Classification of Tweets about Reported Events using Neural Networks

Kiminobu Makino, Yuka Takei, Taro Miyazaki, Jun Goto
NHK Science & Technology Research Laboratories
1-10-11 Kinuta, Setagaya-ku, Tokyo, Japan
{makino.k-gg, takei.y-ek, miyazaki.t-jw, goto.j-fw}@nhk.or.jp

Abstract

We developed a system that automatically extracts "Event-describing Tweets" which include incidents or accidents information for creating news reports. Event-describing Tweets can be classified into "Reported-event Tweets" and "New-information Tweets." Reported-event Tweets cite news agencies or user generated content sites, and New-information Tweets are other Event-describing Tweets. A system is needed to classify them so that creators of factual TV programs can use them in their productions. Proposing this Tweet classification task is one of the contributions of this paper, because no prior papers have used the same task even though program creators and other events information collectors have to do it to extract required information from social networking sites. To classify Tweets in this task, this paper proposes a method to input and concatenate character and word sequences in Japanese Tweets by using convolutional neural networks. This proposed method is another contribution of this paper. For comparison, character or word input methods and other neural networks are also used. Results show that a system using the proposed method and architectures can classify Tweets with an F1 score of 88 %.

1 Introduction

Many companies including news agencies have increasingly been extracting news information from postings on Social Networking Sites (SNSs) such as Twitter and Facebook and using it for various purposes (Neubig et al., 2011; Iso et al., 2016). However, choosing important information for news reports from Twitter is very tough, because Twitter contains a vast amount of posts. For this reason, many researchers have studied how to extract important posts for each purpose (Papadopoulos et al., 2014; Litvak et al., 2016; Zhou et al., 2016; Vakulenko et al., 2017). A system using Neural Networks (NNs) has been developed by using models that are trained by extracting Tweets in factual TV program production, and these systems extract "Event-describing Tweets (EVENT)" which include incidents or accidents information for news reports from a large amount of Tweets (Miyazaki et al., 2017). However, there are many Tweets, so there can be many extracted Tweets which include EVENT for news reports about any event. Hence, people have difficulty monitoring all EVENT. In addition, EVENT are used differently in different types of programs. For these purposes, it is better to display only Tweets suitable to the program contents.

For example, program creators who want to obtain primary reports of an event posted by Twitter users do not require Tweets put out by news agencies and User Generated Content (UGC) sites or Tweets that quote or cite them. The part of new information of these Tweets is able to be gotten by crawling each site, so no longer these Tweets do not include new events information for program creators. We call these Tweets "I: Reported-event Tweets (REPORTED)" and others "II: New-information Tweets (NEW)." Both types of Tweets are requested for different reasons. Only types of Tweets suitable to creators' purpose need to be displayed, but extracting EVENT and classifying them are essentially different processes.

For these reasons, this paper uses a two-stage processing system that separates EVENT from a large amount of Tweets by using an existing system and classifies them into REPORTED and NEW by using text-based machine learning methods in real-time (Section 4). Our proposed method inputs both character and word sequences (Section 5.2). Both proposed and conventional methods use several NN architectures including Recurrent NNs (RNNs) and Convolutional NNs (CNNs) (Sec-

Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, pages 153–163
Brussels, Belgium, Nov 1, 2018.

tion 5). Evaluation results show that the proposed method outperformed the conventional methods in all NN architectures (Section 6).

This paper makes two contributions. One is proposing a new task for classifying extracted EVENT into two classes: REPORTED and NEW. TV program creators need to do this task for program production automatically and do it first to track reports about an event. However, there have been no prior studies about this task. The other is proposing a new method for NN architectures that inputs entire character sequences and entire word sequences in parallel and concatenates them in the intermediate layer and evaluating its performance. This method can utilize the advantages of character sequences (i.e., there are fewer unknown characters than unknown words and it does not need morphological analyzers even when in Japanese which is very difficult to divide words especially for noisy texts) and word sequences (i.e., words are more effective than characters for the task). The method can be used for any other tasks that need to both character and word sequences.

2 Related Work

There are many related works such as related tasks that use Twitter datasets or classify texts and related methods that have NNs architectures using both characters and words in Natural Language Processing (NLP).

Related tasks include topic detection on Twitter task (Papadopoulos et al., 2014; Litvak et al., 2016; Zhou et al., 2016; Vakulenko et al., 2017), binary classification of Tweets (Rosenthal et al., 2017), classification of news related or political stance Tweets (Ghelani et al., 2017; Johnson and Goldwasser, 2016; Volkova et al., 2017), classification of news related articles (Ribeiro et al., 2017), and other classifications in NLP. Binary classification of texts and classification of news related texts and articles are most closely related to this task. However, none of these studies focused on classifying extracted EVENT into REPORTED or NEW. Since no classification method meets the requirements of this paper (i.e., extraction of REPORTED to obtain primary reports and extraction of NEW to collect opinions about reported events or gather follow-up Tweets), no prior research on the same task exists.

For **related methods**, in NLP using machine learning, there is one NLP configuration that uses a word sequence as input and characters as supplemental information (Ma and Hovy, 2016; Grönroos et al., 2017; Heyman et al., 2017; Lin et al., 2017) and another that switches between the character NN and the word NN (Vijayaraghavan et al., 2016). However, the character sequence is one semantic vector set for the entire sequence. There is one NLP configuration that uses gated recurrent units for a word sequence and for CNN output of a character sequence(Liang et al., 2017). However, it does not purely combine characters and words in parallel and takes time to process because it includes recurrent architectures and is not for noisy texts. No method combining the output of an entire purely character sequence and an entire purely word sequence in parallel with a CNN for noisy texts has been studied and evaluated to the best of the authors' knowledge.

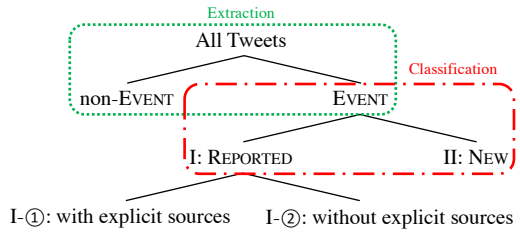

Figure 1: Overview of our Tweets classification.

3 Task Description

The purpose of our system is classification of Tweets for different types of programs. Figure 1 shows the overview of our Tweets classification. Extracting EVENT is not a novel task (Miyazaki et al., 2017), so the proposed task in this paper is classifying EVENT into REPORTED and NEW.

3.1 Classification of REPORTED and NEW

REPORTED are less numerous than NEW. This is because NEW include information about events relevant to few people and that are low priority for many news agencies such as local events as well as events no news agencies know about. Tweet-classification system needs to extract all events information, and the program creators need to judge the priority. Conversely, when Tweets are put out that many people want to cite and opine about, REPORTED quoting these Tweets will increase,

I-①	REPORTED **with explicit sources**	EVENT that are tweeted from or quote news agencies, UGC sites, or others and **include** explicit sources
e.g.		
"Small plane crash. Four people dead on impact. — XXX news I wonder if the plane broke down from the nose."		
I-②	REPORTED **without explicit sources**	EVENT that are tweeted from or quote news agencies, UGC sites, or others and **do not include** explicit sources
e.g.		
"Small plane crash. Four people dead on impact. I wonder if the plane broke down from the nose."		
II	NEW	Any other EVENT

Table 1: Types of EVENT (example is manually translated by author).

and NEW will be hard to monitor and gather in real-time. Similarly, creators who want to collect opinions about reported events or gather follow-up Tweets will not need NEW, which are the majority of EVENT. Therefore, depending on the creator's intention, either NEW or REPORTED should be displayed in real-time. For these reasons, a classification task is needed.

3.2 Two Types of REPORTED

There are two types of REPORTED. One is REPORTED with explicit sources (I-①), which cite news agencies, UGC sites, or other information dissemination agencies, so Tweet-classification systems are expected to easily detect these Tweets by keyword filtering using source names. The other is REPORTED without explicit sources (I-②), which do not cite explicit sources because Twitter users can remove source names. They have a distinctive stylistic character (in Japanese, they often contain sentence ending with a noun or noun phrase and often include date, time, etc.) and can be detected manually. However Tweet-classification systems cannot detect them by simple methods including keyword filtering using source names.

Table 1 shows three types of training data (described in Section 6.1) manually classified by humans. Both I-① and I-② are REPORTED, and this system only classifies Tweets into **two classes**: I: REPORTED and II: NEW. This is because I-① and I-② seem to be used the same way. However, extracting I-② is expected to be harder than extracting I-①, because sources are grounds for deciding whether a Tweet quotes a source or not. For only evaluating the characteristic difference (Section 6), I are classified into I-① and I-②.

4 Configuration of Our System

The structure of our system for classifying EVENT about reported events is shown in Figure 2. Inputs of this system are 10 % of all randomly sampled Tweets in Japanese. The extraction process extracts EVENT and removes non-Event-describing Tweets (non-EVENT). The EVENT for news reports are then input in the classification process, which classifies them into REPORTED and NEW.

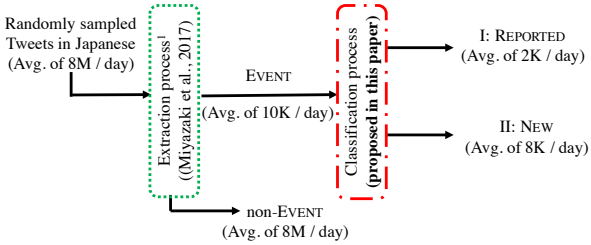

Figure 2: Structure of our system for classifying EVENT about reported events.

The extraction process needs to separate the 0.1% of EVENT from the 99.9% of non-EVENT. Because there are so many non-EVENT, the extraction process needs extensive training. To extract EVENT and classify them into REPORTED and NEW in one process, the system is trained for classification by using Tweets required for training with a large amount of non-EVENT unrelated to classification. Moreover, when systems are extended to classify other types of Tweets or relearn how to classify EVENT about reported events, they need extensive training for Tweet extraction and classification. However, it is not realistic to do such retraining every time classification is adjusted in accordance with a program creator's request. For these reasons, this paper uses a two-stage processing system that includes an extraction process and a classification process.

4.1 Extraction Process

The extraction process uses an existing method (Miyazaki et al., 2017). Figure 3 shows the structure of the extraction process. Tweets are converted into one-hot vector for each character, entered into a Feed-forward NN (FFNN), a Bi-directional Long Short-Term Memory (Bi-LSTM) with an attention mechanism, and 2-layer

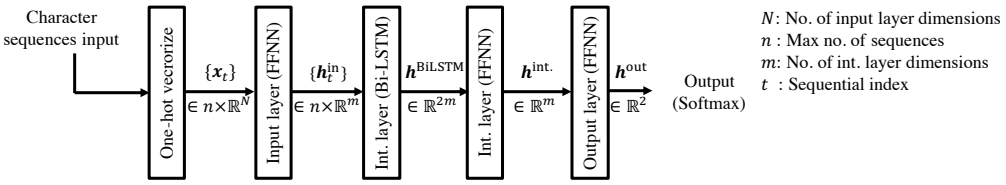

Figure 3: Structure of extraction process.

FFNN, and classified as important (EVENT) or unimportant (non-EVENT). Dimensions of each intermediate layer are set to 200.

In this paper, a model that is trained by supervised training using 19,962 Tweets manually extracted in TV program production for positive samples and 1,524,155 randomly sampled Tweets for negative samples is used. The model which has a 74.4 % F1 score is used. Whether the priority is precision or recall can be changed by varying the threshold of the output depending on the purpose[1].

4.2 Classification Process

The classification process classifies EVENT into REPORTED and NEW by several classification methods using three types of manually classified training data as shown in Table 1. For reasons already mentioned in Section 3.2, this process classified EVENT into REPORTED and NEW.

Input in the classification process is limited to EVENT extracted from the extraction process, so the classification process needs much less training data than the extraction processes. There is a trade-off between the hardware burden caused by the volume of training data, structures of NNs and the improvement of classification accuracy by advanced processing. However, the classification process does not need extensive training and so can use computationally heavy methods within the range where the test phase is performed in real-time. In this paper, the accuracy and leaning speed of these methods are evaluated in experiments.

5 Classification Methods using NNs

For methods to classify REPORTED and NEW, several inputs including the proposed method and several NN architectures are used. In machine learning using sentences, an input sequence is generally divided into characters or words, vectorized, serialized, and used. The contributions of the both words and characters are evaluated by these three methods.

[1]The extraction process is neither the purpose nor the contribution of this paper. This existing method was used only for convenience. The performance evaluation of this paper is for the classification process. When EVENT extracted by any methods are input, the methods are expected to perform approximately the same for the classification process.

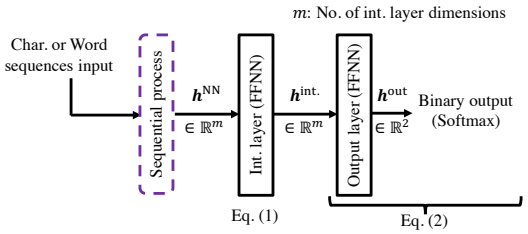

Figure 4: Overview structure using each character sequences or word sequences.

5.1 Conventional Character Input NN / Word Input NN

Figure 4 shows an overview of a structure using either character sequences or word sequences. First, sentences are input to a sequential process NN and output as $\boldsymbol{h}^{\text{NN}} \in \mathbb{R}^m$, where m is the intermediate size. Second, $\boldsymbol{h}^{\text{NN}}$ is inputed to the intermediate layer FFNN ($\boldsymbol{W}^{\text{int.}} \in \mathbb{R}^{m \times m}$, $\boldsymbol{b}^{\text{int.}} \in \mathbb{R}^m$) and output as $\boldsymbol{h}^{\text{int.}} \in \mathbb{R}^m$. Finally, $\boldsymbol{h}^{\text{int.}}$ is input to the output layer FFNN ($\boldsymbol{W}^{\text{out}} \in \mathbb{R}^{2 \times m}$, $\boldsymbol{b}^{\text{out}} \in \mathbb{R}^2$) and the Softmax function and output as binary of classification results. At the training phase, loss is calculated by the cross entropy function between output of the Softmax function and one-hot vector of a correct answer. At the test or use phase, output is calculated by an argmax function of the Softmax function output. A series of processes is obtained as follows,

$$\boldsymbol{h}^{\text{int.}} = a(\boldsymbol{W}^{\text{int.}} \boldsymbol{h}^{\text{NN}} + \boldsymbol{b}^{\text{int.}}) \quad (1)$$

$$output = \text{softmax}(\boldsymbol{h}^{\text{out}}) = \text{softmax}(\boldsymbol{W}^{\text{out}} \boldsymbol{h}^{\text{int.}} + \boldsymbol{b}^{\text{out}}), \quad (2)$$

where $a(\cdot)$ is an activation function and we use a Rectified Linear Unit (ReLU) (Clevert et al.,

2015). For converting sentences into character sequences, sentences are separated into individual characters. For converting sentences into word sequences, sentences are separated using the Japanese morphological analyzer MeCab (Kudo et al., 2004) with the customized system dictionary mecab-ipadic-NEologd (Sato, 2015).

5.1.1 FFNN for Sequential Process

Figure 5 shows the structure of character or word sequences input only using the FFNN. A BOW (Bag of Words / characters) vector $\boldsymbol{x}^{\mathrm{BOW}} \in \mathbb{R}^N$ is input to the input layer FFNN ($\boldsymbol{W}^{\mathrm{in}} \in \mathbb{R}^{m \times N}$, $\boldsymbol{b}^{\mathrm{in}} \in \mathbb{R}^m$) and output as $\boldsymbol{h}^{\mathrm{in}} \in \mathbb{R}^m$, where N is the number of input layer dimensions, so FFNN architectures do not include sequential architectures. A series of processes is obtained as follows,

$$\boldsymbol{h}^{\mathrm{FFNN}} = \boldsymbol{h}^{\mathrm{in}} = a(\boldsymbol{W}^{\mathrm{in}}\boldsymbol{x}^{\mathrm{BOW}} + \boldsymbol{b}^{\mathrm{in}}). \quad (3)$$

Then, $\boldsymbol{h}^{\mathrm{FFNN}}$ is fed to Equation (1) as $\boldsymbol{h}^{\mathrm{NN}}$.

5.1.2 LSTM for Sequential Process

Figure 6 shows the structure of character or word sequences input using a LSTM for the intermediate layer. One-hot vector $\{\boldsymbol{x}_t\} = \{\boldsymbol{x}_0 \in \mathbb{R}^N, \boldsymbol{x}_1 \in \mathbb{R}^N, \cdots\}$ is input to the input layer FFNN ($\boldsymbol{W}^{\mathrm{in}}$, $\boldsymbol{b}^{\mathrm{in}}$) one by one, and output sequences are $\{\boldsymbol{h}_t^{\mathrm{in}}\} = \{\boldsymbol{h}_0^{\mathrm{in}} \in \mathbb{R}^m, \boldsymbol{h}_1^{\mathrm{in}} \in \mathbb{R}^m, \cdots\}$. After that, the output sequences are input to the intermediate layer LSTM using an attention mechanism (Bahdanau et al., 2014) one by one, and the output vector is $\boldsymbol{h}^{\mathrm{LSTM}} \in \mathbb{R}^m$. In accordance with LSTM mechanisms, all series of input are used for training. A series of processes is obtained as follows,

$$\begin{aligned}
\boldsymbol{h}_t^{\mathrm{in}} &= a(\boldsymbol{W}^{\mathrm{in}}\boldsymbol{x}_t + \boldsymbol{b}^{\mathrm{in}}) \quad (t \in [0, n)) \quad (4)\\
\boldsymbol{z}_t &= \tanh(\boldsymbol{W}^{\mathrm{z}}\boldsymbol{h}_t^{\mathrm{in}} + \boldsymbol{R}^{\mathrm{z}}\boldsymbol{h}_{t-1}^{\mathrm{inc.}} + \boldsymbol{b}^{\mathrm{z}}) \quad (5)\\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}^{\mathrm{i}}\boldsymbol{h}_t^{\mathrm{in}} + \boldsymbol{R}^{\mathrm{i}}\boldsymbol{h}_{t-1}^{\mathrm{inc.}} + \boldsymbol{b}^{\mathrm{i}})\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}^{\mathrm{f}}\boldsymbol{h}_t^{\mathrm{in}} + \boldsymbol{R}^{\mathrm{f}}\boldsymbol{h}_{t-1}^{\mathrm{inc.}} + \boldsymbol{b}^{\mathrm{f}})\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \otimes \boldsymbol{z}_t + \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1}\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}^{\mathrm{o}}\boldsymbol{h}_t^{\mathrm{in}} + \boldsymbol{R}^{\mathrm{o}}\boldsymbol{h}_{t-1}^{\mathrm{inc.}} + \boldsymbol{b}^{\mathrm{o}})\\
\boldsymbol{h}_t^{\mathrm{inc.}} &= \boldsymbol{o}_t \otimes \tanh(\boldsymbol{c}_t)\\
\alpha_t &= \frac{\exp\left(\boldsymbol{h}_{l-1}^{\mathrm{inc.}} \cdot \boldsymbol{h}_t^{\mathrm{inc.}}\right)}{\Sigma_{j=t}^{l-1}\exp\left(\boldsymbol{h}_j^{\mathrm{inc.}} \cdot \boldsymbol{h}_t^{\mathrm{inc.}}\right)}\\
\boldsymbol{h}^{\mathrm{LSTM}} &= a\left(\boldsymbol{o}_{l-1} + a\left(\Sigma_{j=0}^{l-1}\alpha_j\boldsymbol{h}_j^{\mathrm{inc.}}\right)\right),
\end{aligned}$$

where n is the maximum number of input sequences and $\boldsymbol{W}^* \in \mathbb{R}^{m \times m}$, $\boldsymbol{R}^* \in \mathbb{R}^{m \times m}$, and $\boldsymbol{b}^* \in \mathbb{R}^m$ in Equations (5) are LSTM parameters ($*$ is each layer name and z is the tanh layer, i the input layer, f the forget layer, and o the output layer). Then, $\boldsymbol{h}^{\mathrm{LSTM}}$ is fed to Equation (1) as $\boldsymbol{h}^{\mathrm{NN}}$.

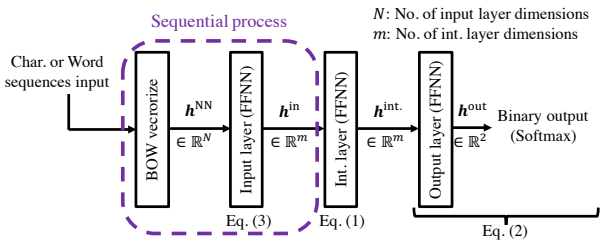

Figure 5: Structure of FFNN using each character sequence or word sequence.

5.1.3 CNN for Sequential Process

Figure 7 shows the structure of character or word sequences input using a CNN for the intermediate layer. When using word input, this process is same as (Zhang and Wallace, 2017). First, one-hot vector $\{\boldsymbol{x}_t\}$ is input to the input layer FFNN ($\boldsymbol{W}^{\mathrm{in}}$, $\boldsymbol{b}^{\mathrm{in}}$) one by one, and output sequences are $\{\boldsymbol{h}_t^{\mathrm{in}}\}$ the same for the LSTM. After that, the output sequences are input to the intermediate convolutional layer (each $\boldsymbol{W}_j^p \in \mathbb{R}^{m \times m}$,$\boldsymbol{b}^p \in \mathbb{R}^m$) using l kinds of filters (filter index is $p = [0, l)$, each filter size is k, and the index in each filter is $j = [0, k)$) with zero padding and input to max-pooling in each filter. Output is l kinds of vectors $\{\boldsymbol{h}^{\mathrm{pool},p} \in \mathbb{R}^m\}$, which are all input to the intermediate layer FFNN ($\boldsymbol{W}^{\mathrm{CNN}} \in \mathbb{R}^{m \times lm}$, $\boldsymbol{b}^{\mathrm{CNN}} \in \mathbb{R}^m$). In accordance with the CNN architectures, a part of a time series relies on k. A series of processes is obtained as follows,

$$\begin{aligned}
\boldsymbol{h}_t^{\mathrm{in}} &= a(\boldsymbol{W}^{\mathrm{in}}\boldsymbol{x}_t + \boldsymbol{b}^{\mathrm{in}}) \quad (t \in [0, n)) \quad (6)\\
\boldsymbol{h}_t^{\mathrm{Conv.},p} &= a\left(\Sigma_{j=0}^{k-1}\left(\boldsymbol{W}_j^p\boldsymbol{h}_{t+j}^{\mathrm{in}} + \boldsymbol{b}^p\right)\right) \quad (7)\\
&\quad \left(p \in [0, l), \boldsymbol{h}_q^{\mathrm{in}} = \mathbb{0}^m \ \ (q \geqq n)\right)\\
\boldsymbol{h}^{\mathrm{pool},p} &= \max_t\left\{\boldsymbol{h}_t^{\mathrm{Conv.},p}\right\} \quad (8)\\
\boldsymbol{h}^{\mathrm{CNN}} &= a(\boldsymbol{W}^{\mathrm{CNN}}\left[\boldsymbol{h}^{\mathrm{pool},0}; \cdots; \boldsymbol{h}^{\mathrm{pool},l-1}\right]\\
&\quad + \boldsymbol{b}^{\mathrm{CNN}}). \quad (9)
\end{aligned}$$

Then, $\boldsymbol{h}^{\mathrm{CNN}}$ is fed to Equation (2) as $\boldsymbol{h}^{\mathrm{int.}}$.

5.2 Proposed Concat Input NN (iii)

This paper proposes a method to input character and word sequences and to concatenate them at

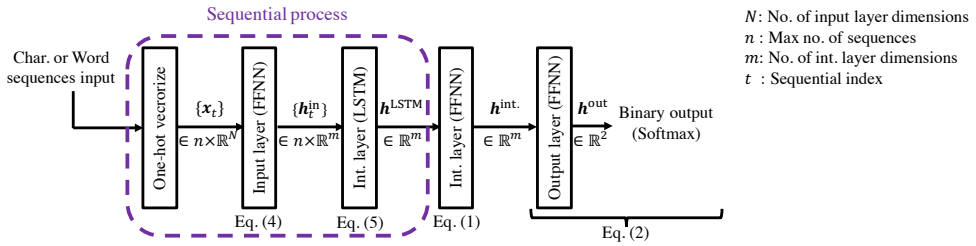

Figure 6: Structure of LSTM using each character sequence or word sequence.

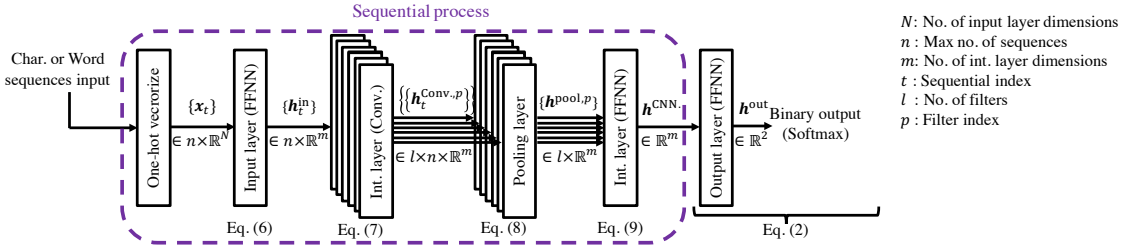

Figure 7: Structure of CNN using each character sequence or word sequence.

the intermediate layer. In Section 5.1, all architectures use only a character or word sequence. However, character sequences have the advantages of there being fewer characters than words and of expressing the input sentence without using high-dimensional input layers. Moreover, in the case of using a written language that does not have spaces between words such as Japanese, morphological analyzers are needed to divide words. Tweets are noisy, so they are very difficult to morphologically analyze accurately. Thus, performance from character sequences does not depend on morphological analyzer performance, which is another big advantage. However, sentences are written by using word sequences, and characters are involved in many words that cover a large number of meanings. In contrast, one word has a limited number of meanings and plays a bigger role in each sentence. For these reasons, the proposed method is expected to exploit the advantages of both characters and words.

Figure 8 shows the structure of character and word sequences input and concatenated at the intermediate layer. Each sequential process NN is described in Section 5.1.1-5.1.3 and surrounded by broken-line boxes in Figures. 5-7 for character sequences and word sequences independently. Output of the each sequential process is $\boldsymbol{h}^{\mathrm{NN}}$. After that, character and word sequences are concatenated, and the subsequent process is the same as that in Section 5.1. A series of processes is obtained as follows,

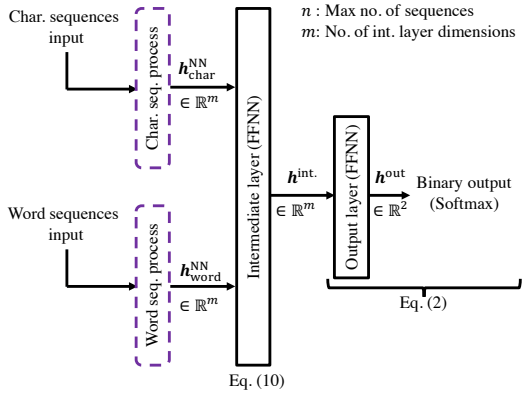

Figure 8: Overview of structure inputting both character and word sequences.

$$\boldsymbol{h}^{\mathrm{int.}} = a\left(\boldsymbol{W}^{\mathrm{int.}}\left[\boldsymbol{h}^{\mathrm{NN}}_{\mathrm{char}}; \boldsymbol{h}^{\mathrm{NN}}_{\mathrm{word}}\right] + \boldsymbol{b}^{\mathrm{int.}}\right) \quad (10)$$

where the intermediate layer FFNN is ($\boldsymbol{W}^{\mathrm{int.}} \in \mathbb{R}^{m\times 2m}$, $\boldsymbol{b}^{\mathrm{int.}} \in \mathbb{R}^{m}$). Then, $\boldsymbol{h}^{\mathrm{int.}}$ is fed to Equation (2).

6 Experimental Evaluation

The performances of classification methods are evaluated in an experimental evaluation. For comparison, baseline methods are used that use keyword filtering or Support Vector Machines (SVMs) (Vapnik and Lerner, 1963), which are well known to having high classification performance (Wang and Manning, 2012). In this paper,

	I-①	I-②	II	Total	Date
Train.	4,273	5,027	35,370	44,670	Jun., 2017
Test	844	1,184	7,972	10,000	Jul., 2017

Table 2: No. of each Tweets.

594 sources of REPORTED from training data are used for the keyword filtering baseline and LinearSVC modules of scikit-learn (Pedregosa et al., 2011) are used for the SVM baseline. Thus, data include REPORTED of various sources. In addition, the Word2Vec vector for the SVM baseline is the average of each 200–dimension word vector that is made with a Wikipedia dump corpus by using Word2Vec skip-gram modules (Mikolov et al., 2013).

6.1 Experimental Conditions

For both training data and test data, EVENT for news reports extracted by the extraction process in Section 4.1 are used. Training data is all 44,670 Tweets obtained in the extraction process on June 6th, 8th, 10th, and 12th, 2017. Test data is 10,000 randomly sampled Tweets obtained in the extraction process output Tweets on July 6th, 8th, 10th, and 12th, 2017. Output data is annotated into three categories I-①: REPORTED with explicit sources, I-②: REPORTED without explicit sources, and II: NEW by an annotator person. Table 2 shows the amount of each type of annotated data. Table 3 shows the configuration of experimental parameters.

6.2 Experimental Results

Table 4 shows precision, recall, and F1 score for each method with input as character, word, or character and word. Table 5 shows recall performance of using REPORTED to evaluate the performance with and without explicit sources. False negatives are judged for only REPORTED and cannot be divided into REPORTED with and without explicit sources, thus precision and F1 score cannot be used in this evaluation. Table 6 shows the time required to learn each NN. Finally, Figure 9 shows the F1 score of SVM baseline methods and CNN architectures trained by each number of random sampled training data.

From Table 4, the keyword baseline method has 94.1% precision and 34.0% recall. From Table 5, its recall is 73.5 % lower when using only I-② than when using only I-① (3.4 % vs. 76.9 %).

All NN architectures outperform all baseline methods. Although word input using FFNN, which has the lowest F1 score of the NN architectures, has the same precision as the SVM word input baseline method, which has highest F1 score of the baseline methods, it has higher recall (76.7 % vs. 75.7 %) and F1 score (84.2 % vs. 83.6 %). Its recall is 22.5 % lower when using only I-② than when using only I-① (67.3 % vs. 89.8 %).

In each NN architecture, the F1 score for character input is 0.3–2.0% higher than for word input. When both characters and words are input, LSTM has the highest F1 score, 0.5-2.2% higher than those of other NNs. When the conventional method is used, the highest F1 score so far is 86.7 % for LSTM architecture using character input. For this LSTM recall is 15.8 % higher when using only I-① than when using only I-② (94.6 % vs. 78.8 %).

The proposed concat input method has a higher F1 score than character input for each NN architecture. Especially, F1 scores for the LSTM and CNN architectures improved from 86.7 % and 86.2 % to 88.2%. With the proposed concat input method, the difference between recall for CNNs with and without explicit sources is only 18.5 % (95% vs. 76.5%), whereas it is 22.1% and 23.8% with character and word input NNs. In addition, the training time of the CNN architecture is almost 1/3 that of the LSTM architecture, as shown in Table 6.

From Figure 9, baseline SVM methods have higher F1 scores than CNN architectures when they are trained by using fewer than 10,000 training data. However, CNN architectures have higher F1 scores than SVM methods when they are trained by using more than 25,000 training data. The architecture using the proposed method has a lower F1 score than other architectures when trained by using fewer than 13,000 training data but has the highest F1 score when trained by using more than 20,000 training data.

6.3 Experimental Result Discussion

For the keyword baseline method, since Tweets with the same source used as training data can be detected for I-① in test data, there are few false detections and overall precision is relatively high. However, the keyword baseline method can barely detect Tweets for I-②. In contrast, all other methods can obtain recall rates of at least 67 % for I-②. This result indicates machine learning meth-

Deep learning framework	Chainer (Tokui et al., 2015)
Gradient descent algorithm	Adam (Kingma and Ba, 2014)
No. of iterations	5
Dim. of each int. layer	200
Mini batch size	100
Dropout (Srivastava et al., 2014) ratio	0.5 (except for the conv. layer and the pooling layer)
Each CNN filter size	2, 2, 3, 3, 4, 4 (for CNN (Zhang and Wallace, 2017))
Initial values of NNs	random
Unknown elements (character and word)	Elements that appear fewer than 10 times in training data
Classification	I: REPORTED and II: NEW (described in Section 4.2)
Evaluating measures	The macro average of 20 trials precision, recall, and F1 score

Table 3: Configuration of experimental parameters.

Input	NN arch.	Pre.	Rec.	F1
Char. input	**FFNN**	92.1	79.3	85.2
	LSTM	86.6	85.4	86.7
	CNN	94.3	79.9	86.2
Word input	**FFNN**	93.3	76.7	84.2
	LSTM	90.3	83.0	86.4
	CNN	93.8	76.9	84.2
Concat input (Proposed)	**FFNN**	93.2	79.6	85.8
	LSTM	91.3	**85.5**	**88.2**
	CNN	93.0	84.2	**88.2**
Baseline (Keyword)		**94.1**	34.0	50.0
Baseline (SVM Char. input)		90.1	76.9	83.0
Baseline (SVM Word input)		93.3	75.7	83.6
Baseline (SVM Word2Vec input)		85.8	80.6	83.1

Table 4: Macro average performance of proposed classification methods (%).

Input	NN arch.	I-①	I-②	I
Char. input	**FFNN**	92.0	70.3	79.3
	LSTM	94.6	**78.8**	85.4
	CNN	92.8	70.7	79.9
Word input	**FFNN**	89.8	67.3	76.7
	LSTM	93.5	75.6	83.0
	CNN	90.8	67.0	76.9
Concat input (Proposed)	**FFNN**	92.0	70.8	79.6
	LSTM	**95.3**	78.5	**85.5**
	CNN	95.0	76.5	84.2
Baseline (Keyword)		76.9	3.4	34.0
Baseline (SVM Char. input)		89.3	68.1	76.9
Baseline (SVM Word input)		87.9	67.1	75.7
Baseline (SVM Word2Vec input)		88.5	74.9	80.6

Table 5: Macro average recall performance for types of each test data (%).

Input \ NN arch.	FFNN	LSTM	CNN
Char. input	60	675	86
Word input	**40**	615	284
Concat input (Proposed)	110	1,200	440

Table 6: Training time of NNs (seconds).

ods are effective for the Tweet classification task in this paper. Moreover, all NN architectures have higher F1 score than SVM methods. This result indicates NN architectures are more effective than SVM methods for the task, especially when they have enough training data.

The CNN or LSTM architectures have higher F1 scores than the FFNN architecture for almost all kinds of input. From this fact, incorporating time series into the learning structure contributes to classifying REPORTED and NEW. CNN architectures have a higher precision but a lower recall than LSTM architectures. Especially for I-②, which is expected to present a higher degree of difficulty than I-①, CNN architectures using character or word input have 67.0 –70.7% recall whereas LSTM architectures have 75.6–78.8 % recall. These results are due to the difference in structure: a CNN uses time series only within the filter size, whereas an LSTM uses the time series of the entire sequence.

In all NN architectures, the proposed concat input method has the best F1 score, followed by character input and word input. It is considered that the advantage of the character sequences described in Section 5.2 exceeds the advantage of the word sequences, and in the proposed concat input method, both elements are automatically selected, so NN architectures are trained more effectively. The CNN architecture was particularly improved, with its recall increasing to 84.2 % for all reported Tweets (I) and to 76.5 % for I-②, only slightly lower than the recall for the LSTM architecture. As a result, when the proposed concat input method is used, though the CNN architecture requires only 37 minutes at 5 epochs, it has almost the same F1 score as the LSTM architecture because the concat input method utilizes the advantages of character sequences (i.e., there are fewer unknown characters than unknown words), and word sequences (i.e., words are more effective than characters for the task).

Considering real-world use, training data can be

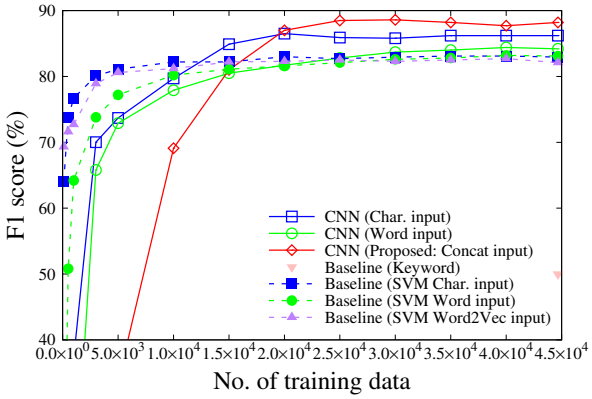

Figure 9: Macro average F1 score for each training data size reduced by all samples.

gathered on the basis of feedback from the TV program production, so training each time the kind of classification changes is also assumed. Under these conditions, a short training time is highly convenient and is a big advantage. From this result, a CNN using the proposed method has the best balance of speed and accuracy, so it is the most suitable for our system.

CNN architectures trained by using a few training data (less than 10,000) have lower F1 scores than SVM methods, seems to be caused by CNN architectures having many training parameters. Specifically, the CNN architecture using the proposed method has the lowest F1 score when it has the most training parameters. However, it has the highest F1 score when it is trained by using a lot of training data. When collecting at least 10,000 and ideally more than 30,000 training data, NN architectures are effective for the classification task in this paper.

7 Conclusion

We developed a system to classify extracted "Event-describing Tweets" for news reports into "Reported-event Tweets" to obtain reports after a primary report or collect opinions and "New-information Tweets" to obtain primary reports and for tracking reports of the same event. A convolutional neural network could classify Tweets with an F1 score of 88 % by using our proposed method, which inputs character and word sequences, concatenates them in the intermediate layer, and outputs them within 37 minutes training time. However, systems using the proposed method also incorrectly extracted Tweets including opinions about news or reports after the primary report without citations. In the future, on the basis of the output of the system using the proposed method, we will consider extending the method to the systems collecting Tweets that mention the same topic, which are used in event detection tasks. This will make it easier for TV program creators to acquire the information they want. The proposed task is important for our systems, so we increase the reliability of datasets by using more annotators. Moreover, we will consider using other tasks for evaluating proposed methods.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR 2015, pages 1–15.

Djork-Arné Clevert, Andreas Mayr, Thomas Unterthiner, and Sepp Hochreiter. 2015. Rectified factor networks. In *Advances in Neural Information Processing Systems 28*, NIPS 2015, pages 1855–1863. Curran Associates, Inc.

Nimesh Ghelani, Salman Mohammed, Shine Wang, and Jimmy Lin. 2017. Event detection on curated tweet streams. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1325–1328, New York, NY, USA. ACM.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2017. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*, WMT, pages 296–302. Association for Computational Linguistics.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2017. Bilingual lexicon induction by learning to combine word-level and character-level representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, EACL 2017, pages 1085–1095. Association for Computational Linguistics.

Hayate Iso, Shoko Wakamiya, and Eiji Aramaki. 2016. Forecasting word model: Twitter-based influenza surveillance and prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 76–86. The COLING 2016 Organizing Committee.

Kristen Johnson and Dan Goldwasser. 2016. "All I know about politics is what I read in Twitter": Weakly supervised models for extracting politicians' stances from twitter. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2966–2977. The COLING 2016 Organizing Committee.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR 2015, pages 1–15.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of Empirical Methods in Natural Language Processing*, EMNLP, pages 230–237.

Dongyun Liang, Weiran Xu, and Yinge Zhao. 2017. Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 43–47. Association for Computational Linguistics.

Bill Y. Lin, Frank Xu, Zhiyi Luo, and Kenny Zhu. 2017. Multi-channel bilstm-crf model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165. Association for Computational Linguistics.

Marina Litvak, Natalia Vanetik, Efi Levi, and Michael Roistacher. 2016. What's up on twitter? catch up with twist! In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 213–217. The COLING 2016 Organizing Committee.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '16, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Taro Miyazaki, Shin Toriumi, Yuka Takei, Ichiro Yamada, and Jun Goto. 2017. Extracting important tweets for news writers using recurrent neural network with attention mechanism and multi-task learning. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, PACLIC 31, pages 363–369. The National University (Phillippines).

Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. 2011. Safety information mining — what can nlp do in a disaster—. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, IJCNLP 2011, pages 965–973. Asian Federation of Natural Language Processing.

Symeon Papadopoulos, David Corney, and Luca Maria Aiello. 2014. Snow 2014 data challenge: Assessing the performance of news topic detection methods in social media. In *Proceedings of the SNOW 2014 Data Challenge*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011.

Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

Swen Ribeiro, Olivier Ferret, and Xavier Tannier. 2017. Unsupervised event clustering and aggregation from newswire and web articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 62–67. Association for Computational Linguistics.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 502–518.

Toshinori Sato. 2015. Neologism dictionary based on the language resources on the Web for Mecab. https://github.com/neologd/mecab-ipadic-neologd. Accessed: 2018-02-01.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of NIPS 2015 workshop on machine learning systems (LearningSys)*.

Svitlana Vakulenko, Lyndon Nixon, and Mihai Lupu. 2017. Character-based neural embeddings for tweet clustering. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 36–44. Association for Computational Linguistics.

Vladimir Vapnik and Aleksander Lerner. 1963. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. In *the 10th International Workshop on Semantic Evaluation*, volume abs/1606.05694 of *SemEval-2016*, pages 413–419.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL '17, pages 647–653. Association for Computational Linguistics.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, IJCNLP 2017, pages 253–263. Asian Federation of Natural Language Processing.

Deyu Zhou, Tianmeng Gao, and Yulan He. 2016. Jointly event extraction and visualization on twitter via probabilistic modelling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '16, pages 269–278. Association for Computational Linguistics.