RICT at the NTCIR-14 QALab-PoliInfo Task

Jiawei Yong[1], Shintaro Kawamura[1], Katsumi Kanasaki[1], Shoichi Naitoh[1], and Kiyohiko Shinomiya[1]

Ricoh Company, Ltd.
{kai.yuu,shintaro.kawamura,katsumi.kanasaki,shohichi.naitoh,kshino}@jp.ricoh.com

Abstract. Our RICT team tackled segmentation and classification subtasks in NTCIR-14 QA Lab-PoliInfo. As our technical characteristic in segmentation task, we regard segments as retrieval objects, and utilize cue-phase-based methods including a nobel semi-supervised learning method to detect segment boundary. Here we have presented 5 methods for formal run. As to classification task, we train our supervised learning model by all utterances without topic effects and utilize outlier detection technologies to alleviate the imbalance training data problem. Here we have submitted 7 runs for formal run. Since evaluation results show that our approach achieves higher scores than average, the main contribution for classification we made is providing a feasible system to deal with a small number of imbalance training data problem especially in the regional assembly minutes. Meanwhile, we also made contribution for segmentation to grasp distinguishing features of regional assembly minutes by our efficient method.

Team Name. RICT

Subtasks. Segmentation task (Japanese), Classification task (Japanese)

Keywords: Segmentation · Classification · Semi-supervised method · LSTM · Isolated forest

1 Introduction

Our RICT team participated in the segmentation and classification subtasks of the NTCIR-14 QA Lab-PoliInfo task [9]. Each section in this paper has two subsections, one for segmentation and the other for classification, since there is no direct relationship between the two subtasks.

1.1 Segmentation

The PoliInfo segmentation subtask is the task that is going to find a sequence of assembly utterances that corresponds to the given summary. The segmentation of utterances is not mandatory to find the correspondence. However, if a part of utterances on a topic is chosen, the contents can be misunderstood. This is the reason why we consider the task consists of the segmentation step and the search step. The segmentation step splits the whole utterances into segments, each of which deals with a topic. The search step finds a segment or a sequence of segments that corresponds to the summary.

Both the text segmentation and the text search have been widely studied.

TextTiling [8] and C99 [4] are well known as unsupervised text segmentation methods. They rely on lexical cohesion through a segment. Cue phrases, which often appear in assembly utterances, are not considered. Eisenstein [6] proposes another unsupervised method, where different segments are assumed to be derived from different language models, and texts around segment boundaries are derived from another language model that reflects cue phrases. Although this approach is interesting, because cue phrases are considered in an unsupervised method, the effects of cue phrases are small in the experiment in the paper.

Cue phrases play important roles in supervised segmentation [3]. Supervised approaches that employ neural networks also appear recently [10, 2]. A large dataset like Wikipedia is used to train a model. It is difficult to directly apply such a trained model to assemly minutes, because utterances in an assembly have unique characteristics. This means a labeling job is required to prepare training data.

The segment search task is similar to the document search task. Widely used TF-IDF scheme can be also applied to our case. The length of a segment can be, however, chosen in the search. In contrast the length of each document is fixed. Although Okapi BM25 [14] prioritizes shorter documents, the formula has more than one heuristic parameters.

Semantic textual similarity has been recently studied [1]. In our case, however, the words used in utterances are also used in the summary. There is no strong reason to consider semantic similarity.

1.2 Classification

The classification subtask of NTCIR-14 QALab-PoliInfo aims to pick up fact-checkable speech utterances from Tokyo Metropolitan Assembly minutes. For that purpose, organizers prepare 3 labels for this subtask. Relevance, the 1st label, is used to clarify whether the speech utterance is relevant to present topics. Fact-checkability, the 2nd label, is used to tell whether it is possible to verify the fact in utterance or not. Stance, the 3rd label, is used to estimate the potential stance to the present topic including Support, Against and Other. When all 3 labels above are clarified, we could classify the speech utterances into Fact-checkable Support, Fact-checkable Against or Other categories under the rules provided by organizer.

Relevance and Fact-checkability labels can be regarded as 2-class text classification issues since their values can only be 0 (irrelevant and uncheckable) or 1 (relevant and checkable). As to Stance label, since its values can be 0 (Other), 1 (Support), or 2 (Against), it can be regarded as multi-class text classification problem. Moreover, in consideration of the training data's imbalance distribution, we summarize the useful related works as below.

There are some valuable researches (especially learning method) focusing on text classification have been carried out in recent years. For text classification, Tagami [16] states that supervised learning method RNN LSTM shows best performance if there are adequate high-quality dataset for training. On the other hand, traditional machine learning method SVM [17] shows enormous potential within small-scale training data. Different from assembly minutes this time, they extract the features with an eye to social network tweets. We believe the politician's speech has unique features which can be distinguished with social media comments.

To alleviate the bad influence of imbalance distribution problem, we also consider the minority class as outliers, thus leading to an outlier detection problem. One-class SVM [5] and Isolation Forest [11] are well known as good solutions for this problem. Due to our own experiments, the Isolation Forest shows the higher performance compared with One-class SVM. What's more, the cosine similarity [12] is often used as an addition to make results better.

2 Methods

2.1 Segmentation

The segmentation subtask is divided into two steps, the segmentation itself and the segment search. The segmentation step receives a set of assembly minutes, and it splits the utterances into a series of segments. The search step receives a summary sentence together with a date and a speaker of the assembly, and it finds a segment or a sequence of segments that corresponds to the summary.

In our dry run, TextTiling in the NLTK package was used for the segmentation step, and the Amazon Elasticsearch Service was used for the search step.

Although TextTiling does not depend on cue phrases in texts, assembly utterances contain many cue phrases, which humans often rely on to recognize the boundaries of topics. In the formal run, we compare some methods that use cue phrases to find the segment boundary.

Regarding the search step, we found that the given summary sometimes corresponds to a sequence of more than one segments. In the dry run, contiguous segments were combined when all of them have relevance scores that are higher than a threshold. It is, however, difficult to fix a specific value for the threshold. In the formal run, we use our own logic without relying on off-the-shelf search engines.

Dataset preparation The training dataset provided by the task organizer is not optimum to train the segmentation step. We, therefore, prepare another dataset, which is annotated by ourselves.

Utterances are split into segments so that the segments cover all the utterances of each speaker. The dataset consists of two parts, which are based on the assembly minutes of different days. The first one, which is used as a training dataset, splits 4804 utterances into 995 segments. The second one, which is used as a development dataset, splits 3438 utterances into 683 segments.

Although the utterances in a segment are grouped according to topics, each segment often starts with a cue phrase such as "次に" (next), and ends with another cue phrase such as "見解を求めます。" (the position is requested). The annotator heavily relies on the cue phrases for tagging.

Cue-phrase-based segmentation In the formal run, cue phrases, which appear in the first or the last utterance of each topical segment, are used for the segmentation step. The problem is modeled as a classification problem, which determines whether each utterance starts a segment or not.

The following methods for the segmentation are compared. The numbers correspond to the identifiers of the runs submitted in the formal run. The number 6, however, has been added after the formal run.

1. Rule-based
   During the annotation, we found many cue phrases. The regular expression used for the first utterance and the one for the last utterance are shown in Table 1. An utterance is classified as the first one in a segment, if and only if the text of the utterance matches the opening pattern, or the text of the previous utterance matches the closing pattern but not the opening pattern.

Table 1. The regular expressions shown here are used to find cue phrases.

	regular expression
opening pattern	^まず|^最初に|^初めに|^次に|^次いで|^最後に|^終わりに
	|^[一二三四五六七八九十]+点目
	|^[^、]+についてで(す|あります|ございます)(が|けれど)
	|^終わり(ま|で)す。|^以上で|^ありがとうございま
	|他の質問に(ついて|つきまして)は
closing pattern	伺い[^、]*ます。|お尋ね[^、]*します。|お答えください。
	|(見解|所見|答弁)を求め[^、]*ます。
	|(いかがで|どうで)(しょうか|すか)。
	|.+質問を(終わります|終了します)。

2. Support vector machine (SVM)
   An SVM classifier is learned from the training dataset we prepared.
   Each utterance is first broken into words by MeCab with IPAdic. The lemmas of 10 words at the beginning of the utterance and 10 words at the end of the utterance are extracted and converted into a bag-of-words vector. The BoW vectors are then compressed into 100-dimensional vectors with the latent semantic indexing (LSI). For each utterance, the vector of the utterance and the one of the previous utterance are concatenated. The 200-dimesional vectors are used as feature vectors for the SVM classifier.

3. Semi-supervised learning
   Instead of using the training dataset that contains a gold standard, the method is going to learn from the original minutes data. The data contain 84905 utterances.
   The first utterance of a speaker is also the first utterance of a segment. We can construct a classifier from features of such utterances. Applying the classifier to the whole minutes, we can extract candidates of the first utterances of segments. The previous utterance of each candidate is a candidate of the last utterance of a segment. We can then construct another classifier from features of the previous utterances. Applying the classifier, we can extract candidates of the last utterances of segments. The next utterance of each candidate is a candidate of the first utterance of a segment. We can gradually improve the results in this way. The iterative process is inspired from the bootstrapping approach for named entity recognition [13].
   In our experiment, the BoW vectors also used for the SVM method are compressed into 200-dimensional vectors in this time. Two classifiers for the first utterances and last utterances are constructed in each iteration. Positive samples are first collected from the boundaries among speakers, and negative samples are randomly chosen around positive samples. The logistic regression is used for the classifier, because we would like to tune the balance between recall and precision. When the estimated probability exceeds a threshold value, a sample from all the utterances is considered to be positive. For each boundary between two utterances, if it is the boundary among speakers, or the utterance just after the boundary is classified as a first utterance, or the utterance just before the boundary is classified as a last utterance, the boundary is considered as a boundary between two segments in the next iteration.
4. No segmentation
   No segmentation is used for the third submitted run. This means that each utterance constitutes a segment.
5. Long short-time memory (LSTM)
   An LSTM classifier is learned from the training dataset we prepared.
   For each utterance, 10 words at the beginning of the previous utterance, 10 words at the end of the previous utterance, 10 words at the beginning of the utterance, and 10 words at the end of the utterance are collected. The word embeddings of the 40 words are concatenated and given to a uni-directional LSTM network. The 100-dimensional Word2Vec model pretrained by Tohoku university [15] is used.
6. Hierarchical attention network (HAN) [18]
   This run was executed after the formal run, and the result is not submitted. A HAN classifier is learned from the training dataset we prepared.
   For each utterance, 10 words at the beginning of the previous utterance, 10 words at the end of the previous utterance, 10 words at the beginning of the utterance, and 10 words at the end of the utterance are collected. The surface forms of words are used instead of lemmas in this experiment. The sequence of 10 words are given to the word-level network, which consists

of an embedding layer, a bi-directional GRU (gated recurrent unit) layer, and an attention layer. The embedding layer here is trainable and randomly initialized. The 4 results of the word-level networks are given to the sentence-level network, which consists of a bi-directional GRU and an attention.

Segment search This step finds a segment or a sequence of contiguous segments that corresponds to each sample of a dataset. The sample contains a date, a topic, and a subtopic along with two speakers and two summary sentences. A speaker and a summary sentence correspond to a question, and another pair corresponds to the answer to the question.

The date and the speaker are first used to find a sequence of segments. There is little difficulty to match the speaker, because the name of the speaker is recorded in the assembly minutes, but the position of the speaker is given in the summary dataset. Fortunately, remarks for speaker changes are also recorded in the minutes. We can use the remarks to map the speaker, because each remark includes both the position and the name of a speaker.

The same speaker often answers to more than one questioners, and the answers sometimes contain similar contents. In order to distinguish the answers, we choose the sequence of answer segments that follows the question segments found.

We then use the summary sentence together with the subtopic to identify corresponding segments. Hereinafter, a summary sentence together with a subtopic is called a summary. Characters in the summary are first converted into full-width characters, and digits are converted into Chinese characters to match with texts in the minutes. The summary is then broken into words by MeCab with IPAdic.

We construct a simplified probabilistic model to find the sequence of contiguous segments. Only the set of words in the summary is used, and the order of the words is ignored. Let us assume words $t_i (i = 1, \ldots, k)$ in the summary appear in $df(t_i)$ utterances among all the N utterances. Provided words independently appear in utterances, the provability that all the words appear in a sequence of n contiguous utterances is approximately $P = \prod_{i=1}^{k} n \frac{df(t_i)}{N}$. If the function idf is defined as $idf(t_i) = \log(\frac{N}{df(t_i)})$, $\log(\frac{1}{P}) = \sum_{i=1}^{k} idf(t_i) - \lambda k \log(n)$, where the weight parameter $\lambda = 1$. We search the sequence of contiguous segments that maximizes the value of the idf-based equation.

In fact, words do not independently appear in utterances. We experiment by changing the weight λ.

If multiple occurrences of a word in a segment are separately counted, the equation is modified as $\sum_{i=1}^{k} tf(t_i) idf(t_i) - \lambda(\sum_{i=1}^{k} tf(t_i)) \log(n)$, where $tf(t_i)$ is the number of occurrences of the word t_i. Let us call this the tf-idf-based equation.

Parameter tuning The hyperparameters of the segmentation step are tuned by observing the results on the development dataset we prepared. The most sensitive parameter is the threshold of the probability in the semi-supervised segmentation, which will be discussed in Section 4.1. The dropout technique is used for the methods that use neural networks to avoid overfitting.

The search step has a tunable parameter λ. The parameter is tuned so that a high F-measure is accomplished on the training dataset provided by the task organizer. The portion of data that corresponds to the training dataset we prepared is excluded, because the learning processes of some methods depend on the portion.

2.2 Classification

As mentioned in previous section, there are some similarities and differences among Relevance, Fact-checkability, and Stance text-classification. Therefore, we believe some common methods can be useful for several label judgements at the same time. Here we divide Classification into 3 phases where we put some ideas in. Since we obtain a large amount of training data from organizers, we plan to adopt several learning methods on prepared training data to achieve these classification goals.

Data preparation Phase The raw training data provided by organizers are labeled by at least 3 workers. Unfortunately, the low kappa coefficients among workers show that there are various understandings about labeling thus leading to a doubt of label correctness. Hence, we shall determine which part should be utilized as our real training data by leveraging the quantity and quality as follows.

1. Unanimous training data
   Unanimous training data prefers to quality rather than quantity. Here we only pick up the data with same labels from all workers. For example, if 3 workers label the same utterance as 0 in Fact-checkability which means they all think it is uncheckable, the labeled utterance would be part of unanimous training data. After this process, we have picked up about 4710 utterances from 31808 raw dataset.
2. Majority training data
   In contrast to unanimous training data, majority training data places a higher value on quantity since high performance of supervised learning method often comes from data with large scale. Here we pick up the data with same labels from majority along with sacrificing a part of labeling accuracy. For example, if at least 2 of 3 or 3 of 5 workers label the same utterance as 1 in Relevance, which means that the majority think it is relevant to present topic, the labeled utterance would be selected as majority training data. After this process, we have picked up around 10291 utterances from 31808 raw dataset.

Preprocessing Phase As we have mentioned in Section 1.2, there are several learning methods proving a high value in text-classification. However, there is a gap between existing methods and present specific issues since we need let system know which features should be learned.

1. morphological analysis
   Here we utilize open source Japanese morphological analysis engine MeCab combined with NEologd dictionary to extract words and the part of speech of them.
2. feature word filtering
   There is no need to apply all extracted words as features since it can easily cause underfitting problem. We shall pay more attention on the explicit feature words. For example, the explicit feature words for Fact-checkability are shown in Table 2.

Table 2. Explicit feature words for Fact-checkability.

Part of Speech	Examples
Proper Noun	大阪,鈴木
Number Noun	2012,4回
Measure Noun	個,つ,本,冊
Time Adverb	平日,以来,いつか,曜日
Independent Verb	投げる,行う
Accessory Word	て,つつ,および,ので
Dependent Verb	しまう,ちゃう,願う
Specific Punctuation	「,」,（,）
Specific Noun	報告,提案,状況
Pattern	**Examples**
Past Tense	た,ました,でした
Present Continuous Tense	て/でおり,て/であります,て/でございます
Fixed Expression	たところ,とは

3. word representation
   There are several word representation methods such as word2vec, GloVe already applied in text-classification. Since TOHOKU university has provided pre-trained word2vec [15] model on the internet, here we adopt it to represent each feature words as a 100-dimension vector for learning phase.

Learning Phase Learning methods can be roughly divided into supervised learning and unsupervised learning methods. Although semi-supervised learning methods become hot spot research recently, we only utilize the supervised and unsupervised learning methods as follows for classification subtask.

1. Support vector machine (SVM)
   SVM is a classical machine learning method which has shown higher performance than other similar methods in most circumstances. Since SVM ignores the order of words in utterance, we just take the average of word2vec vectors for one sentence as SVM input. Moreover, since balanced training data for SVM classifier results in a better prediction by our experiments, here we constrain the positive and negative samples as 1:1.
2. Long short-time memory (LSTM)
   As mentioned in Section 1.2, LSTM classifier shows better performance if there are adequate high-quality data for training. One noteworthy part of LSTM is that all input utterances shall be in the same length (word count) for Keras [7]. The preprocessed data would be learned through the structure of LSTM model shown in Fig. 1 as below.

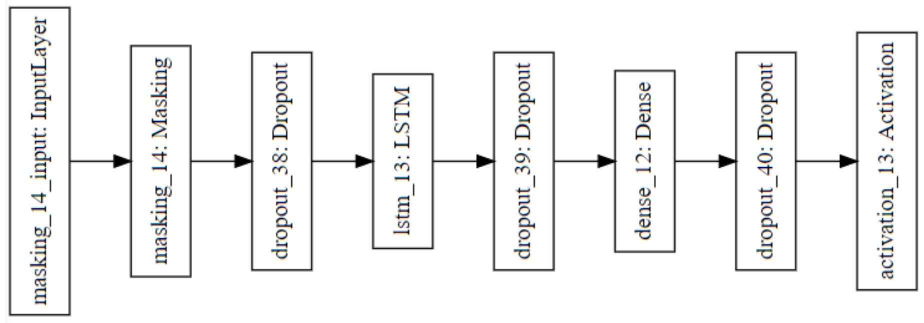


Fig. 1. The structure of LSTM model from iPython which has several dropout layers to alleviate overfitting problem.

3. Isolated forest
   Imbalance problem causes several difficulties such as over-sampling and under-sampling related to learning. Isolation Forest is an extension of the decision tree, which detects abnormal values with the depth. Specifically, it is based on the following two assumptions. Normal values are large in quantity, and all are similar (dense, difficult to separate) Abnormal values are small in quantity, and different from each other (sparse, easy to separate).
   In the implementation, we have constructed a tree by random feature selection, and then divided it into several sub trees. The average value of "shallowness" is regarded as the final abnormal score (threshold). As shown in Fig. 2, stance judgment (3 value classification) has applied Isolation Forest to determine whether data is in range of label 0, label 0 & 1, or label 0 & 2 (label 0 is majority).

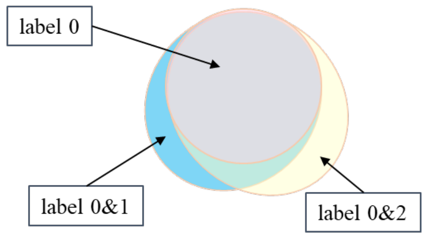


Fig. 2. Three ranges of normal values in Isolation Forest are used for stance judgement.

Since we achieve our text classification tasks by going through above 3 phases, here is a summary about our labeling methods for each classification task. 8 methods used in the submitted runs are listed in Table 3. Note that all methods are all through the same preprocessing, here we only list the combinations of data preparation and learning classifier for each method.

Table 3. The combinations of submitted methods of runs.

Labeling method	Data preparation	Learning classifier
1.Relevance, Stance (1)	Majority training data	Isolated forest(1)
2.Relevance, Stance (2)	One or more positive examples	Isolated forest(2)
3.Relevance, Stance (3)	Majority training data	Isolated forest+ Cosine similarity
4.Relevance, Stance (4)	Majority training data	Cosine similarity
5.Stance (5)	Unanimous training data	LSTM
6.Fact-Checkability (1)	Unanimous training data	LSTM
7.Fact-Checkability (2)	Unanimous training data	SVM
8.Fact-Checkability (3)	Majority training data	LSTM

Methods 1/2 are applying Isolation Forest as it is to Relevance and Stance classification tuned by different parameters. Since there are still many misjudged cases, not only the determination of the normal /abnormal range, but also the relationships with each label center of gravity have been considered. Methods 3/4 are utilizing the conjunction of Isolation Forest and Cosine similarity for Relevance and Stance classification. This is because even if it is judged as majority in Isolation Forest, labels can be corrected using the similarity with the center of gravity of other minority labels to further improve the accuracy. Method 5 is utilizing LSTM model on unanimous training data for Stance classification. This method pays more attention to the majority category than above methods. Methods 6/8 are utilizing LSTM model on unanimous training data and majority training data respectively. Here we just hope to confirm which of quality and quantity has more impact on accuracy. Method 7 is also based on unanimous training data which is trained by SVM model. Here we hope to verify if the SVM model fits data with a small scale.

3 Results

3.1 Segmentation

The results of the segmentation step are summarized in Table 4. The segmentation methods are applied to the development dataset we prepared. Recall and precision are based on the classification problem that labels the first utterance of a segment as positive. Pk is a metric widely used for segmentation tasks.

Table 4. The performance of the segmentation step. The mean values of 5 runs are shown. Regarding Pk, a smaller value is better.

method	recall	precision	Pk
rule-based	0.977	0.928	0.040
SVM	0.943	0.846	0.085
semi-supervised	0.812	0.798	0.119
LSTM	0.818	0.858	0.141
HAN	0.970	0.931	0.043

Two sets of results for the PoliInfo segmentation subtask are shown in Table 5 and Table 6. Table 5 shows the results when the steps are applied to the training dataset provided by the task organizer. The portion of data that corresponds to the training dataset we prepared is excluded from the dataset. The remaining data include 298 questions and 298 answers. Table 6 shows the results when the steps are applied to the test dataset with the gold standard provided by the task organizer. The test data include 83 questions and 83 answers.

Table 5. The performance of the methods when applied to the training dataset. The mean values of 5 runs are shown.

	question			answer		
segmentation method	recall	precision	F1	recall	precision	F1
rule-based	0.921	0.945	0.933	0.969	0.969	0.969
SVM	0.900	0.935	0.917	0.943	0.813	0.873
semi-supervised	0.893	0.925	0.909	0.913	0.731	0.812
no segmentation	0.745	0.845	0.792	0.758	0.834	0.794
LSTM	0.911	0.707	0.789	0.932	0.955	0.942
HAN	0.918	0.941	0.929	0.965	0.962	0.963

Table 6. The performance of the methods when applied to the test dataset. Some values are different from the ones shown in the overview paper of the PoliInfo task [9], because the mean values of 5 runs are shown here.

	question			answer		
segmentation method	recall	precision	F1	recall	precision	F1
rule-based	0.851	0.913	0.881	0.949	0.903	0.925
SVM	0.819	0.851	0.834	0.913	0.939	0.925
semi-supervised	0.836	0.760	0.796	0.907	0.814	0.858
no segmentation	0.828	0.715	0.767	0.680	0.839	0.751
LSTM	0.916	0.690	0.780	0.909	0.925	0.914
HAN	0.871	0.874	0.873	0.949	0.921	0.934

Besides the no segmentation case, the idf-based equation is used with the weight λ 0.4 for questions and 0.7 for answers. In the no segmentation case, the tf-idf-based equation with λ 0.7 for questions and 0.9 for answers gives the better results.

3.2 Classification

Before we submit the runs for formal run, we have locally evaluated the performance of each methods by dividing data into training and test data (4:1). Table 7 shows our local evaluation results for each method. This is helpful for us to verify the effectiveness of each method visually. Note that Methods 1-4 perform evaluation and parameter determination with the following standards from the view of detecting abnormal values.

Note that the F1 score in Table 7 for each method is not unified. Method 1/5/6/7/8 takes the highest micro F1 score of majority, and in contrast to method 1, methods 2/3 takes the highest F1 score of the minority label (Relevance: “0”, Stance: “1” and “2”). Therefore, the score is relatively lower. As to method 4, The F1 score is calculated by cosine similarity without parameter adjustment or model selection.

Table 8 shows the results of formal runs with the gold standard provided by the task organizer and the method combinations for submitted runs. We have combined our 8 methods into 7 runs for the test data (365 sets). The same results also are shown in the overview paper of the PoliInfo task [9]. In classification, we generally have got a higher score than the average.

4 Discussion

4.1 Segmentation

By comparing Table 4 and Table 5, we see a better segmentation tends to give a better result. The segmentation step before the search step is important, though not mandatory.

Table 7. The local evaluation results for our methods applied to the training dataset.

Labeling method	F1-score
1.Relevance, Stance (1)	0.88 (Relevance),0.84 (Stance)
2.Relevance, Stance (2)	0.20 (Relevance/label 0)
	0.14 (Stance/label 1)
	0.09 (Stance/label 2)
3.Relevance, Stance (3)	0.805 (Relevance) 0.625(Stance)
	0.29 (Relevance/label 0)
	0.37 (Stance/label 1)
	0.22 (Stance/label 2)
4.Relevance, Stance (4)	0.814 (Relevance) 0.631(Stance)
5.Stance (5)	0.97
6.Fact-Checkability (1)	0.94
7.Fact-Checkability (2)	0.89
8.Fact-Checkability (3)	0.87

Table 8. The final results of formal runs for class label.

		Support			Against			Other		
Submitted Run	Accuracy	R	P	F	R	P	F	R	P	F
Run1(method1,6)	0.933	0.0	NaN	NaN	0.0	NaN	NaN	1.000	0.933	0.965
Run2(method1,7)	0.932	0.002	0.091	0.004	0.004	0.111	0.008	0.998	0.933	0.964
Run3(method1,8)	0.893	0.118	0.145	0.130	0.111	0.117	0.114	0.949	0.940	0.944
Run4(method2,6)	0.894	0.114	0.143	0.127	0.111	0.117	0.114	0.950	0.939	0.944
Run5(method3,6)	0.933	0.0	NaN	NaN	0.0	0.0	NaN	1.000	0.933	0.965
Run6(method4,6)	0.933	0.0	NaN	NaN	0.0	NaN	NaN	1.000	0.933	0.965
Run7(method1,5,6)	0.932	0.084	0.141	0.141	0.042	0.407	0.076	0.994	0.937	0.965

It is not surprising that the rule-based method gives a good performance for the segmentation, because the human annotator also has similar patterns for cue phrases in mind. Although the SVM method and the LSTM method are not so successful as the rule-based method, we think machine learning methods can be better or at least comparable to the rule-based method. This is the reason why we tried the HAN method after the formal run. The HAN method now accomplishes the performance close to the rule-based method.

The semi-supervised method is interesting, because we need not prepare a large training dataset. No additional labeling jobs are actually necessary in our case, because we can rely on boundaries between speakers to get initial labels. There can be some cue-phrase patterns that do not appear in training data. The semi-supervised method can also find such patterns.

Fig. 3 shows how recall and precision in the segmentation step change according to iterations of the semi-supervised method. The plots show that recall and precision converge in the shown cases. When the probability threshold parameter is 0.80, precision decreases with iterations, which is not preferable. (The iteration 0 is exceptional, where precision is 1, because boundaries between speakers are always boundaries between segments.) Recall is considered to be more important than precision for the segmentation step, because segments will be concatenated in the search step. We choose the threshold 0.85 and the 8th iteration for the submission.

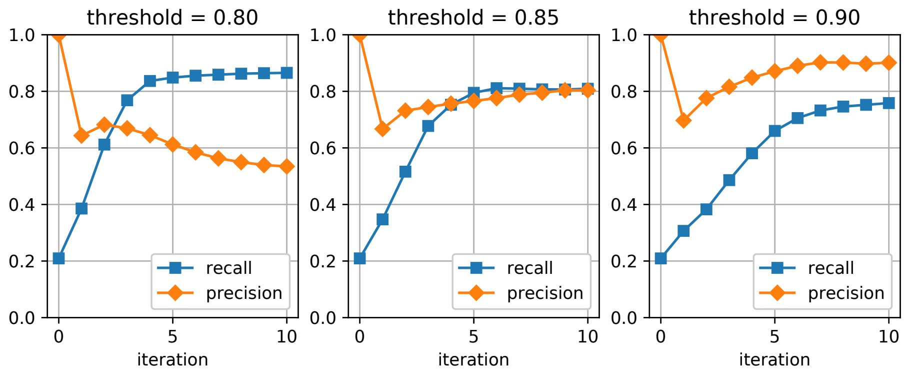


Fig. 3. As for the semi-supervised method, the convergence through iterations depend on the probability threshold parameter.

Fig. 4 shows the effect of the weight parameter λ in the search step. Recall, precision, and F-measure for the PoliInfo segmentation subtask are plotted. The rule-based segmentation and the test dataset provided by the task organizer are used in this case. A smaller λ yields longer sequences of segments that means higher recall and lower precision. There is an optimum value of λ to get the highest F-measure. The optimum value for answers is different from the one for questions. We think the differences in length and characteristics of utterances are the reasons of the difference.

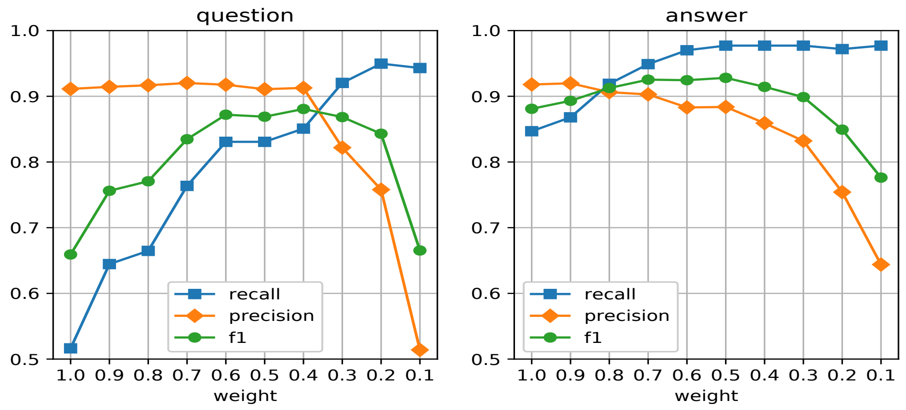


Fig. 4. The balance between recall and precision changes according to the weight parameter λ.

By comparing Table 5 and Table 6, we see the performances for the test dataset are often worse than the performances for the training dataset. This is little unexpected, because the training dataset is not used to train our models, though used to tune hyperparameters. In fact, statistical tests show no significant differences for most metrics. The size of the dataset is not large enough to correctly compare results.

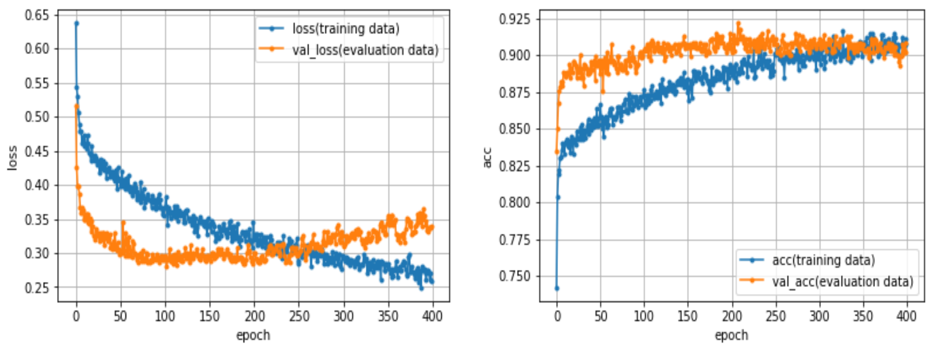
4.2 Classification

From observing the table results in the section 3.2, there are some findings for Relevance, Fact-Checkability, Stance text classification respectively.

For Relevance, it was confirmed that the abnormal value detection method using Isolation Forest contributes to the performance improvement of minority labels. What's more, correction by cosine similarity was also effective. By utilizing the approach of NLP, it may be necessary to separate the data distribution of majority / minority.

For Fact-checkability, we can find that No.6 method(Fact-Checkability(1)) with unanimous training data and LSTM classifier has achieved better results not only in our local evaluation experiments but also in the formal run than No.7,8 methods. The reason for that maybe is the high-quality unanimous training data. Since the unanimous training data is strictly selected by all-same standard, it is closest to data correctness on behalf of correct answer.

On the other hand, the LSTM method also shows the good performance to give a prediction at the cost of long execution time. Fig. 5 shows how loss and accuracy value changes according to iterations during the local evaluation stage. Although we apply Dropout, L1 Normalization process to alleviate the problem, we can see that there is also a trend of overfitting. However, we can draw a conclusion that classifier has grasped and learned the features of utterance.



(a) Loss (b) Accuracy

Fig. 5. The curve of loss and accuracy change values in the local evaluation of fact-checkability classification.

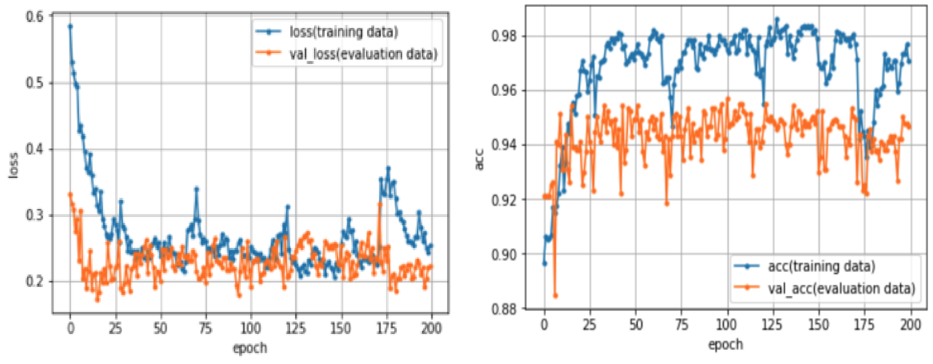
For Stance, the deep learning method LSTM also achieve a good result for both local evaluation experiments and formal run. We assume that all support utterance or against utterance have common features regardless of different topics. Unlike the LSTM result above for fact-checkability, here the curve of loss, accuracy values in the Fig. 6 shows that the overfitting problem has been almost restrained instead of great fluctuation. Here we adopt the method of raising the learning rate to alleviate the unstable problem.

There are some cases in which the LSTM method is better than the abnormal value detection method using Isolation Forest. We can draw a conclusion that the Stance judgment is more sensitive to words order than word meaning. Honestly, the correct answers of tasks (Relevance / Stance) are subjective depending on one's beliefs, ideas, environment, experience, etc. Therefore, we need to set criteria of correct answer in the first step.

5 Conclusions

Our contributions concerning the segmentation subtask are summarized here.

1. We showed the assembly utterances can be effectively segmented by cue phrases.



(a) Loss (b) Accuracy

Fig. 6. The curve of loss and accuracy change values in the local evaluation of Stance classification.

2. Although a rule-based approach gives good segmentation results, we showed a neural network approach can accomplish almost the same precision and recall.
3. We also proposed a semi-supervised method for the segmentation, where no additional labeling jobs are required.
4. We proposed a simple but effective statistical model to find a segment that corresponds to a given summary. It can be a baseline for more advanced methods that take syntactic or semantic features into account.

The main contributions to classification subtask are summarized in the below.

1. We have showed the assembly utterances can be classified by learning methods with a high accuracy.
2. We have treated the imbalance of training data as “abnormal value detection problem”. To solve this problem, we have proposed Isolation Forest and correction method based on cosine similarity, and the effect on minority label has been verified.
3. Although deep learning LSTM has obtained a better result in fact-checkability and Stance classification tasks, it costs much more time and resources than regular machine learning algorithm SVM and others. Maybe we need leverage the balance of accuracy and calculation cost in the future work.
4. The feature words and patterns are extremely important for politician utterance analysis since they could guide model to recognize the real impact factors on classification. The fully utilization of these features leads to a good learning process.
5. The selection of training data in the preprocessing stage also acts an important role for training and prediction. We shall determine the training data in consideration of quality and quantity.
6. From the results of LSTM, it seems that there is a possibility of improvement in accuracy by acquiring distributed representation considering word order.

References

1. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. pp. 385–393. Association for Computational Linguistics (2012)
2. Badjatiya, P., Kurisinkel, L.J., Gupta, M., Varma, V.: Attention-based neural text segmentation. In: European Conference on Information Retrieval. pp. 180–193. Springer (2018)
3. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Machine learning **34**(1-3), 177–210 (1999)
4. Choi, F.Y.Y.: Advances in domain independent linear text segmentation. In: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference. pp. 26–33 (2000)
5. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Computing Surveys (Csur) **40**(2), 5 (2008)
6. Eisenstein, J., Barzilay, R.: Bayesian unsupervised topic segmentation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 334–343. Association for Computational Linguistics (2008)
7. Gulli, A., Pal, S.: Deep Learning with Keras. Packt Publishing Ltd (2017)
8. Hearst, M.A.: Texttiling: Segmenting text into multi-paragraph subtopic passages. Computational linguistics **23**(1), 33–64 (1997)
9. Kimura, Y., Shibuki, H., Ototake, H., Uchida, Y., Takamaru, K., Sakamoto, K., Ishioroshi, M., Mitamura, T., Kando, N., Mori, T., Yuasa, H., Sekine, S., Inui, K.: Overview of the NTCIR-14 QA Lab-PoliInfo task. In: Proceedings of the 14th NTCIR Conference (2019)
10. Koshorek, O., Cohen, A., Mor, N., Rotman, M., Berant, J.: Text segmentation as a supervised learning task. In: NAACL-HLT (2018)
11. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
12. Nguyen, H.V., Bai, L.: Cosine similarity metric learning for face verification. In: Asian conference on computer vision. pp. 709–720. Springer (2010)
13. Riloff, E., Jones, R., et al.: Learning dictionaries for information extraction by multi-level bootstrapping. In: AAAI/IAAI. pp. 474–479 (1999)
14. Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at trec-3. In: Proceedings of the Third Text REtrieval Conference (1994)
15. Suzuki, M., Matsuda, K., Sekine, S., Okazaki, N., Inui, K.: Neural joint learning for classifying wikipedia articles into fine-grained named entity types. In: Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Posters. pp. 535–544 (2016)
16. Tagami, T., Asano, H., Yanai, H., Yamashita, R., Komiya, A., Fujimura, A., Machino, A., Inui, K.: Support for searching for articles to be verified for fact checking. In: JSAI (2018)
17. Vapnik, V.: The nature of statistical learning theory. Springer science & business media (2013)
18. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489 (2016)