Classification Between Rumors and Explanations of Rumors Based on Common and Difference Subsequences of Sentences

Xiaoping Sun, Junsheng Zhang, Yufei Sang

Classification between Rumors and Explanations of Rumors based on Common and Difference Subsequences of Sentences

Xiaoping Sun[1], Junsheng Zhang[2] and Yufei Sang[3]

[1] Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
sunxiaoping@ict.ac.cn

[2] Institute of Scientific and Technical Information of China, Beijing, China
zhangjs@istic.ac.cn

[3] School of Economics, Capital University of Economics and Business, Beijing, China
syf0810@126.com

Abstract. Preventing explosion of rumors on the Internet asks for a quick automatic detection mechanism that can detect rumors according to the given true information. Previous automatic rumors detection models are mainly built by training a supervised classification model on a labeled dataset containing rumor samples and true information samples. However, in many real cases, there is only one short piece of available true information sample given by an authority in form of an explanation or a correction of a rumor. The explanation sample is often short and very similar to rumor samples, making it difficult to train a discriminative classifier to check whether a piece of information is a rumor or an explanation of rumors. It is necessary to build a model to detect whether a short text is a rumor text or an explanation of rumors, which can be used to as evidences for detecting and refuting rumors. In this paper, we presented a sentence preprocessing method that extracts the leftmost longest common sequence to obtain the common and difference subsequences between the rumor text and its explanation text to compose samples and train a supervised model for classification between rumors and explanation of rumors. Experiments show the effectiveness of the proposed method.

Keywords: Rumors, Explanation of rumors, Classification, Sentence subsequence.

1 Introduction

With the boom of the mobile Internet and social media, rumor spreading has caused huge impact on social development, especially during the pandemic of Covid-19 since 2019 [1]. Rumors can be quickly spread on the Internet in form of short texts through social media platforms like Weibo or Twitter [2]. For example, during the pandemic of COVID-19, there are lots of rumors on that drinking much more tea or vinegar can protect people from infection. To prevent rumors from being spread on the Internet,

automatic rumor detection techniques becomes necessary tools [3, 4]. Supervised automatic rumor detection models mainly treat the problem as a binary classification problem over a training sample set consisting of positive and negative sample cases [5]. There lacks interpretation in such supervised models. Evidence aware models leverage clues extracted from positive or negative samples to give evidences on the prediction results [6]. However, training samples of rumors are often imbalanced, especially in COVID-19 rumor detection tasks. There is still a challenge in finding positive samples because most posted information are rumors and there is often only one or few piece of information published by an authority as an explanation of rumors. Detecting whether a text is a rumor or an explanation of a rumor can help make a more explainable rumor classification model. This issue has not been fully recognized in the misinformation detection research. Moreover, the text of an explanation information is often very similar to texts of those widely disseminated rumors, which makes it difficult to discriminate a rumor text from an explanation text of the rumor by a classification model trained over imbalanced data sample set. To address this problem, we present a data preprocessing method that extracts common and difference subsequence between a rumor text and its corresponding explanation text to compose a rumor sample and its explanation sample for training a supervised model. Using common and difference subsequences as the training samples can make the sample features more discriminative and thus can be used to build more accurate classification model for predicting rumor text and their explanation text. We collected a set of samples of COVID-19 rumors, where each sample has a rumor text, a refutation text and an explanation text given by a medical authority. Experiments show that using the common and difference subsequences of text samples to train a supervised model can improve the accuracy of classification on rumors and explanation of rumors.

2 Related work

Automatic misinformation detection models can be categorized into two classes, content-based methods [7] and network-based methods [3, 8]. In content-based methods, detecting misinformation is modeled as a binary classification problem. A classifier is trained on a given sample set consisting of true information samples and misinformation samples. SVM and deep learning models have been investigated for this task. The core challenge in content-based methods is the feature selection of misinformation sample data sets [9]. Many language features have been investigated for building misinformation classifier [10, 11]. In network-based methods, features of propagation network of misinformation on the Internet are leveraged to detect the misinformation event [12, 13].

The difficulties of misinformation detection lie mainly in that misinformation propagation are often event related [5, 14]. Texts are often short and samples are often imbalanced because in the beginning stage of the propagation of an event, most available samples are misinformation samples and there is even no positive information available. Moreover, explanation, refutation and correct information are published by

authorities. So the number of positive information samples is often extremely small at the beginning stage of an event, which makes it difficult to obtain a balanced training sample set. For example, in an authority site where refutation on COVID-19 rumors are published, each statement about a rumor has one explanation and one conclusion description, but there are already hundreds or thousands of rumors on the social networks [15]. Those official explanation or conclusion can be used to detect misinformation but there is still few work on leveraging this information due to that the official explanation text is often quite similar to rumor text, which makes it difficult to train a supervised model for detecting whether a text is a rumor, or is an explanation of a rumor. Evidence-aware misinformation prediction has been studied [1, 6, 16]. However, most works need an enough number of positive samples to train a model and give clues on a prediction that why a text is misinformation or not. It is necessary to investigate how to leverage the explanation text of rumors to detect misinformation. Moreover, explanation texts given by an authority to clarify rumors can be used as explanation of classification results on rumors. The first step is to make a classification between a piece of rumor and an explanation of rumors. Automatic detection of explanation texts of rumors can help find the explanations for rumors.

3 Model

Given a set of short texts consisting of samples of pairs of rumors and their corresponding explanations, the task is to build a predictor to judge whether one given input sample text is a rumor or an explanation of a rumor. To discriminate a rumor from its explanation, a solution is to train a supervised classification model on a training set consisting of two sets of samples: one sample set contains rumor texts, and another contains explanation texts. Then we use the sample set to train a classifier that predicts a sample's type. Since the texts of explanation are often very similar to the rumor texts, we proposed to use sentence difference of samples texts as the texts of samples to build a classifier.

The main idea is that the difference between a rumor text and its explanation text is more representative than the original texts of the rumor and its explanation. It is because that a pair of rumor and explanation often shares too many words, and the difference between a rumor and its explanation often contains those common words among samples, which can be ignored in training a model. We leverage the difference part between sentences to train a classifier for classifying samples into two classes: a rumor or an explanation of a rumor.

Fig.1 depicts the whole framework of training a classification model from samples of rumor texts and explanation texts. The first part is the training sample preprocessing module that extracts difference and common subsequences from rumor texts and their corresponding explanation texts. Specifically, the difference subsequences can be obtained from the results of common subsequences. Then, the original sample texts are transformed into a set of subsequences of common parts and difference part of the original texts of samples. Those subsequences of texts are concatenated to form

a new text string for each sample. Finally, those newly composed samples are used as the training samples to train a supervised classification model.

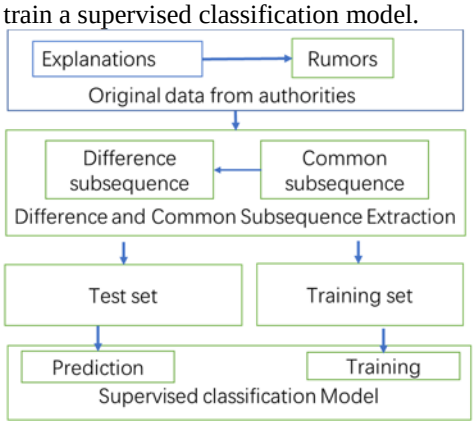

Fig. 1. Framework of processing explanations and rumors for training supervised classification models.

3.1 Data Samples

To ease discussion, we use the following symbols to represent samples. Given a rumor sample data set *M*, each sample *t* in *M* contains four items:

(1) *t.T*: the sentence text of a rumor *t*;

(2) *t.L*: a label indicating if the rumor *t.T* is a piece of rumor, true information or unknown information;

(3) *t.E*: the explanation text to show why the rumor is labeled as *t.L*.

(4) *t.F*: the conclusion text on the rumor label, i.e., giving the final judgement based on the explanation.

This kind of samples can be collected from those official rumor information management websites, where each sample consists of an explanation of refutation on one rumor. We will use this sample data set to train a predicator for text classification between rumor and explanation, or conclusion of a rumor based on sentence subsequences.

Table 1 shows three examples of samples. Each row represents one sample, consisting of a sample label, sample sentence texts, an explanation and a conclusion. There are three different labeled samples. For example, the first row is labeled as a rumor, saying that "*people do not need wearing mask after vaccination*". The explanation is given on why this argument is false and the conclusion is given by a famous doctor who is an authoritative professor in COVID-19 medication. From this example, one can see that the three text components of the first sample case are quite different in syntax level. The explanation text is much longer than the rumor text and contain words from the original rumor's text, while the conclusion text is totally different from the rumor text. The explanation of row 1 shares words "*vaccination*" and "*protection*" with the rumor text, so this common part can be deemed as the core content related to both the rumor and the explanation of the rumor. If these two words are

treated as features, then the explanation text could be also classified as a rumor text. To reduce this ambiguity in training samples, we assume that the difference part between rumor text and explanation texts are more discriminative features. For example, in row 1, the difference part between the explanation and the rumor text is about why one should pay attention to the protection even after vaccination.

Table 1. Examples of sample sets

t	t.L(Label)	t.T(Text)	t.E(Explanation)	t.F(Conclusion)
1	RUMOR	We do not need wearing a mask for protection after vaccination.	China's vaccine is mainly whole virus inactivated vaccine. Its safety is very good. If its effect is evaluated, the antibody appears after the first dose, which is about 60% and 70%. 14 days after the second dose, its antibody level can reach nearly 90%. There is no antibody until at least two to three weeks after the vaccination. There is still a risk of infection between the vaccination and the production of the antibody, and personal protection should also be paid attention to.	Zhong Nanshan denied it.
2	TRUE	The dog in the infected person's home is weakly positive	A news report confirmed that a pet dog was tested weakly positive for the virus test, according to the Hongkong Wenhui network. Hongkong food and health secretary Chen Zhaoshi stressed on 28, there is no evidence that dogs will spread COVID-19 to humans. She called on pet owners to pay attention to personal protection. Chen Zhaoshi said that the dog would receive medical supervision from the veterinarian and take samples for testing. It would not be returned until the test results were negative.	Chen Zhaoshi, director of the Hongkong food and food administration, stressed that although there was a weak positive finding, there was no evidence that dogs would transmit COVID-19 to humans.
3	UNKNOWN	The virus will stay in the throat for 4 days. Can it be remedied ?	There is a rumor that "COVID-19 has to stay in the throat for 4 days before reaching the lung. At this point, the patient will begin to have symptoms of cough and sore throat. But if you drink a lot of water and gargle with warm water, salt or vinegar, you can eliminate the virus." There is still no positive or negative evidence show that the virus will stay in the throat for 4 or more days. In addition, brine, vinegar and water did not kill COVID-19.	There is no research data to support this statement.

The second row in Table 1 is a sample of true information saying that a virus test on a pet dog of a patient shows a positive result. It is true according to the official report. The explanation and the conclusion both repeat the details of the official report. There are many common parts in the rumor, explanation and conclusion text. When removing those common part from the explanation text, the difference part is the official confirmation that can be used as features to determine the text type.

The third row in Table 1 contains an unknown sample case which means that it is still unclear that whether the text is a rumor or not. Similarly, the difference part can be used to distinguish the conclusion text from the original rumor text. However, the explanation text contains a subsequence string that repeats the original rumor text, makes it difficult to obtain a discriminative feature for classifying the rumor text and the explanation text for this sample case. When removing those common parts, it can be more distinguishable as an explanation sample. Thus, the observation on the sample texts shows that both the common parts and the difference parts between texts can play important roles in making discriminative features for training a supervised classification model that can be used to distinguish the explanation of rumors from rumors.

3.2 Sentence subsequence preprocessing

To extract common and difference part from sample texts in the training set *M*, sentence text *S* of a rumor or an explanation is firstly represented as an *n*-gram character sequence $S=\{S_1, S_2, ..., S_n\}$, where S_i is a *i*th *n*-gram character in *S*. *n*-gram is a widely used text feature model that takes every *n* consecutive words from a text sequence as one keyword unit. In processing Chinese texts, an *n*-gram unit consists of *n* consecutive characters from the text. In this work, we use the uni-gram model as the basic unit for representing rumor texts because the word segmentation problem can be skipped when using uni-gram to represent Chinese text sentences. Thus, each uni-gram of a sentence is treated as a character.

A subsequence *s* of *k* characters in *S* is a character sequence $s=\{S_j, S_{j+1}, ..., S_{j+k}\}$ for $1 \le j \le n-k$. Two subsequences of characters are equal if each corresponding character is the same.

Two types of subsequences are defined for two sentence sequences of characters. A common subsequence contains characters shared by two sequences while a difference subsequence records the different part between two sentence sequences.

Definition 1. (Common subsequence). Given two sentence character sequences *S* and *T*, the common part between two sentence character sequences is a set of subsequences, denoted as $C(S, T)=\{s_1, s_2, ..., s_m\}$, where s_i is a subsequence in both *S* and *T*. That is, *C*(*S*, *T*) contains sub-characters that are shared by two sentences *S* and *T*.

Definition 2. (Difference subsequence). The difference between *S* and *T* contains s of characters that are obtained by excluding their common part from *S*, denoted as $D(S, T) = S–C(S, T)$. Similarly, the difference between *T* and *S* is defined as $D(T, S) = T–C(S, T)$. So the difference between two sentence character sequences is not symmetric.

Both the common part and difference part between two text character sequences can be used as features to train a supervised classification model. Thus, we need to extract common sequences and difference sequences for two text character sequences. Note that the difference subsequence in **Definition 2** is defined based on the common subsequence in **Definition 1**. Thus, to obtain the difference between two sentences *S* and *T*, we first compute the common part shared by *S* and *T* and then the difference part can be obtained from the common part.

The common subsequence is obtained by the algorithm *computecomm* in Fig.2. In the algorithm, line 4 to 19 is the loop to traverse each character of *S* to find if there is any common character in *T*. During the loop, if there is a common character from *S* for current character of *T*, then it will be added to the current common subsequence list (*subc1* in line 6 of *computecomm*) until it encounters a different character of *T*. Then current common subsequence is recorded and is initialized for the next possible common subsequence. It can be shown that the algorithm *computecomm* can find leftmost longest common subsequences of *S* and *T* in at most $O(N_1*N_2)$ time, where N_1 and N_2 is the length of *S* and *T*.

```
ALGORITHM:computecomm
INPUT:   S: Sentence character sequence
         T: Sentence character sequence
OUTPUT subsame: list of common sub-sequence of S and T
         allc: string of common sequence part
 1 n1 = len(S);n2 = len(T)
 2 i = 0,j = 0
 3 subc1 = [];subsame = []
 4 while(i<n1):
 5     if (S[i] == T[j]):
 6         subc1.append(i #find the start of common part
 7         i = i + 1;j = j + 1
 8         if j>=n2:
 9             j = 0
10         continue
11     else:
12         if len(subc1)>0 #record the common part
13             subsame.append(subc1)
14             subc1 = []
15             j = 0
16         else:
17             j = j + 1
18         if j>=n2:
19             i = i + 1;j = 0
20 if len(subc1) > 0:
21     subsame.append(subc1);subc1 = []
22 subsamestr = [];allc = ''
23 for s in subsame:
24     st = ''
25     cstart = s[0]
26     cend = len(s)
27     for c in s:
28         st = st + c1[c]
29     subsamestr.append([st,cstart,cend])
30     if len(st)>0:
31         allc = allc +segword(st) + ' '
32 return subsamestr, allc
```

Fig. 2. Algorithm for finding the common part from sentence character sequence *S* and *T*.

Definition 3. The leftmost longest common subsequence of *S* in *T* is a character sequence $l_1=\{S_{i+0}, S_{i+1}, ..., S_{i+k}\}$ in *S* s.t. that there is a subsequence $l_2=\{T_j, T_{j+1}, ..., T_{j+k}\}$ in *T* with $S_i=T_j$, $S_{i+1}=T_{j+1}$, ..., and $S_{i+k}=T_{j+k}$, but there is no such subsequence of $l_1=\{T_{p+0}, T_{p+1}, ..., T_{p+n}\} = \{S_{i+0}, S_{i+1}, ..., S_{i+n}\}$ with $p<j$ and $n \le k$.

That is, we only locate the first occurrence of a common subsequence of *S* in *T*. For example, let *S*="*the covid-19 syndrome include cough, chill and muscle pain*" and *T* = "*cough, chill are common syndrome. cough, chill and muscle pain are also reported widely*". Here, "*cough, chill*" is a common subsequence for *S* and *T*, but "*cough, chill and muscle pain*" is also a common subsequence for *S* and *T*. In this example, "*cough, chill*" is the leftmost longest common subsequence while "*cough, chill and muscle pain*" is not because when matching "*cough, chill and muscle pain*" of *S* in *T*, "*cough, chill*" is already matched before. Thus, "*cough, chill and muscle pain*" will not be extracted as a longer common subsequence. We do not use the longest common subsequence because that the longest common sequence will overlap those shorter common subsequences. The leftmost longest common subsequence is the first occurrence of a common sequence. Using the leftmost longest common subsequence can avoid those shorter common subsequences being overlapped.

THEOREM 1. All leftmost longest common subsequences of S in T can be obtained by the algorithm *computecomm* with time complexity of $O(N_1*N_2)$.

PROOF. Assume there is such a common subsequence $l_1=\{S_i, S_{i+1}, ..., S_{i+k}\}$ starting at S_i in S, with corresponding subsequence $l_2=\{T_j, T_{j+1}, ..., T_{j+k}\}$in T. That is, S_{i-1} is different from T_{j-1}. Since the algorithm *computecomm* iterates every character of S started at line 4, it must meet the character S_{i-1} and thus, any previous subsequences will be recorded and a new subsequence will be started by line 12-15 where iteration in T is restarted from the beginning by line 15. Thus, the next loop will check S_i by starting from the very beginning part of T to find the common character for S_i. If there is a character $T_p = S_i$, then, it will be recorded into the common subsequence, denoted as *subc1* in line 6 of the algorithm *computecomm*, until meeting a different character, then the recorded subsequence *subc1* is a common subsequence within l_1 and l_2 and is the left most common subsequence of S in T. If it is not the left most, there must be another subsequence l_3 starting before *subc1*. But it is impossible, because the checking point starts at the beginning item of T for l_1, l_3 cannot be missed by *subc1*. Thus, *subc1* must be a left-most common subsequence of S in T. The whole scanning procedure in T for each character of S is at most one time. So the time complexity of the algorithm *computecomm* is $O(N_1*N_2)$. □

After obtaining common parts of S and T, we remove those common subsequences from S to obtain the difference part between S and T. Fig. 3 depicts the procedure *computediff* that leverages the algorithm *computecomm* in Fig.2 to get the difference subsequences. Note that during the procedure of locating the common subsequences in the algorithm *computecomm* in Fig.2, the starting and ending position of a common subsequence is also recorded in line 29. We can use the position information to remove common parts from sentence character sequence S for obtaining the difference between S and T. In *computediff* , first, *computecomm* procedure is called to obtain the common subsequence list that contains all the matched leftmost common subsequence. Then, *computediff* iterates each character of S in line 4. Within each loop of a character of S, there is another loop for each common subsequence, where if current character of S is within a common sequence, then skip this character and record the difference subsequence that has been located and go for the next character (line 11 to line 14). If current character of S does not exist in any common subsequence, then put it into the current difference subsequence and go to the next character at line 16 and line 17.

The loop from line 4 to line 17 in the algorithm *computediff* also takes $O(N_1*N_2)$ because it has one loop for each character of S and within each loop there is one scan for all the common subsequence that has at most N_2 items.

```
ALGORITHM: computediff(S, T)
INPUT:     S: Sentence character sequence
           T: Sentence character sequence
OUTPUT:    difflist: difference part of S and T
           allc: string of difference sub sequence
n1 = len(S);n2 = len(T)
i = 0; l2 = len(T); diff = [];difflist=[];
commlist, comstr = compuecomm(S,T)
while(i<n1):
    iinsame = False
    for c in commlist:
        if i>=c[1] and i <c[1]+c[2]:
            iinsame = True
            break
        if iinsame:
            if len(diff)>0:
                difflist.append(diff)
                diff = []
            i = i + 1
        else:
            diff.append(c1[i])
            i = i + 1
if len(diff)>0:
    difflist.append(diff)
allc = ''
for sc in difflist:
    if len(sc)>0:
        allc = allc + " ".join(sc) +" "
return difflist,allc.strip()
```

Fig. 3. ALOGRITHM *computediff* for finding the difference part from sentence character sequence *S* and *T*.

3.3 Training a supervised classification model by subsequences of sample sentences

We use common and difference parts of sentence of sample texts to build a new sample set from the original sample set *M* for training a supervised classification model. Specifically, for each sample case *t* in *M*, we use the difference and common parts between rumor text *t.T*, rumor explanation *t.E* and rumor conclusion *t.F* to compose the samples to train a classification model for determining three classes: *rumor*, *explanation*, and *conclusion* class.

That is, for each sample *t* with *t.T* as a *rumor* sample, *t.E* as an *explanation* sample and *t.F* as a *conclusion*, we apply the common subsequence extractor *C*(*S,T*) and difference subsequence extractor *D*(*S, T*) to compose three new sample texts from the original texts of *t.T*, *t.E* and *t.F*, which are used as three new sample texts, represented by *t'.T*, *t'.E* and *t'.F* for three classes: *rumor*, *explanation*, and *conclusion* respectively. After processing all samples of *M*, we build a new sample set M_e and we can train a supervised model P_e over M_e to classify a text into three classes: *rumor*, *explanation*, and *conclusion*.

For each training sample *t* in *M*, we consider three different strategies for transforming *t* into *t'*. In the follow three strategies, the *explanation* sample is obtained by the difference between the original rumor text and the original explanation, and the

conclusion sample is obtained by the difference between the original rumor text and the original explanation text. The difference is how to compose a rumor sample text:

Case (1): $t'.T = t.T$, $t'.E = D(t.E, t.T)$, and $t'.F = D(t.F, t.T)$. The original rumor text is kept as a new sample of *rumor* class;

Case (2): $t'.T = C(t.T, t.E)$, $t'.E = D(t.E, t.T)$, and $t'.F = D(t.F, t.T)$. The *rumor* sample text is the common part between the original rumor text and the original explanation;

Case (3): $t'.T = C(t.T, t.E) \cup C(t.T, t.F)$, $t'.E = D(t.E, t.T)$, and $t'.F = D(t.F, t.T)$. The *rumor* sample text combines the common part between the original rumor and the original explanation and the common part between the original rumor text and the original conclusion text.

The rational is that the shared common part of the rumor and its explanation is the core text of the rumor. The difference between the rumor text and the explanation text is the representative part of explanation. The conclusion text is also represented by the difference parts between the original rumor and its conclusion text.

After processing the original training data sample set M, we can obtain a new training sample set M_e consisting samples for *rumor*, *explanation* and *conclusion* class. Then, we can used M_e to train a supervised classification model like SVM model or Bayesian model.

4 Experiments

To evaluate the proposed method, we collect a set of officially published COVID-19 rumor refutation information data set consisting of 134 rumors and their corresponding explanation texts and conclusion texts. Then we apply the preprocessing in Section 3.3 to build three new different training sample text sets with different configurations using sentence common parts and sentence difference operators. For comparison, a simple sample set without using the preprocessing method is built where the original texts of rumor, explanation and conclusion of a sample are directly used as three new samples for class *rumor*, *explanation* and *conclusion* (*Original* in the Table 2).

Three types of classification models are evaluated on the training sample sets, naive Bayesian model (NB1 and NB2), SVM model (SVC1, SVC2 and SCVRBF) and kNN model (KNN_5_distance and KNN_5_uniform, using two different distance model in KNN). Table 2 shows the accuracy of the models obtained by training on the original sample data sets and three data sets consisting of sentence difference and common parts. It can be seen that training on the sample data set obtained by configuration defined in Case (1), (2) and (3) can help models achieve improvements on the prediction quality. Moreover, the training sample set obtained by using configuration in Case (3) can achieve the best overall performance (the best average scores shown in Table 2 and Table 3). In Case (3) configuration, the rumor sample text combines all the common subsequences obtained with the explanation text and the conclusion text. It can help reserve more information of rumor samples. Table 3 shows the models' performance on three classes. In most tests, the improvement is obvious when using

the common parts and difference part as sample texts to train supervised classification models compared with the models trained on the original sample texts.

Table 2. Accuracy of models training on different sample sets

Model	Original	Case(1)	Case(2)	Case(3)
NB1	0.843	0.891	0.891	0.891
NB2	0.333	0.333	0.333	0.333
SVC1	0.925	0.963	0.965	0.968
SVC2	0.928	0.965	0.965	0.965
SVCRBF	0.866	0.878	0.893	0.888
KNN_5_distance	0.836	0.930	0.923	0.928
KNN_5_uniform	0.746	0.881	0.883	0.888
Average Score	0.783	0.834	0.836	**0.837**

Table 3. F-score of performances on three classes by models training on different sample sets.

Performance on classes	Original	Case(1)	Case(2)	Case(3)
NB1-Rumor	0.861	0.889	0.889	0.889
NB2-Rumor	0.500	0.000	0.000	0.000
SVC1-Rumor	0.949	0.985	0.985	0.989
SVC2-Rumor	0.949	0.985	0.985	0.985
SVCRBF-Rumor	0.913	0.907	0.933	0.924
KNN_5_distacne-Rumor	0.847	0.950	0.942	0.946
KNN_5_uniform-Rumor	0.736	0.908	0.917	0.920
NB1-Explanation	0.820	0.865	0.862	0.867
NB2-Explanation	0.000	0.500	0.500	0.500
SVC1-Explanation	0.933	0.957	0.964	0.964
SVC2-Explanation	0.935	0.960	0.964	0.964
SVCRBF-Explanation	0.867	0.847	0.864	0.858
KNN_5_distacne-Explanation	0.804	0.906	0.899	0.906
KNN_5_uniform-Explanation	0.706	0.843	0.846	0.851
NB1-Conclusion	0.857	0.924	0.928	0.921
NB2-Conclusion	0.000	0.000	0.000	0.000
SVC1-Conclusion	0.891	0.946	0.947	0.950
SVC2-Conclusion	0.897	0.951	0.947	0.947
SVCRBF-Conclusion	0.807	0.889	0.889	0.889
KNN_5_distacne-Conclusion	0.859	0.936	0.929	0.933
KNN_5_uniform-Conclusion	0.808	0.893	0.889	0.897
Average Scores	0.759	0.811	0.813	0.**814**

5 Conclusion

In this paper, we proposed a method to extract common and difference subsequences of sentence texts as samples to train supervised classification models to classify samples into rumor, explanation and conclusion class. The leftmost longest common subsequence is extracted as the common part from two sample sentence texts. Difference part between two sample texts is obtained by removing common subsequences from the origin sentence. Common and difference subsequences are combined in different ways to compose samples of rumor, explanation and conclusion classes. Experiments shows that when using the difference subsequence between rumor and explanation texts as the explanation samples, using the difference subsequence between rumor and conclusion texts as the conclusion samples, and combining common subsequences obtained from both explanation and conclusion texts as the rumor samples, the model achieves the best overall performance of classification accuracy compared with models trained on the original sample texts. The subsequences of sample sentences can be further investigated to reflect richer features of samples for training supervised models.

Acknowledgement

This work was partially supported by the Joint Project of CAS and Austria on ADaptive and Autonomous Data Performance Connectivity and Decentralized Transport Decision-Making Network (ADAPT, No. 881703) and the Innovation Funding Project “Internet Fake News Detection Method Research” (No.MS2021-05) granted by Institute of Scientific and Technical Information of China.

References

1. Martinez Monterrubio, S. M., Noain-Sánchez, A., Verdú Pérez, E. and González Crespo, R. Coronavirus fake news detection via MedOSINT check in health care official bulletins with CBR explanation: The way to find the real information source through OSINT, the verifier tool for official journals. Information Sciences, 574 (1), 210-237(2021).
2. Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M. and Procter, R. Detection and Resolution of Rumours in Social Media: A Survey. ACM Comput. Surv., 51(2), 1-36(2019).
3. Sharma, K., Qian, F., Jiang, H., Ruchansky, N., Zhang, M. and Liu, Y. Combating Fake News: A Survey on Identification and Mitigation Techniques. ACM Trans. Intell. Syst. Technol., 10(3), 1-42 (2019).
4. Harper, L., Herbst, K. W., Bagli, D., Kaefer, M., Beckers, G. M. A., Fossum, M. and Kalfa, N. The battle between fake news and science. Journal of Pediatric Urology, 16(1), 114-115(2020).
5. Bondielli, A. and Marcelloni, F. A survey on fake news and rumour detection techniques. Information Sciences, 497 (2019/09/01/ 2019), 38-55(2019).

6. Popat, K., Mukherjee, S., Strötgen, J. and Weikum, G. Where the Truth Lies: Explaining the Credibility of Emerging Claims on the Web and Social Media. In Proceedings of the Proceedings of the 26th International Conference on World Wide Web Companion (Perth, Australia, 2017). pp.1003–1012, International World Wide Web Conferences Steering Committee, 2017.
7. Pan, J. Z., Pavlova, S., Li, C., Li, N., Li, Y. and Liu, J. Content Based Fake News Detection Using Knowledge Graphs. Springer International Publishing, City, 2018.
8. Zhou, X. and Zafarani, R. Network-based Fake News Detection: A Pattern-driven Approach. *SIGKDD* Explor. Newsl., 21(2), 48–60(2019).
9. Thorne, J., Vlachos, A., Christodoulopoulos, C. and Mittal, A. FEVER: a large-scale dataset for Fact Extraction and VERification. In Proceedings of the North American Chapter of the Association for Computational Linguistics (4/16/2018, 2018), pp.809-819, Association for Computational Linguistics, 2018.
10. Volkova, S., Shaffer, K., Jang, J. Y. and Hodas, N. Separating Facts from Fiction: Linguistic Models to Classify Suspicious and Trusted News Posts on Twitter. Association for Computational Linguistics, pp. 647–653, Vancouver, Canada, 2017.
11. Wadden, D., Lin, S., Lo, K., Wang, L. L., Zuylen, M. v., Cohan, A. and Hajishirzi, H. Fact or Fiction: Verifying Scientific Claims. In Proceedings of the Empirical Methods in Natural Language Processing (4/30/2020, 2020), pp. 7534–7550, Association for Computational Linguistics, 2020.
12. Wu, K., Yang, S. and Zhu, K. Q. False rumors detection on Sina Weibo by propagation structures. In Proceedings of the International Conference on Data Engineering (4/13/2015, 2015), pp. 651-662, 2015.
13. Tschiatschek, S., Singla, A., Rodriguez, M. G., Merchant, A. and Krause, A. Fake News Detection in Social Networks via Crowd Signals. In Proceedings of the The Web Conference (4/23/2018, 2018), pp. 517-524, Republic and Canton of Geneva, CHE, 2018.
14. Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J. and Zittrain, J. L. The science of fake news. Science, 359(6380), 1094-1096(2018).
15. Grimes, D. R. Health disinformation & social media: The crucial role of information hygiene in mitigating conspiracy theory and infodemics. EMBO Reports, 21(11), (2020).
16. Samadi, M., Talukdar, P., Veloso, M. and Blum, M. ClaimEval: integrated and flexible framework for claim evaluation using credibility of sources. In Proceedings of the Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, pp.222-228, Phoenix, Arizona, (2016).