International Journal of Innovative Computing, Information and Control
Volume **15**, Number **1**, February **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **189–198**

A SPOILER DETECTION METHOD FOR JAPANESE-WRITTEN REVIEWS OF STORIES

Atsushi Ueno, Yuu Kamoda and Tomohito Takubo

Graduate School of Engineering
Osaka City University
3-3-138, Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
{ ueno; takubo }@eng.osaka-cu.ac.jp; kamoda@kdel.info.eng.osaka-cu.ac.jp

Received May 2018; revised September 2018

Abstract. *Many users of online shopping sites consult product reviews before making a purchase. Some products, such as comics, novels, and movies, contain stories. Reviews of them sometimes contain spoilers, which often disappoint readers of the reviews. Some methods have been proposed for classifying between spoiler and non-spoiler sentences in English-written reviews. Their performance is greatly degraded when applying them to Japanese-written reviews. This paper proposes a method for classifying between spoiler and non-spoiler sentences in Japanese-written reviews of products with stories. This method performs two kinds of learning: word vector learning using fastText and classification learning using a Long Short-Term Memory network. By combining these two techniques, our method can cope well with Japanese-written reviews, which often include slang, new words, and subject-omitted sentences. The effectiveness of our proposed method is verified through experiments using product reviews on an online shopping marketplace, Rakuten Shopping Mall.*

Keywords: Spoiler detection, Product reviews, Vector representations of words, FastText, Long short-term memory

1. Introduction.

In recent years, users of online shopping sites are increasing. Many of the sites have a product review function, by which users can freely describe about products. Many users consult the reviews that other users have written before making a purchase. Some products, such as comics, novels, and movies, contain stories. Reviews of them sometimes contain *spoilers*, which reveal plot elements of the stories. Spoilers often disappoint users because they reduce thrills and enjoyment when reading or watching for the first time.

In order to cope with this problem, Iwai et al. [1-3] proposed methods for classifying between sentences with spoiling information (they are called *spoiler sentences*) and sentences without it (they are called *non-spoiler sentences*) in reviews of products with stories. They were targeting reviews written in English. We have applied their newer method proposed in [2,3] to reviews written in Japanese and found that the performance is greatly degraded (The details are explained in Section 5). The main cause for this degradation seems to be that it is difficult for their methods to cope with subject-omitted sentences which are common in Japanese-written reviews. The experiments in [1] showed that "I", "he", and "she" are included among important words that have especially high mutual information with classification between spoiler and non-spoiler sentences. "I" ranks first in all three categories investigated (comics, novels, and DVDs). "He" ranks second (comics), sixth (novels), and fourth (DVDs). "She" ranks fifth (comics) and seventh (novels). We have conducted a similar investigation into product reviews on an online shopping marketplace,

DOI: 10.24507/ijicic.15.01.189

Rakuten Shopping Mall, which are written in Japanese. No subjective pronouns are included in the top 10 list. The cause of this difference seems to be that they are often omitted in Japanese-written reviews. They are important in the classification in English-written reviews and not used much in Japanese-written reviews, which seems to be the reason why the performance decreases greatly. Their methods also have a problem that they cannot utilize "unknown words", which are not included in the vocabulary of the training dataset. Reviews of products with stories usually contain a lot of slang and new words, including the names of imaginary people, things, and places. In order to reduce unknown words, their methods need a huge number of sentences with labels of "Spoiler" or "Non-Spoiler" in the training dataset. Labeling must be done manually, requiring a lot of labor. In their experiments, the vocabulary of the training dataset is composed of words appearing in 450 reviews, but it may not be enough to reduce unknown words sufficiently.

This paper proposes a method for classifying between spoiler and non-spoiler sentences in Japanese-written reviews of products with stories. It can be used to hide only spoiler sentences in each review. In order to cope with the above problems, it performs two kinds of learning. In the first stage, it learns the vector representations of words (they are called *word vectors*) using *fastText* [4]. Words in reviews are represented by these vectors. In this stage, the method can use a huge number of sentences without labeling as "Spoiler" or "Non-Spoiler". Therefore, the authors expect that the method can cope well with reviews including a lot of slang and new words. In the second stage, each sentence in reviews is represented as a time series of word vectors, and our method learns the classification between spoiler and non-spoiler sentences by a *Long Short-Term Memory* (LSTM) network [5], which is a kind of *Recurrent Neural Networks* (RNNs) and one of the state-of-the-art models for learning from time series data. The authors expect that many subject-omitted sentences can be successfully classified by making use of their time series structure.

This paper is organized as follows. Section 2 describes related work. Section 3 explains the outline of an LSTM network used in our method. Section 4 describes the details of our proposed method. In Section 5, its effectiveness is verified through experiments using product reviews on an online shopping marketplace, Rakuten Shopping Mall. Section 6 concludes the paper and discusses the future work.

2. Related Work.

Let us assume that we want to detect spoilers in a comment about a story. There are two approaches to this task: one uses the main or synopsis text of the story, and the other uses neither of them. In the former approach, a classifier specialized for the story can be trained, and the classification accuracy is generally high. There are several previous studies on this approach. Guo and Ramakrishnan [6] proposed a method for detecting spoilers in comments about movies on an online database site, Internet Movie Database, utilizing synopsis texts about each movie posted on the site. Maeda et al. [7] showed that words related to serious spoilers of a novel are frequently used in the latter half of the story and showed the possibility of detecting spoilers based on words with such distributions. In implementing a method based on this approach, texts about stories (main or synopsis texts of stories) need to be prepared. In the case of detecting spoilers of novels, the main texts can be utilized while copyright issues are carefully managed. In the case of detecting spoilers of comics, movies, or television dramas, some synopsis texts need to be prepared. Although famous works already have a lot of such texts, synopsis texts about non-famous and/or new works need to be created manually. We have adopted the latter approach because preparing texts about thousands of stories requires a lot of labor.

There are also several previous studies on the latter approach. Golbeck [8] proposed a method for avoiding spoilers about television shows on Twitter by blocking tweets including the name of each show or the first or last names of any characters of the show. Our purpose is to classify sentences in a review into spoiler and non-spoiler ones. It is obvious that there are many spoiler sentences each of which has neither the title of the story nor the names of any characters. Boyd-Graber et al. [9] proposed a method for detecting spoiler sentences in wiki pages about television shows by using a Support Vector Machine (SVM) classifier. This method uses word unigram and bigram features and additional metadata features (genre, length, first air date, nation of origin, and episode number). They showed that using these metadata features improves performance. However, it is unclear what types of metadata features are effective and obtainable for reviews of products with stories, which this paper is targeting.

Iwai et al. [1] proposed a method for classifying between spoiler and non-spoiler sentences in English-written reviews of products with stories. First, this method extracts words that have high mutual information with classification between spoiler and non-spoiler sentences. Then, each sentence is represented by the *Bag-of-Words* (BoW) model using the extracted words, and the method learns the classification between spoiler and non-spoiler sentences by a classifier. They tested five common classifiers and showed that Naive Bayes and SVM classifiers are effective. They also showed the effectiveness of generalization of people's names, which replaces people's names in reviews with tags such as "⟨character⟩", "⟨author⟩", and "⟨actor⟩". Iwai et al. [2,3] proposed a spoiler detection method improved from the one in [1]. The method utilizes contextual information about sentences as well as information about words contained in each sentence. It uses the following two properties as the contextual information of each sentence: the position of the sentence in a review and the probabilities that each of the surrounding sentences is a spoiler. They showed that utilizing the contextual information improves the performance of detecting spoilers. However, as mentioned in Section 1, when applying this method to Japanese-written reviews of products with stories, the performance was greatly degraded. It is the purpose of this paper to propose a method to reduce this performance degradation.

Recommender systems that utilize online comments on the Internet have also been widely studied. For example, a system for recommending news articles [10] and systems for recommending tweets [11,12] based on users' activities on Twitter have been proposed. It is important for recommender systems to find items that attract a specific user's interest. It is good if the recommended items attract him with high probability. They do not have to find all the items that can attract him. Therefore, precision is more important than recall. On the other hand, it is important for spoiler detection methods, which are the subject of this paper, to find spoilers with high probability because a few spoilers can reveal an important plot of a story. Therefore, a spoiler classifier should have a high recall score for class "Spoiler". In this respect, spoiler detection methods differ from recommender systems. Moreover, recommender systems need to be personalized because each user gets interested in different items. Spoiler detection methods need not necessarily be personalized because there are many sentences that many users commonly feel spoiled. This paper proposes a method for detecting sentences that many users commonly feel spoiled without considering personalization.

3. LSTM with Forget Gates.

In classification learning, the authors use an LSTM network, which is a kind of RNNs and one of the state-of-the-art models for learning from time series data. Instead of ordinary neurons, LSTM networks use memory blocks (or LSTM modules), which contains one or more memory cells and some gating units that

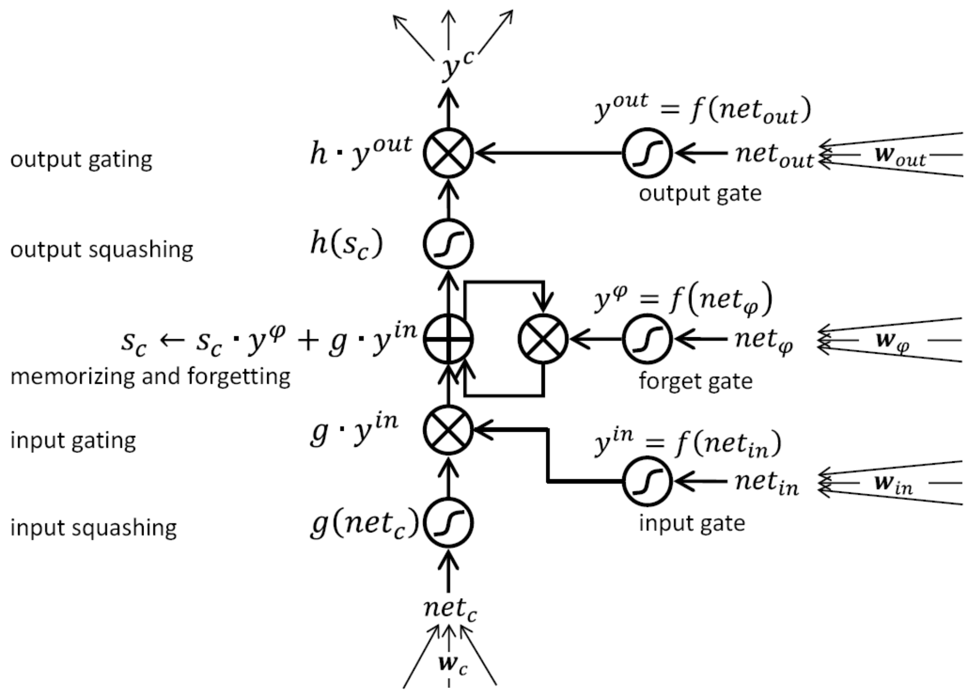

Figure 1. Memory block of LSTM with forget gates. (This figure is based on and modified from Figure 1 of [5].)

control signal flow. The authors use *LSTM with forget gates*, which was proposed by Gers et al. [5]. This section explains the outline.

Figure 1 shows a memory block with a single memory cell. There are inputs from the bottom to the block, and it outputs upward. In addition, there are three gates (input, output, and forget gates) on the right, and each gate receives the same inputs as the block receives from the bottom. Inputs are multiplied by the weights of the connections ($\boldsymbol{w}_c$, $\boldsymbol{w}_{in}$, $\boldsymbol{w}_{out}$, and $\boldsymbol{w}_{\varphi}$), and the weighted sums are net_c, net_{in}, net_{out}, and net_{φ}. They are used after squashed by sigmoid functions (f and g). If the block outputs the squashed weighted sum of the inputs to it ($g(net_c)$), it becomes exactly the same as an ordinary neuron. This block has a single memory cell, and s_c denotes its state. The input gate controls how much the current inputs get reflected in the current state. The forget gate controls how much the state at the previous time step gets reflected in the current state. Both the squashed weighted sum of the inputs and the previous state are processed by the gates and summed up to be the current state. The output gate controls how much the current state is output to the outside of the block. In this way, the state of the cell can be kept the same as the previous time step, and it can also forget its value completely and be set to the squashed weighted sum of the current inputs. It is possible to output the current state as it is, and it is also possible not to output at all. The function of the gates are decided by the weights of the connections, and it is possible to train them to perform appropriate control. Because memory cells can remember values over arbitrary time intervals, a network with LSTM modules can learn long-term dependency relationship in time series data.

4. **Proposed Method.** The authors have developed a method for classifying between spoiler and non-spoiler sentences in Japanese-written reviews of products with stories. In the method, each sentence in reviews is represented as a time series of word vectors, which are learned by fastText. Then, the word vectors are fed to an LSTM network in the order of the time series, and the sentence is determined to be a spoiler or non-spoiler one.

4.1. **Preprocessing.** First, all reviews are divided into sentences by the line feed code and the following 10 characters: “。”, “？”, “?”, “！”, “!”, “…”, “♪”, “✻”, “★”, and “☆”, which are used to show the end of a sentence in Japanese, sometimes with some feelings. Secondly, all sentences in reviews are divided into words, and all the words are classified into word classes, such as noun and verb. The morphological analyzer *MeCab*[1] with the neologism dictionary *mecab-ipadic-NEologd* v0.0.5 is used for this process. Thirdly, the method extracts nouns (including adjectival nouns), verbs, and adjectives in each sentence. Only the words in these three word classes are used in the classification process.[2] Finally, inflected words are returned to their base forms, and each sentence in the reviews is represented as a series of words in the three word classes in order of appearance.

4.2. **Learning of word vectors.** The authors use fastText, which provides a model for learning word vectors using a Neural Network (NN). This model is a variant of the *skip-gram* model [13]. It places word vectors close to each other if the words are often used in similar contexts. To put it plainly, words with similar meanings are converted into similar vectors.

In word vector learning, our method can use both spoiler and non-spoiler sentences as training data without distinction. Therefore, a huge number of sentences without labeling as “Spoiler” or “Non-Spoiler” can be used. In performing classification between spoiler and non-spoiler sentences, even if it encounters a word not in the vocabulary of the training dataset of the classification learning, if similar words are in the vocabulary, it can make use of this word via these similar words. The word can practically be regarded as a “known word”. Learning word vectors from a huge number of sentences can greatly reduce unknown words encountered in classification.

In the fastText model, each word is represented as a bag of character n-grams. This model can make use of subword information, and therefore it can convert rare words not in the vocabulary of the training dataset of the word vector learning into vectors. This also contributes to the reduction of unknown words encountered in classification. The authors expect that by sufficiently reducing unknown words, our method can cope well with reviews including a lot of slang and new words.

4.3. **Classification between spoiler and non-spoiler sentences.** In our method, each sentence in reviews is represented as a series of word vectors in order of appearance of the words. This representation has much more information than BoW representation, which is used in the methods of Iwai et al. [1-3]. Firstly, it includes the information about the similarity of words. Our method can make use of information of words with similar meanings. Secondly, it includes the information about the order of words in sentences. In classification learning, the authors use an LSTM network explained in Section 3, which is one of the state-of-the-art models for learning from time series data. By making use of this information-rich representation, the authors expect that many Japanese sentences whose subjects are omitted can be successfully classified between spoiler and non-spoiler sentences.

For this classification task, our method uses a three-layer RNN classifier. Figure 2 shows the structure of the network. The input layer has the same number of units as the dimension size of word vectors and receives one of them in the order of words in each sentence. In the experiments in this paper, the dimension size of word vectors is set to 100, which is the default value when using fastText.[3] The hidden layer is an LSTM

[1]MeCab ver. 0.996. http://taku910.github.io/mecab/.

[2]In a preliminary experiment, the authors tested another version of the method that uses all the words in a sentence, and its performance was inferior to that of the above proposed one.

[3]https://github.com/facebookresearch/fastText.

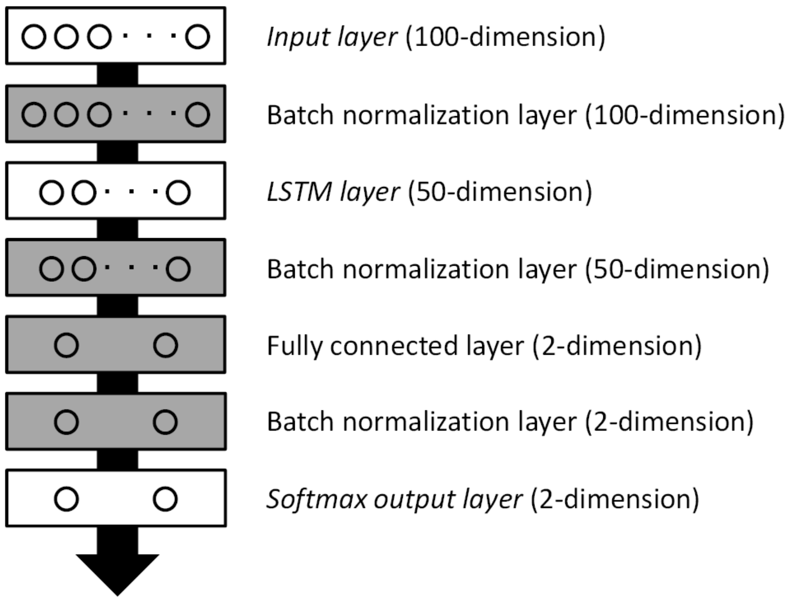
Input layer (100-dimension)
Batch normalization layer (100-dimension)
LSTM layer (50-dimension)
Batch normalization layer (50-dimension)
Fully connected layer (2-dimension)
Batch normalization layer (2-dimension)
Softmax output layer (2-dimension)

Figure 2. Structure of the LSTM network

layer, in which each module has recurrent connections from each other and a memory cell for remembering a value over arbitrary time intervals. By using these mechanisms, a network with LSTM modules can learn long-term dependency relationship in time series data. In the experiments in this paper, this layer has 50 modules. A word vector is fed to the network at a time step, and each sentence is fed over several time steps. In order to determine whether each sentence is a spoiler or not, the output of the hidden layer is propagated to the next output layer at the time the whole sentence has been fed. The output layer has two units: one indicates the probability that the input sentence is a spoiler and the other indicates the probability that it is a non-spoiler. The softmax function is used for calculating the probabilities. The network has full connections from the input to the hidden layer and the hidden to the output layer. Cross entropy is used as the loss function of the network. The *dropout* technique is applied by stochastically “dropping out” some input and hidden units during learning in order to avoid overfitting to the training dataset. The dropout rates for the input and hidden layers are set to 0.2 and 0.5, respectively. The *batch normalization* technique is used between the input and hidden layers and between the hidden and output layers in order to speed up learning. In the experiments in this paper, the batch size is set to 64. The network is implemented by Keras,[4] which is a high-level neural networks API. The hyperparameters of the network other than mentioned above are set to the default values when using Keras.

5. **Experiments and Discussion.**

In this section, the effectiveness of our proposed method is verified through experiments using product reviews on an online shopping marketplace, Rakuten Shopping Mall (Rakuten Ichiba).

5.1. **Datasets.**

Rakuten Ichiba is one of the largest e-commerce sites in Japan. Rakuten, Inc. provides a dataset that includes product data for over 150 million items in Rakuten Ichiba as well as over 64 million user reviews about these items. It is called the Rakuten dataset.[5] Product reviews in this dataset are used for word vector learning, classification learning, and classification testing.

[4]https://keras.io/.
[5]https://www.nii.ac.jp/dsc/idr/en/rakuten/rakuten.html.

The dataset for classification learning/testing is composed as follows. Firstly, the authors extract all product items that are included in the categories whose names include the word "コミック" or "漫画" ("comic" or "manga" in English). Secondly, items with fewer than five reviews are excluded from the extracted items. Thirdly, 100 items are randomly selected from the remaining items. Finally, five reviews are randomly selected for each selected item. For all the sentences in the obtained 500 reviews, three people in our laboratory have manually performed labeling as "Spoiler" or "Non-Spoiler". A sentence has been judged as a spoiler if more than two people have labeled it as "Spoiler". Otherwise, it has been judged as a non-spoiler sentence. The authors have obtained 182 spoiler and 1,167 non-spoiler sentences from the reviews.

The training dataset for word vector learning consists of all the sentences in all the product reviews other than that used for the above dataset for classification learning/testing. It contains over 200 million (202,113,226) sentences. This number is overwhelming compared with the size of the dataset for classification learning/testing (1,349 sentences). This huge number of sentences can be used since they do not require manual labeling as "Spoiler" or "Non-Spoiler".

5.2. **Experimental settings.** In word vector learning, our method uses the *negative sampling* algorithm [13] in order to approximate the softmax function. The number of negatives sampled is set to five, which is the default value when using this algorithm in fastText.[6] The minimum and maximum lengths of character n-grams are set to three and six, respectively. These values are the same as those shown to be reasonable in the original paper of fastText [4]. The minimal number of word occurrences = 5; and the initial learning rate = 0.05. All the other hyperparameters for word vector learning are set to the default values when using fastText.

In classification experiments, 10-fold cross validation technique is used. First, the dataset is divided into 10 subsets in a manner where all sentences of reviews for one particular product are included in the same subset. Then, eight subsets (the training dataset) are used for training a classifier, another subset (the validation dataset) is used for deciding when learning should be stopped, and the other subset (the test dataset) is used for testing the classifier. This process is repeated in 10 rounds while rotating the subsets. The performance is measured by averaging the results of the tests in the 10 rounds. Because the training dataset is imbalanced, in which there are several times as many non-spoiler as spoiler sentences, the authors use the cross entropy loss function weighted according to the number of data in each class.

5.3. **Experiment 1: Comparison between optimizers.** In this section, seven optimizers provided by Keras are compared in order to decide which is the best one to use in our method. They are Stochastic Gradient Descent (SGD), Adagrad, RMSprop, Adadelta, Adam, Adamax, and Nesterov Adam (Nadam) optimizers. They are all based on gradient descent. Figure 3 shows their family tree. SGD is the ancestor of all the others. In the standard gradient descent algorithm, each gradient is calculated using all training data, whereas SGD updates the weights of the connections (parameters) gradually and repeatedly using gradients calculated for each training data. Adagrad is an algorithm developed based on SGD, and it adjusts the learning rates automatically so that the learning rate of a parameter decreases if it has been updated significantly in the past. RMSprop is an algorithm developed based on Adagrad, and instead of evenly considering all past updates, it calculates the learning rates with priority given to the latest updates. Adadelta is also an algorithm developed based on Adagrad and also calculates

[6]https://github.com/facebookresearch/fastText.

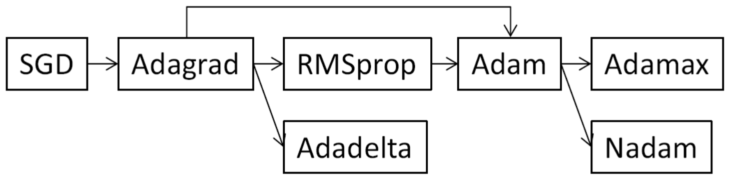

FIGURE 3. Family tree of optimizers

the learning rates with priority given to the latest updates. Furthermore, in Adadelta, the initial value of the learning rates does not need to be tuned because the units of updates are equal to those of the parameters. Adam is an algorithm developed based on Adagrad and RMSprop, in which both adjustment of the learning rates and momentum in updating parameters are incorporated. Adamax is a simpler variant of Adam. Nadam is an algorithm incorporating Nesterov momentum into Adam. All the hyperparameters of these optimizers are set to the default values when using Keras.[7] The performances are measured by precision, recall, and F-measure for class "Spoiler".

Table 1 shows the results of this experiment. The column "#Epochs" shows the number of epochs until the completion of learning. These numbers have been decided so that learning stops around the peak before the learning curve drops sharply. The number for SGD optimizer is particularly large because the learning with it is much slower than those with the other optimizers. The best is SGD optimizer in terms of precision, Nadam optimizer in terms of recall, and Adadelta optimizer in terms of F-measure. Because a few spoilers can reveal an important plot of a story, a spoiler classifier should have a high recall score for class "Spoiler". However, an excessively low precision score increases non-spoiler sentences that are filtered out unnecessarily and reduces information that readers of reviews can get. Therefore, Adadelta optimizer is adopted to use in our method, which has the highest F-measure score and the second highest recall and precision scores.

TABLE 1. Comparison between optimizers

Optimizer	#Epochs	Precision	Recall	F-measure
SGD	3000	**0.494**	0.484	0.489
Adagrad	200	0.461	0.643	0.537
RMSprop	100	0.459	0.577	0.511
Adadelta	200	0.467	0.670	**0.551**
Adam	100	0.467	0.582	0.518
Adamax	100	0.441	0.632	0.519
Nadam	25	0.417	**0.720**	0.528

5.4. **Experiment 2: Comparison with the previous method.** In this section, our method is compared to the newer method of Iwai et al. [2,3]. We have implemented their method in order to apply it to reviews written in Japanese with the same experimental settings as used for our method. However, the process of generalization of people's names in their method has not been implemented because it is difficult in Japanese text to determine if a word is a part of a person's name (including nickname) or not.

Table 2 shows the results of this experiment. The first row of the table shows the results of their method when applied to English-written reviews of comics, which are shown in Table 7 of [2]. The second and third rows show the results of their and our

[7]It is recommended to leave the hyperparameters at their default values when using Adagrad, Adadelta, Nadam, and RMSprop (except the learning rate) on the official site: https://keras.io/optimizers/.

Table 2. Comparison with the previous method

	Precision	Recall	F-measure
Method of Iwai et al. [2,3] (English. In the original paper)	0.80	0.77	0.78
Method of Iwai et al. [2,3] (Japanese)	0.39	0.40	0.39
Proposed method (Japanese)	**0.47**	**0.67**	**0.55**

methods respectively when applied to Japanese-written reviews with the experimental settings in this section. As shown in this table, the performance of their method is greatly degraded when applied to Japanese-written reviews. On the other hand, our method can appreciably reduce this performance degradation. It copes with subject-omitted sentences by utilizing the LSTM network, which can make use of the time series structure of sentences. It also copes with sentences containing a lot of slang and new words by making use of information of words with similar meanings that are included in an overwhelmingly huge number of sentences in product reviews. The high recall score for class "Spoiler" of our method is also favorable since spoilers should be concealed as much as possible.

5.5. **Discussion.** In this section, the authors investigate cases in which our method misclassified spoiler sentences as "Non-spoiler" and consider countermeasures because reading a spoiler can be a misery. In Experiment 2, our method misclassified 60 out of 182 spoiler sentences as "Non-spoiler". Among them, there were some sentences beginning with "The content is . . . " ("内容は..." in Japanese). The word "content" ("内容" in Japanese) is not useful for the classification because it is also included in a lot of non-spoiler sentences (34 non-spoiler and 7 spoiler sentences). The authors have found that many of these misclassified sentences are short, and therefore it is difficult to classify them correctly based only on themselves. The authors expect that a spoiler detection method that takes account of the preceding and following sentences may be effective.

Our method also misclassified sentences with slang words such as "stick together" ("くっつく" in Japanese). Although this word is commonly used when something and something stick together, it is mainly used when someone and someone start dating in Japanese-written reviews of comics. The main reason why our method cannot cope with this slang word with such an important meaning seems to be that the word vector learning has been performed on the training dataset that consists of reviews of all products. The authors expect that such misclassification can be reduced by using a word vector representation suitable for the analysis of reviews of stories.

In the dataset for classification learning/testing, there are some sentences each of which is labeled as "Spoiler" and seems to be a non-spoiler sentence in the authors' opinions. The judgement of whether each sentence is a spoiler or non-spoiler one differs from person to person. In order to solve this problem, it is effective to create a classifier specialized for each user based on one created by the proposed method.

6. **Conclusions.** This paper has proposed a method for classifying between spoiler and non-spoiler sentences in Japanese-written reviews of products with stories. There are two key problems to be addressed. Firstly, many sentences in reviews are informal and include a lot of slang and new words. Secondly, subjects are often omitted in Japanese-written reviews. In order to cope with these problems, our method performs two kinds of learning. In the first stage, it learns the vector representations of words using fastText. Words in reviews are represented by these vectors. In this stage, our method can use an overwhelmingly huge number of sentences without labeling as "Spoiler" or "Non-Spoiler".

Therefore, it can cope well with reviews including a lot of slang and new words. In the second stage, each sentence in reviews is represented as a time series of word vectors, and our method learns the classification between spoiler and non-spoiler sentences by an LSTM network, which is one of the state-of-the-art models for learning from time series data. Therefore, our method can successfully classify many Japanese sentences whose subjects are omitted by making use of their time series structure. In the experiments using product reviews on an online shopping marketplace, Rakuten Shopping Mall, our method showed much better performance than one of the best previous methods when both were applied to Japanese-written reviews. The authors might go on to improve the method so that it can take account of surrounding sentences, develop a word vector representation suitable for the analysis of reviews of stories, and develop a classifier specialized for each user.

Acknowledgment. In this work, the authors use the Rakuten dataset which Rakuten, Inc. provides for research purpose with the cooperation of the National Institute of Informatics (NII). We would like to thank Rakuten, Inc. and NII.

REFERENCES

[1] H. Iwai, K. Ikeda, Y. Hijikata and S. Nishida, Proposal of a plot classification method for user reviews, *IEICE Trans. Inf. & Syst. (Japanese Edition)*, vol.J96-D, no.5, pp.1222-1234, 2013 (in Japanese).

[2] H. Iwai, Y. Hijikata and S. Nishida, Plot classification method using contextual coherence for review sentences, *IPSJ Trans. Databases*, vol.7, no.2, pp.11-23, 2014 (in Japanese).

[3] Y. Hijikata, H. Iwai and S. Nishida, Context-based plot detection from online review comments for preventing spoilers, *Proc. of 2016 IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pp.57-65, 2016.

[4] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, *Trans. Assoc. for Computational Linguistics*, vol.5, pp.135-146, 2017.

[5] F. A. Gers, J. Schmidhuber and F. Cummins, Learning to forget: Continual prediction with LSTM, *Neural Computation*, vol.12, no.10, pp.2451-2471, 2000.

[6] S. Guo and N. Ramakrishnan, Finding the storyteller: Automatic spoiler tagging using linguistic cues, *Proc. of the 23rd Int. Conf. on Computational Linguistics*, pp.412-420, 2010.

[7] K. Maeda, Y. Hijikata and S. Nakamura, A basic study on spoiler detection from review comments using story documents, *Proc. of 2016 IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pp.572-577, 2016.

[8] J. Golbeck, The Twitter mute button: A web filtering challenge, *Proc. of Int. Conf. on Human Factors in Computing Systems*, pp.2755-2758, 2012.

[9] J. Boyd-Graber, K. Glasgow and J. S. Zajac, Spoiler alert: Machine learning approaches to detect social media posts with revelatory information, *Proc. of American Society for Information Science and Technology*, vol.50, no.1, pp.1-9, 2013.

[10] F. Abel, Q. Gao, G. J. Houben and K. Tao, Analyzing user modeling on Twitter for personalized news recommendations, *Proc. of the 19th Int. Conf. on User Modeling, Adaption, and Personalization*, pp.1-12, 2011.

[11] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao and Y. Yu, Collaborative personalized tweet recommendation, *Proc. of the 35th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp.661-670, 2012.

[12] R. Harakawa, D. Takehara, T. Ogawa and M. Haseyama, Sentiment-aware personalized tweet recommendation through multimodal FFM, *Multimedia Tools and Applications*, vol.77, no.14, pp.18741-18759, 2018.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems 26*, pp.3111-3119, 2013.