document Frequency, is used to diminish the effect of words that appear too often in a collection. The IDF of $word_j$ is computed as follows:

$$IDF\left(word_j\right) = \log \frac{N}{\left(word_j\right)}$$

Where $DF\left(word_j\right)$ represents the quantity of documents containing $word_j$. We simply calculate a TF- IDF score for each word and obtain the top n important words. These words can be seen as lexical-level features of each event type, which constitute the event trigger look up table.

We use pre-trained Word2Vec vectors for our word embeddings, exclude stopwords from sentences at the same time. Inspired by Word2Vec [12], which can automatically learn relationships between words like *vec*(“king”) – *vec*(“man”) = *vec*(“queen”) – *vec*(“woman”), we compute cosine similarity between the vectors of feature words from lookup table and the vectors of words from a sentence. We then average the results as a similarity vector corresponding to a specific event type. All of similarity vectors forms the penultimate layer of CSV-CNN along with the feature vector from the max-pooling layer.

The procedure of CSV-CNN is shown in the following.

Procedure CSV-CNN

Load_data()

Input: data set file obtained from the dataset folder

Output: x, y, vocabulary, vocabulary_inv, vocabulary_vector, similiar_vector

```
for(int i1 = 1;i1<files.length;i1++)
    splits the data into words
      set the label, obtain keywords to each class(lookup table),
end
output(sentences, labels, keywords_labels)
for(int i2 = 0;i2<number of sentences;i2++)
     Pads all sentences to the same length
end
Output(sentences_padded)
for(int i3 = 0;i3<number of words;i3++)
     Builds a vocabulary mapping from word to index
     Builds a vocabulary vector matrix mapping from word to vector
End
Output(vocabulary, vocabulary_inv, vocabulary_vector)
for(int i4=0;i4<number of sentence;i4++)
      for(int i5=0;i5<number of words in a sentence;i5++)
         Maps sentencs and labels to vectors based on a vocabulary
      end
end
Ouput(x,y)
for(int i6=0;i6<number of sentence;i6++)
     cleaning for stopwords of sentence, computes similarity for
each sentence with keywords_labels
End
Ouput(similiar_vector)
```

CSV-CNN (*x, y, vocabulary, vocabulary_inv, vocabulary_vector, similiar_vector*)

```
Initialize CSV-CNN parameters;
for(int i=0;i<number of input sentence;i++)
   for(int j=0;j<number of word of each sentence;j++)
          Builds a embedding mapping from words in each sentence
to vector
    end
end
output(embedding);
for(int k = 0; k<number of windows in the convolution; k++)
     Create a convolution + maxpool layer for each filter size
end
output(feature map)
concatenate feature map with similiar_vector
softmax and regularization
output(predictions)
```

CSV-CNN training algorithm

```
Initialize CSV-CNN parameters;
for(int i=0;i< Number of training epochs;i++)
   for(training example x_batch, y_batch)
        calculates the cross-entropy loss for each class
        Backward compute_gradients
        Update_parameters(parameters, gradient)
   end
end
Output(parameters)
```

4 Experiments

4.1 Datasets

Although CNN has been demonstrated to have high performance in previous work on event detection [13], the dataset utilized for evaluation is based on newswire articles and does not apply to forensic investigation scenarios. Due to privacy constraints, actual cases and their related data are always not available for academic research. Therefore, we conducted a performance evaluation on a similar data. We utilized the Enron email dataset [19], which was made public during the legal investigation. This dataset contains messages belonging to 158 employees in Enron Corporation before its bankruptcy. We select sentences from these messages to tag events of interest. To this end, we establish four categories: movement, transaction, meet and correspondence. Each category contains 100 sentences. These four events usually occur in crime cases. Several example sentences are shown in Table 1. We performed experiments on one dataset containing 400 sentences with 1,271 unique terms (Fig. 2).

Table 1. Example sentence of dataset

Event type	Example sentence
Meet	We met with Gov Davis on Thursday evening in LA
Movement	I was at PIRA today and got a preview of their presentation on Oct 11–12
Correspondence	I sent you an earlier email about the blue files
Transaction	I transferred $10,000 out of the checking account on Monday 2/28/00

Table 2. Similar vector of example sentence

Meet	Movement	Correspondence	Transaction	Example sentence
0.08648066	0.07980033	0.04879327	0.02744178	We met with Gov Davis on Thursday evening in LA
0.0313917	0.0246127	0.04972452	0.0223587	I was at PIRA today and got a preview of their presentation on Oct 11–12
0.03896217	0.10931941	**0.28144109**	0.08396807	I sent you an earlier email about the blue files
0.0236352	0.05342571	0.07086471	0.04953432	I transferred $10,000 out of the checking account on Monday 2/28/00

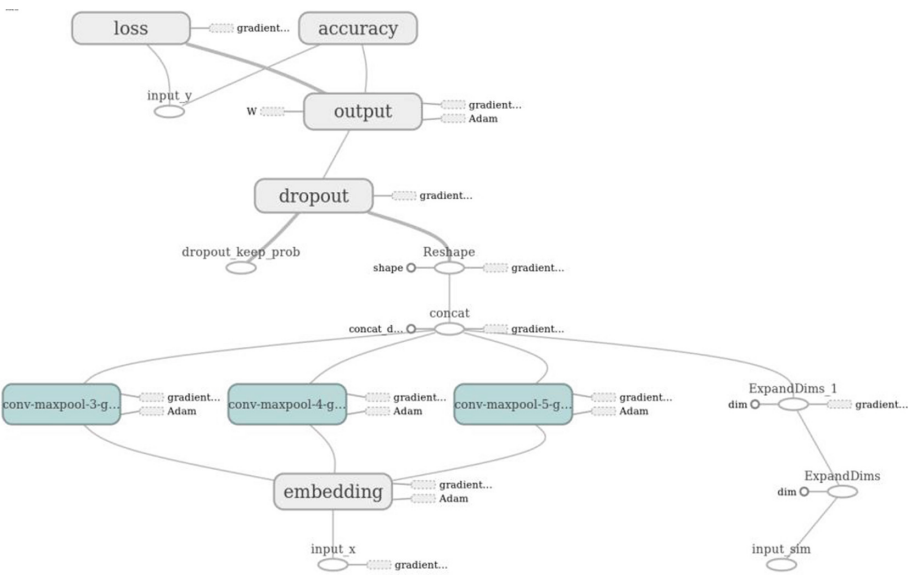
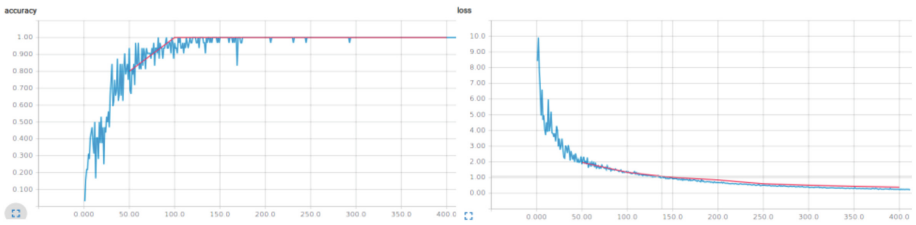


Fig. 2. Architecture of CSV-CNN

4.2 The Performance and Analysis

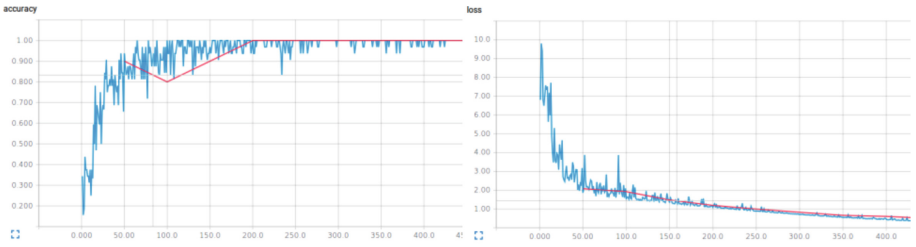
First, it demonstrates similar vector of our example sentences from our dataset as shown in Table 2. Although the similar vectors do not classify all of the example sentences independently, the first and the third ones can be classified by them. The result from Table 2 has proved the similar vector helps to find clues at the lexical-level indirectly.

The performance of CSV-CNN and standard CNN are shown in Figs. 3 and 4 respectively. Training metrics from the figures are not smooth because we use small batch sizes. It suggests that to achieve better results require a large corpus in the future work. As we can see from the figures, our model shows better at the accuracy and loss aspects.



(a) Accuracy (b) loss

Fig. 3. Accuracy and loss plots of CSV-CNN (blue is training data, red is 10 % dev data). (Color figure online)



(c) Accuracy (d) loss

Fig. 4. Accuracy and loss plots of CNN (blue is training data, red is 10 % dev data). (Color figure online)

5 Conclusions

In this paper, we present an improving framework based on CNN. We use prede-fined event typed sentences from enron dataset to evaluate the performance. The experiment shows that our solution achieves excellent results. In future work, we tend to refine our

method in extracting specific information in wider scale. We plan on testing our method over a more appropriate large corpus to evaluate.

References

1. Pollitt, M.: A history of digital forensics. In: Chow, K.P., Shenoi, S. (eds.) Advances in Digital Forensics VI. IFIP Advances in Information and Communication Technology, pp. 3–15. Springer, Heidelberg (2010)
2. Al-Zaidy, R., Fung, B.C.M., Youssef, A.M., et al.: Mining criminal networks from unstructured text documents. Digital Invest. **8**(3), 147–160 (2012)
3. ADC. https://www.ldc.upenn.edu/collaborations/past-projects/ace
4. Kim, Y.: Convolutional neural networks for sentence classification (2014). arXiv preprint arXiv:1408.5882
5. Grishman, R., Sundheim, B.: Message understanding conference - 6: a brief history. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING), vol. 1, pp. 466–471. Kopenhagen (1996)
6. Best, C., Piskorski, J., Pouliquen, B., et al.: Automating event extraction for the security domain. In: Chen, H., Yang, C.C. (eds.) Intelligence and Security Informatics: Techniques and Applications. Studies in Computational Intelligence, vol. 135, pp. 17–43. Springer, Heidelberg (2008)
7. Hong, Y., Zhang, J., Ma, B., et al.: Using cross-entity inference to improve event extraction. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 1127–1136. Association for Computational Linguistics (2011)
8. Li, Q., Ji, H., Hong, Y., et al.: Constructing information networks using one single model. In: EMNLP, pp. 1846–1851 (2014)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
10. Graves, A., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6645–6649. IEEE (2013)
11. Goldberg, Y.: A primer on neural network models for natural language processing (2015). arXiv preprint arXiv:1510.00726
12. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
13. Nguyen, T.H., Grishman, R.: Event detection and domain adaptation with convolutional neural networks, vol. 2, p. 365, Short Papers (2015)
14. Hua, W., Wang, Z., Wang, H., et al.: Short text understanding through lexical-semantic analysis. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE), pp. 495–506. IEEE (2015)
15. Kucukyilmaz, T., Cambazoglu, B.B., Aykanat, C., et al.: Chat mining: predicting user and message attributes in computer-mediated communication. Inf. Process. Manage. **44**(4), 1448–1466 (2008)
16. Agarwal, S., Godbole, S., Punjani, D., et al.: How much noise is too much: a study in automatic text classification. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 3–12. IEEE (2007)

17. Walther, J.B., D'Addario, K.P.: The impacts of emoticons on message interpretation in computer-mediated communication. Soc. Sci. Comput. Rev. **19**(3), 324–347 (2001)
18. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1986)
19. Cohen, W.W.: Enron email dataset. http://www.cs.cmu.edu/~enron/. Accessed 21 Aug 2009
20. Bach, J.: Modeling motivation in microPsi 2. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 3–13. Springer, Heidelberg (2015)
21. Abadi, M., Agarwal, A., Barham, P., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems (2016). arXiv preprint arXiv:1603.04467