DETECTION ON RUMOR PROPAGATION AND LATENT EVOLUTION

RUMOR DETECTION USING RNN AND VISUALIZATION MODEL OF RUMOR PROPAGATION AND EVOLUTION

Wang Yilin 517030910327

Wednesday 17th June, 2020

Abstract

Weibo is becoming an ideal social platform for rumor to spread because of the diversity of information, freedom of speech and explosive propagation speed of a microblog.

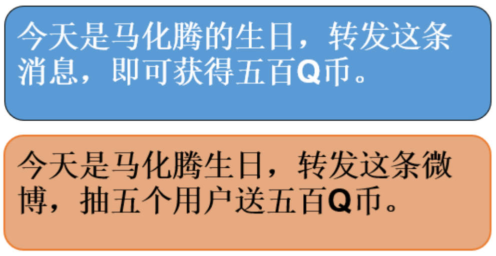
Therefore, an effective detection mechanism is indispensable. Instead of relies on platforms to refute rumors, which is usually with a delay, a better option is to design a algorithm to detect if a spreading microblog contains a rumor text.

In this paper, I put forward a model to detect rumor based on the propagation structure of a rumor, and further, put forward a mechanism to detect if a rumor evolved and becomes more confusing than the source.

1 Introduction

Rumor is always a problem in social network, because of the diversity of information in social platform like Weibo or twitter, rumors are becoming more and more rampant and it is more difficult for people to distinguish between a rumor and a normal news. This shows more significant during the COVID-19, even The People's Daily might also publish unconfirmed information, and thus causing many people rush to the pharmacy to buy Shuanghuanglian to resist COVID-19.

It is already confirmed that, it is impossible to judge if a microblog is a rumor or not, both for human and machines. That is because, a rumor always has the same grammatical structure as a news, and human can not judge if this is real or not. As for a machine, some machine learning methods also do no helps when only given the text. For example, as is shown in Figure 1.



今天是马化腾的生日，转发这条消息，即可获得五百Q币。

今天是马化腾生日，转发这条微博，抽五个用户送五百Q币。

Figure 1: Impossible to detect rumor based on text

The first text is a classical rumor that wildly spread in Qzone ten years ago, and the second text is a sweepstakes information in Weibo, and these two texts is almost the same, and most machine learning models like LSTM can hardly capture the latent meaning difference in a bulk of texts. In fact, studies show that it is almost impossible to detect a rumor only based on text information.

In most studies, features are extracted from the structure o f how a rumor propagated and all texts published by many other users related to the source. In this paper, I also chose to detect a rumor based on the latent information during the rumor propagating, collecting all users' forwarding text to do this work.

However, in most cases, rumor doesn't become a rumor the moment the source was published. For example, some one might published a microblog reporting a campus bullying, but some malicious marketing account might add highly coloured details, trying to mislead other users, and confirm them that some thing more than a campus bullying is happening, and thus a rumor is generated.

Such phenomenons shows that during a rumor's propagation, evolution might happen, making a truth lose the original meaning and becomes a rumor, or making a rumor more confusing than original.

In rumor evolution, the evolved text might be almost the same as the original text, but the meaning already changed significantly. To detect if evolution is happening during a text's propagation is difficult, because on the one hand, the evolved text is similar in the form, on the other hand, the detecting model should be able to know if a similar text has different meaning. Such a problem is similar to detect a rumor only based on text information, but given a target text to be compared.

In this paper, I put forward two distance metrics to detect evolution for a rumor. The core idea is to find texts that show up during propagation that close to the source in form, but far away in meaning.

2 Related work

Rumor detection is already well studied by now, there are many ideas to implement rumor detection.

Sejeong Kwon put forward a method to detect a rumor over varying time windows in 2017 [1], which take advantage of structural and temporal features, and also the linguistic features of users' comments during the propagation of a rumor.Tetsuro Takahashi also extracted features of a rumor and to judge if a twitter is a rumor [2].

Besides extracting features from structure and comments, much more implicit features can be used like popularity orientation, internal and external consistency, sentiment polarity and opinion of comments, social influence, opinion retweet influence, and match degree of messages [3]. And also, some work take advantaged of the rumor-denying information from official platform to extract features to train a classifier [4].

However, because the complexity of rumor detection and the limitation of human's knowledge, a better idea is to extract features automatically, therefore, deep learning is preferred now.

Jing Ma put forward a method using recurrent neural networks to detect a rumor in 2016 [5], which automatically extract features from text of comments during the propagation, and take structure of propagation as a time series, therefore, the while rumor structure can be fed into a RNN model. Further, Ma put forward a novel RNN model, which changed a time series into a tree structure, which maintained more information during propagation [6]. This paper drew many ideas from these two papers.

More machine idea tricks is used in rumor detection, for example, attention mechanism is used in RNN model to detect rumors [7]. Besides, multimodal features is combined with an RNN model with attention mechanism [8].

3 Rumor Detection

In this part, I put forward two RNN models to detect rumors.

3.1 Structure of a Rumor

As mentioned above, to detect a rumor, I must proceed to extract features from the structure of rumors, here shows the structure in Figure 2.

As is shown, when a rumor is published as a source, other uses will forwarding this rumor, and that is how a rumor spreads.

Each forwarding has a time stamp, and therefore all forwarding texts can be sorted

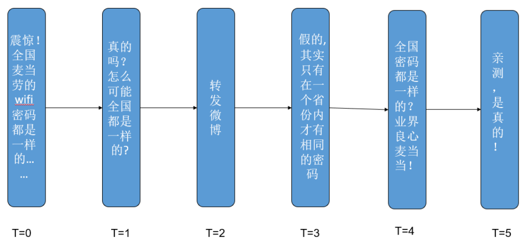

震惊！全国麦当劳的wifi密码都是一样的……
真的吗？怎么可能全国都是一样的?
转发微博
假的，其实只有在一个省份内才有相同的密码
全国密码都是一样的？业界良心麦当当！
亲测，是真的！
T=0
T=1
T=2
T=3
T=4
T=5

Figure 2: Structure of a Rumor

in a time series. Such a time series is perfect for applying RNN to do detection.

As can be seen, in this time series, there are texts expressing suspicious to the source like $T = 1$. Also, some forwarding might in turns confirm the source information like $T = 5$, which makes this rumor more confusing for normal users. Besides, some forwarding texts provides no meaningful information like $T = 2$ and $T = 4$. However, there might be some malicious forwarding texts like $T = 3$, which denied the source text, but then changed the meaning of the source and published another rumor. Normal users who read this ”correction” might more likely to trust $T = 3$ and spread this information more wildly than the source. Thus, $T = 3$ text is more like a rumor than the source, and which enhanced the propagation of a rumor.

All texts in a time series can be considered as a voting process, a user will express his own opinion on the source text, and judge if the source is a rumor. However, normal users can hardly judge is a text is a source because the perplexity of a rumor. In most cases, a user only shows a emotional tendency. A text forwarded at $T = n$ would express emotional tendency based on text at $T = n - 1$, or ideally, all texts from $T = 0$ to $T = n - 1$. Such emotional tendency will enhance or weaken the emotional tendency of the past texts. For example, when a user confirmed a former user's query, then the whole time series will become more rumor-like, and more people would ignore the query and believe the new text. When a user express suspicious or confused, or denying, or even confirming, a rumor's feature is getting more clear, and makes a rumor more rumor. Therefore, information contained in all texts in the time series can be used to judge if a text is a rumor.

3.2 Recurrent neural network

A recurrent neural network (RNN) is a class of neural networks where connections between nodes form a directed graph along a temporal sequence. Therefore, RNN can exhibit temporal dynamic behavior. When a new input comes, RNN can update its hidden state by former hidden state and the new input. This makes RNN applicable to variable length sequences of inputs like speech recognition and semantic recognition.

A classical RNN is like this: given input $(x_1, x_2, x_3, ..., x_T)$, RNN model would update the hidden state as $(h_1, h_2, h_3, ..., h_T)$, and also generate a output series $(o_1, o_2, o_3, ..., o_T)$.

There are three weight matrix in RNN as U, W, V. And here is how RNN update hidden state and generate output:

$$\begin{aligned} h_t &= \tanh\left(Ux_t + Wh_{t-1} + b\right) \\ o_t &= Vh_t + c \end{aligned}$$

However, the problem is that RNN might be faced with vanishing or exploding gradients, a classical RNN model can not capture the long-distance temporal dependencies using gradient based optimizing method, in other word, classical RNN is short-sighted. Therefore, some improved unit is put forward, which is called Long Short-Term Memory(LSTM), and a simplified cersion named as Gated Recurrent Unit(GRU). By adding such units into a RNN model, a model can learn much more information in a long-distance.

Different from classical RNN, LSTM maintains another cell named as memory cell c.

Here is how it updates.

$$
\begin{aligned}
i_t &= \sigma\left(x_t W_i + h_{t-1} U_i + c_{t-1} V_i\right) \\
f_t &= \sigma\left(x_t W_f + h_{t-1} U_f + c_{t-1} V_f\right) \\
\tilde{c}_t &= \tanh\left(x_t W_c + h_{t-1} U_c\right) \\
c_t &= f_t c_{t-1} + i_t \tilde{c}_t \\
o_t &= \sigma\left(x_t W_o + h_{t-1} U_o + c_t V_o\right) \\
h_t &= o_t \tanh\left(c_t\right)
\end{aligned}
$$

Here, σ is a sigmoid function, and f_t is defined as a forget gate, which determines how many information to be dropout, i_t is defined as the input gate, which determines how many new memory to be input into the memory cell. Then, the memory update itself by combining some past memory and new memory as c_t. Besides, output and hidden state is also updated.

Before introducing the model used in this job, I will clarify some details in my job.

Since all texts in a time series are sorted by time stamp, then they are perfectly fit to LSTM, however, time stamps can be sorted by order or reverse order. I name sort by order as top-down method, and reverse order as bottom-up method. In this job, I chose the bottom-up method. In this way, x_0 is the text publish latest, and the x_T is the source text. The reason that I chose bottom-up method is that, texts published earlier in the time series always has greater influence the the later, which indicates that many texts later is forwarded from the same text.

As can be seen in Figure 3, $T = 2, T = 3, T = 5$ are all forwarded from $T = 1$, therefore, when $T = 1$ is input, the memory cell and hidden state can update according to the input's semantic and former inputs' semantic. However, top-down method is also qualified in doing this job, but to keep consistent to the later job, I still chose bottom-up method.

3.3 Tree-GRU

As mentioned above, a text is be forwarded from a former microblog. In 3.2, I naively sorted all microblogs' text by time stamp.

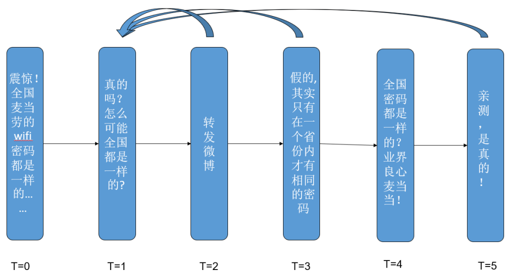

震惊！全国麦当劳的wifi密码都是一样的……
真的吗？怎么可能全国都是一样的?
转发微博
假的,其实只有在一个省份内才有相同的密码
全国密码都是一样的？业界良心麦当当！
亲测，是真的！
T=0
T=1
T=2
T=3
T=4
T=5

Figure 3: Bottom-up method

However, though this works in my experiment, but it is does not consistent with common sense: a micorblog's semantic information is most likely to corresponding to the microblog it forwarded from. For example, the user published microblog $T = n$ is forwarded from $T = m$, but this user might never read microblog $T = n - 1$, therefore, when updating hidden state and memory cell when inputting $T = m$, the model should only update based on $T = n$ but not $T = n - 1$. That means a model should know which microblogs are forwarded form the input microblog, and those not forwarded from input microblog should do no change to the hidden state and memory cell.

Therefore, the time series should be changed into a tree structure, with each node representing a microblog published in the propagation tree. If node A is the parent of node B, that means B is forwarded from A. For example, here shows a rumor tree in Figure 4.

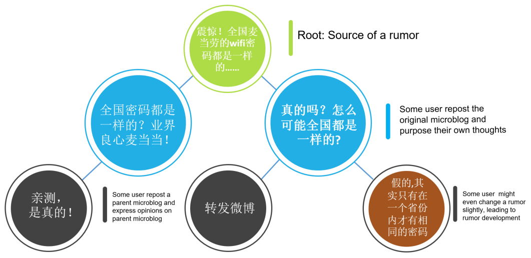

震惊！全国麦当劳的wifi密码都是一样的……
Root: Source of a rumor
全国密码都是一样的？业界良心麦当当！
真的吗？怎么可能全国都是一样的?
Some user repost the original microblog and purpose their own thoughts
亲测，是真的！
Some user repost a parent microblog and express opinions on parent microblog
转发微博
假的,其实只有在一个省份内才有相同的密码
Some user might even change a rumor slightly, leading to rumor development

Figure 4: Tree of a Rumor

Such a tree structure can not be fed into a

RNN model, therefore, a novel model is nedded.

In this part, I used a variant GRU model, which is named as tree-GRU. Different form classical GRU that fed with a time series, a tree-GRU sort all nodes by layers, and fed one layer at each time step, when updating, tree-GRU would check the parent-child relationship, and only update the state by input nodes' children.

Tree-GRU also applied bottom-up method. In the first step, tree-GRU would only get all leaf nodes in the deepest layer, and then update all states. Then in the second step, tree-GRU would get all nodes in the second deepest layer, and update corresponding the parent-child relationship. Here shows the difference in the formula.

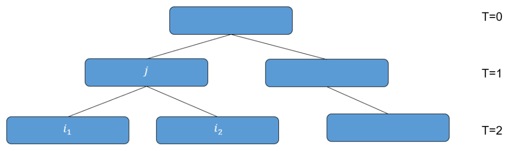


Figure 5: Bottom-Up Tree Model

$$
\begin{aligned}
h_i &= h_{i_1} + h_{i_2} \\
r_j &= \sigma\left(w_r x_j + U_r h_i\right) \\
z_j &= \sigma\left(w_z x_j + U_z h_i\right) \\
\widetilde{h}_j &= \tanh\left(w_h x_j + U_h\left(h_i * r_j\right)\right) \\
h_j &= \left(1 - z_j\right) * h_i + z_j * \widetilde{h}_j
\end{aligned}
$$

As can be seen, the only difference between original GRU and tree-GRU is how hidden state h is updated.

4 Detection of evolution

In this part, I put forward a mechanism to detect if some nodes in a propagation tree evolved. On this basis, I put forward a Visualization model of rumor propagation and evolution.

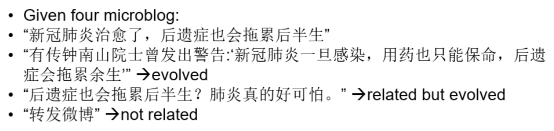
4.1 Definition

In today's social network, a rumor might changes while spreading, and becomes more confusing and hard to discern. And because different rumor has different shapes of propagation tree, the root might not be the most confusing and widespread. A evolved rumor indicates such a case. It is necessary to detect if a rumor(or a true news)

To clarify what a evolved rumor should be like, here shows the requirements for a evolved rumor:

- A evolved rumor should be the root of a sub-tree of the whole propagation tree of the source node.
- The sub-tree should be detected as a rumor.
- Evolved text should be similar with the source text in formal
- Evolved text should be much different from the source text in meaning.

Here is an example:



- Given four microblog:
- “新冠肺炎治愈了，后遗症也会拖累后半生”
- “有传钟南山院士曾发出警告:‘新冠肺炎一旦感染，用药也只能保命，后遗症会拖累余生’” →evolved
- “后遗症也会拖累后半生？肺炎真的好可怕。” →related but evolved
- “转发微博” →not related

Figure 6: An example of rumor evolution

The first text is the rumor source text, which is a lie without foundation, which is less convincing for normal users. However, the second text changed the original text as said by a medical expert, making the rumor more confusing for people. However, in most cases, texts in a propagation tree is usually like the third one, though related to the original text, but no change from the source. Besides, there are texts with no meaning in the propagation tree.

This indicates two problems: First, to detect evolution, a model should be able to check if a suspicious text has similar meaning to the source, because those texts with no meaning are not evolved. Second, a model should be able to detect if the meaning of a related text has changed from the source.

4.2 Distance metric

Because that this part of job is a unsupervised job, so the basic idea is to detect evolution based on distance.

Here defines distance D as the semantic distance to measure the similarity between two texts in semantic. Semantic analysis is a classical problem in natural language processing, and many tools can be used to measure the similarity in semantic. In this part, I chose LSTM to learn the distance D, which returns a float from zero to one, indicating how similar two texts is in semantic.

However, it is not enough to detect evolution only based on D. For example, some users might directly copy the text from the source and re-publish it. In this case, $D = 1$, but no evolution has happened.

Therefore, another distance metric J, which measures the distance between two text in form. This idea is based on that, a evolved text will always change the form of the source, while still be related and expressing similar meaning. To measure the similarity of two texts in form, I used the distance of two texts' deep learning features to represent the formal similarity.

When calculating J, Euclid distance is not a good choice. That is because, texts in a propagation can not fully distribute in the whole space. A simple explanation is that, all meaningless texts like "forward microblog" and some simple emoji will be more closer in form, and more likely to assemble in the same cluster, while those texts with similar texts are likely to assemble in another cluster. When naively use Euclid distance, J will more prefer those meaningless texts, however, what I want J to find are texts in the same cluster with source, but still far from the source to find latent evolved text.

To solve this problem, I chose to measure the distance J by the length of the shortest path by hoops, and I stipulate that each text can only be connected to the nearest k texts. In this metric, a related rumor with more hoops from the source is usually much different from the source in form, which fits in the requirement for a evolved text. And, for those unrelated texts, they will not benefit from the long distance in Euclid, and their J will not be much higher than those true evolved texts.

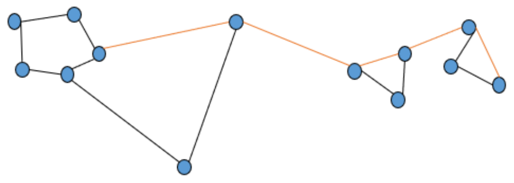


Figure 7: Distance of form

To find a text evolved, I combined these two distance metrics to decide if a text evolved. I define E_i as the evolution degree between rumor source and text i, which measures the confidence for text i to be a evolved node.

$$E_i = D_i + \lambda J_i$$

After generating deep learning features for each text, these features can be considered as projected onto a hyperplane. J_i indicates the distance from source to the text i, when constrained on the hyperplane. While D_i indicates the similarity in semantic, which can be considered as a distance in logistic. When J in constrained on the hyperplane, D is the distance beyond the hyperplane.

Object the hyperplane in a higher dimension, the dimension with logistic. This hyperplane is curly in logistic, or semantic. Therefore, this hyperplane shows like this.

In this figure, A is the source of a rumor, while B is close to A in semantic, however, the distance of form is also close to A, which indicates that B represents the same meaning as A, and no evolution happens. As for C, though C is also close to A in semantic, but the form distance from A to C along the hyperplane is much longer than A to B. This shows that C has similar, or at least related meaning as A, but C significantly changed the form of A, indicating that some malicious user might did some change to the source, which leading to an evolution.

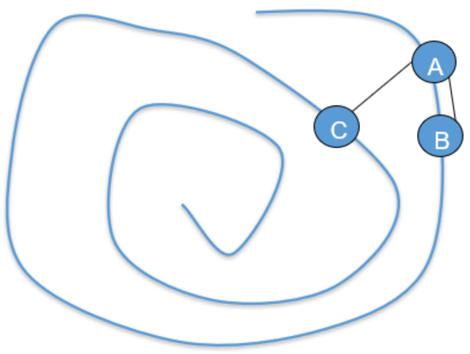


Figure 8: Hyperplane of all texts

5 Experiment

In this part, I will introduce how I implemented rumor detection model and evolution detection model, and the visualization model of a rumor propagation tree.

5.1 Rumor detection

5.1.1 Data preprocessing

In this part, I used the dataset from *http : //alt.qcri.org/ wgao/data/rumdect.zip*, which containing 2313 rumors and 2351 non-rumors.I split train and test by 60%:40%.

To establish a vocabulary, I removed all punctuation and emoji from all texts in the dataset. Then used jieba to split a text into a list of words.

In order to get a vocabulary that can represents all texts' most representative word, I used term frequency-inverse document frequency(tf-idf) to sort all words in the dataset, which is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Tf-idf contains two parts. Term frequency represents the frequency of a word, which is the raw count of a term in a dataset. Inverse document frequency is a measure of how much information the word provides, which checks if a word is common or rare across all documents.

$$td - idf(t, d, D) = tf(t, d) * idf(t, D)$$

By sorting all words by tf-idf, I get a vocabulary with size equal to 20000.

When transferring a text into a numerical representation, I set the max length of a text vector as 15, which is a relative best choice in my experiment.

In order to evaluate the performance of models in early detection, I set the length of a tree propagation as $150, 300, 800$.

5.1.2 LSTM model

I implemented a LSTM model to detect a rumor at first. To extract features of words, I added an embedding layer to project a word into a 8-dim vector. Then fed data into a LSTM model, which contains 32 units. In order to avoid overfitting, dropout is set as 0.2. To get the prediction of the input, a fully connected layer is added after LSTM layer, and applied sigmiod function to output a probability.

Here shows the structure of LSTM model.

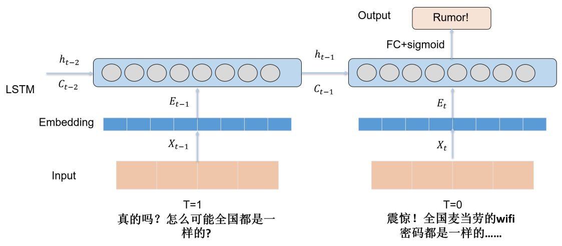


Figure 9: Structure of LSTM model

Here shows the performance of LSTM model.

propagation length	accuracy
150	0.8574
300	0.8715
800	0.8598

As can be seen, the accuracy is highest when maximum propagation length is 300, which is an acceptable length for early detection. However, when propagation length is 800, the accuracy decreased, that is because most rumors in this dataset is shorter than 800, which means that too many zeros were padded into a rumor, which cases a waste of time and influenced the performance.

5.1.3 Tree-GRU model

To implement a Tree-GRU, I construct a tree for all texts in a propagation tree according to the forwarding relationship as parent and child. For each node, maintain its text and hidden state h, while the text can be transferred in to a vector by the same vocabulary and the embedding layer trained in LSTM detection model.

For each time doing forward propagation, just traverse all nodes by layers from leaf to the root, and compute hidden state h_j for node j based on all node j's child nodes' hidden state $(h_{i1}, h_{i2}, ..., h_{in})$. In tree-GRU, different from a classical GRU, which only has one predecessor input, but many predecessor hidden states with variant numbers.

Here shows the structure in tree-GRU.

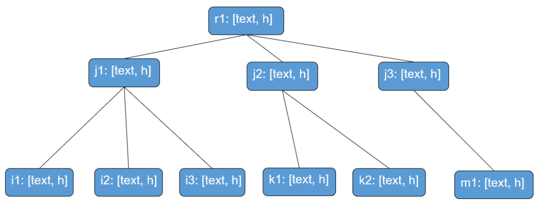


Figure 10: Structure of tree-GRU

Here, there are two strategies to combine $(h_{i1}, h_{i2}, ..., h_{in})$ to get h_i: directly sum or weighted sum.

- $h_i = \sum_{k=1}^{n} h_{ik}$
- $h_i = \sum_{k=1}^{n} w_{ik} h_{ik}$

For the first strategy, which assume that all child nodes are equally important. In this case, the hidden state used to update is the sum of all child nodes' hidden state.

Here shows the performance of the first strategy:

propagation length	accuracy
150	0.8912
300	0.9182
800	0.9201

However, naively sum all hidden state is not a good choice, because in a propagation tree, the sub-tree corresponding to a child node are usually with different size, that means, the importance of a child node's hidden state is usually different, therefore, a better choice is to consider different child node's weight. So here defines the weight of a child based on its sub-tree's size.

$$w_{i,k} = n * \frac{S_{i,k}}{\sum_{k'=1}^{n} S_{i,k'}}$$

In this case, the larger a sub-tree is, then the more impact this child node can make to the parent node's hidden state.

Here shows the performance of the second strategy:

propagation length	accuracy
150	0.8989
300	0.9214
800	0.9171

As can be seen in this table, the second strategy out performs the first one when the propagation length is 150 and 300, but worse than the first one in 800.

However, this is an acceptable result. First, early detection is usually more important in rumor detection, and weighted strategy performs better. Second, because about 65% rumors in the dataset is shorter than 800, so this case is of no reference value.

5.2 Evolution detection

In this part, I will introduce how I implemented evolution detection.

5.2.1 Semantic distance

To measure how similar two texts are in semantic, I trained a LSTM model to do this job.

At first, I took advantage of an existing tool named as *text2vec*, which has a function to measure how similar two texts is. Take Figure

6 for example, *text2vec* indicates that the similarity for the last three texts to the first one is $0.9412, 0.8098, 0.5039$.

However, this tools mostly measures if two texts has similar topic and talking about the same thing. However, what I want is not only find texts that talking about the same thing, but expressing almost the same meanings as the source. So this tool in not enough. Here shows an example in semantic similarity.

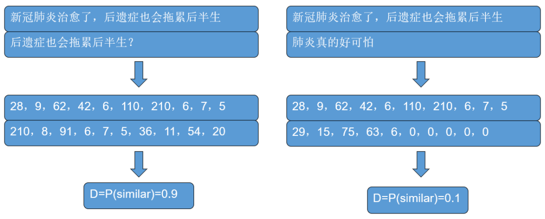
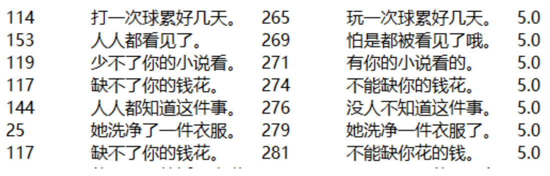


Figure 11: Semantic similarity

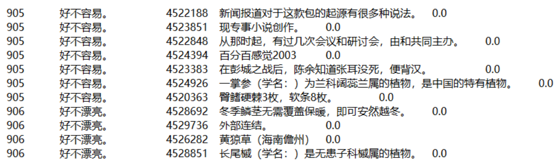
In this part, I took advantage of a dataset from *https://github.com/IAdmireu/ChineseSTS*, which can be used to learn the semantic similarity.

Here shows an example in the dataset.



114	打一次球累好几天。	265	玩一次球累好几天。	5.0
153	人人都看见了。	269	怕是都被看见了哦。	5.0
119	少不了你的小说看。	271	有你的小说看的。	5.0
117	缺不了你的钱花。	274	不能缺你的钱花。	5.0
144	人人都知道这件事。	276	没人不知道这件事。	5.0
25	她洗净了一件衣服。	279	她洗净一件衣服了。	5.0
117	缺不了你的钱花。	281	不能缺你花的钱。	5.0

Figure 12: An example of similar semantic



905	好不容易。	4522188	新闻报道对于这款包的起源有很多种说法。	0.0
905	好不容易。	4523851	现专事小说创作。	0.0
905	好不容易。	4522848	从那时起，有过几次会议和研讨会，由和共同主办。	0.0
905	好不容易。	4524394	百分百感觉2003	0.0
905	好不容易。	4523383	在彭城之战后，陈余知道张耳没死，便背汉。	0.0
905	好不容易。	4524926	一掌参（学名：）为兰科阔蕊兰属的植物，是中国的特有植物。	0.0
905	好不容易。	4520363	臀鳍硬棘3枚，软条8枚。	0.0
906	好不漂亮。	4528692	冬季鳞茎无需覆盖保暖，即可安然越冬。	0.0
906	好不漂亮。	4529736	外部连结。	0.0
906	好不漂亮。	4526282	黄猄草（海南儋州）	0.0
906	好不漂亮。	4528851	长尾槭（学名：）是无患子科槭属的植物。	0.0

Figure 13: An example of different semantic

By training a classical LSTM model with embedding layer, LSTM layer and fully connected layer, the semantic model can return a prediction to measure how two texts similar in semantic, with input as a $2 * max\ length$ matrix generated from a vocabulary.

By training such a model. I get the MSE loss as 0.00738 on average for each sample in test set. And the prediction of the examples in Figure 6 is $0.9158, 0.7218, 0.0756$, which perfectly fits in this job.

5.2.2 Form distance

To learn the form distance J between two text, this part is all based on such a theory: If two texts are similar in form, then the distance between their deep learning features should also be close.

The reason that I chose deep learning features to compute distance is that, in most cases, texts in a propagation are usually much shorter than the source, thus, if directly counts how many words show up in both texts or compute the distance after embedding layer, too many zeros in a short text will impact the distance.

To extract the deep learning features, or in other word, to project all texts on to a hyperplane, I trained a classical auto-encoder to do this job.

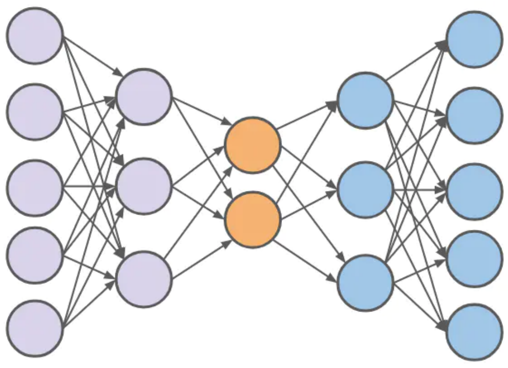


Figure 14: Auto-encoder

However, different from the usual application of auto-encoder as a dimension reducing tool, in this part, I project texts into high dimension, and thus extract deep leaning features with dimension as 128.

After such projection, all texts can be considered as a point on a hyperplane. As mentioned above, it is not a good idea to calculate the Euclid distance between texts. Therefore, I assume that a point can only be connected to the nearest 6 points, and replace Euclid distance with the shortest path's hoops.

Here use the same example in Figure 6, the form distance is $3, 5, 17$.

However, in some rumor's propagation tree, not all texts can be connected by the nearest-6-point strategy, and thus texts might form into independent clusters, and the distance can not be calculated. In this case, I connected two texts in each cluster with the least Euclid distance.

5.2.3 Evolution detection

To combine above two distance metric, here defines E_i as the evolution degree between rumor source and node i.

$$E_i = D_i + \lambda J_i$$

In this part, λ is a hyper-parameter, and should be carefully chose. I tried a varies of values, and at last I chose $\lambda = 0.15$ to get a better performance.

By sorting all texts in a propagation tree by E, I can check which text in this tree are most likely to be evolved.

5.3 Visualization of a rumor propagation tree

In this part, I will combine the rumor detection part and evolution detection part to put forward a visualization model

As mentioned above, a evolved rumor will enhance the propagation of the rumor, which means that such a evolved rumor should also be a rumor when only checking its sub-tree. Therefore, to comprehensively evaluate a text in a propagation tree, I should consider the confidence for a text to be a rumor, and the probability for it to be a evolved rumor, and the propagation scope of it.

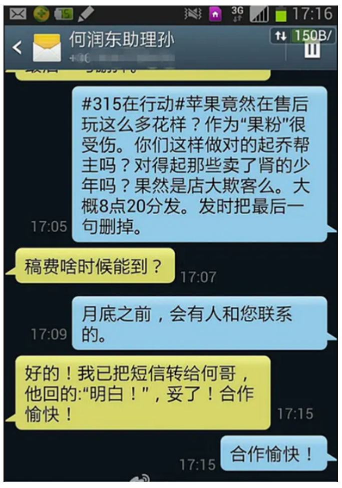
Here gives some definition:

- C_i: Confidence for node i to be a rumor source.
- N_i: How many users forwarded this microblog.
- E_i: Evolution degree between rumor source and node i.

Based on the definition above, here defines the metric of propagation contribution P for a text in propagation tree:

$$P_i = C_i * E_i * \log N_i$$

Based on all the metric above, I put forward a model to visualize the propagation of a rumor. Here shows an example:



#315在行动#苹果竟然在售后玩这么多花样？作为"果粉"很受伤。你们这样做对的起乔帮主吗？对得起那些卖了肾的少年吗？果然是店大欺客么。大概8点20分发。发时把最后一句删掉。 17:05

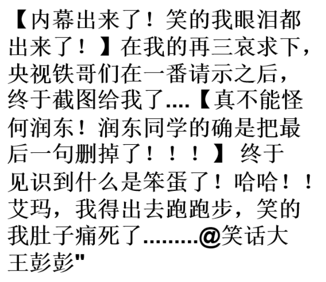
稿费啥时候能到？ 17:07

月底之前，会有人和您联系的。 17:09

好的！我已把短信转给何哥，他回的:"明白！"，妥了！合作愉快！ 17:15

合作愉快！ 17:15

Figure 15: An example of rumor



【内幕出来了！笑的我眼泪都出来了！】在我的再三哀求下，央视铁哥们在一番请示之后，终于截图给我了....【真不能怪何润东！润东同学的确是把最后一句删掉了！！！】终于见识到什么是笨蛋了！哈哈！！艾玛，我得出去跑跑步，笑的我肚子痛死了.........@笑话大王彭彭"

Figure 16: An example of rumor

This is a rumor years ago, it says that He was paid to give malicious negative feedback on iPhone.

Here is the visualization result of this rumor:

In this result, each bubble represents a microblog in a propagation tree. And the size of

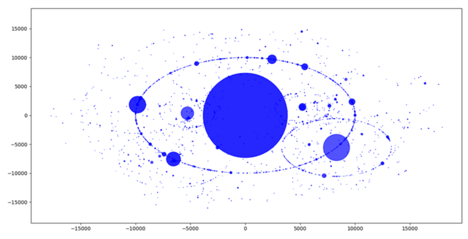


Figure 17: Visualization for a propagation tree

a bubble represents the propagation contribution of this text in the rumor propagation tree. And the transparency of a bubble represents the evolution degree of each text. For each text in the tree, all its child would surround it and forms into a circle. And the source text is right in the center of the picture.

Here shows some bubbles' text in this tree:

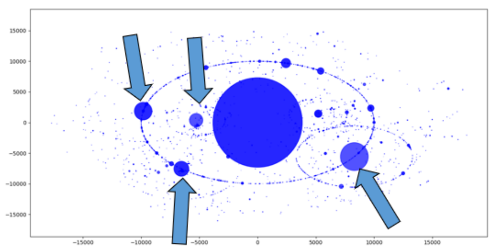
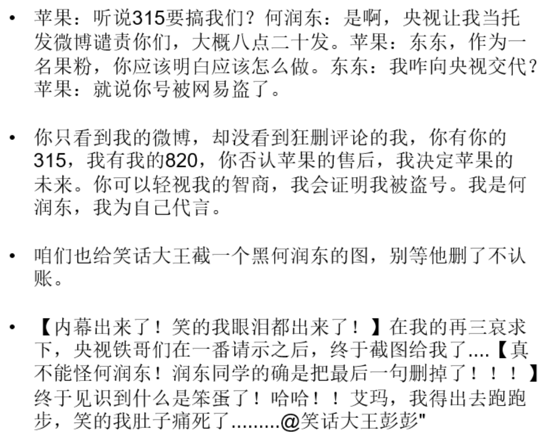


Figure 18: Some bubbles

As can be seen, the first two texts in this example shows more obvious evolution, which saying that He's account was hacked, and showing a sense of humor. While, in the last two examples, they are either not very related to the source, or totally the same as the source, which should be of lower evolution degree. As can be seen, their bubbles in the result is more close to the transparent.

6 Conclusion

In this project, I put forward two model to detect rumor based on RNN. The first model is based on LTSM, and the second is a novel



- 苹果：听说315要搞我们？何润东：是啊，央视让我当托发微博谴责你们，大概八点二十发。苹果：东东，作为一名果粉，你应该明白应该怎么做。东东：我咋向央视交代？苹果：就说你号被网易盗了。

- 你只看到我的微博，却没看到狂删评论的我，你有你的315，我有我的820，你否认苹果的售后，我决定苹果的未来。你可以轻视我的智商，我会证明我被盗号。我是何润东，我为自己代言。

- 咱们也给笑话大王截一个黑何润东的图，别等他删了不认账。

- 【内幕出来了！笑的我眼泪都出来了！】在我的再三哀求下，央视铁哥们在一番请示之后，终于截图给我了....【真不能怪何润东！润东同学的确是把最后一句删掉了！！！】终于见识到什么是笨蛋了！哈哈！！艾玛，我得出去跑跑步，笑的我肚子痛死了.........@笑话大王彭彭"

Figure 19: Texts of some bubbles

GRU, which performs in a tree structure. Results shows that tree-GRU performs better than the sequencial LSTM. Besides that, I used two distance metric to measure the similarity between two texts. I put forward a learning method to measure the semantic similarity, and a shortest-path-based method to measure the form similarity between each projected text. By combining these two distance metric, and considering the propagation scope and rumor index, I put forward a model to visualize the propagation of a rumor.

References

[1] S. Kwon, M. Cha, and K. Jung, "Rumor detection over varying time windows," *PloS one*, vol. 12, no. 1, 2017.

[2] T. Takahashi and N. Igata, "Rumor detection on twitter," in *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, 2012, pp. 452–457.

[3] Q. Zhang, S. Zhang, J. Dong, J. Xiong, and X. Cheng, "Automatic detection of rumor on social network," in *Natural Language Processing and Chinese Computing*. Springer, 2015, pp. 113–122.

[4] F. Yang, Y. Liu, X. Yu, and M. Yang, "Automatic detection of rumor on sina weibo," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, 2012, pp. 1–7.

[5] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumors from microblogs with recurrent neural networks," 2016.

[6] J. Ma, W. Gao, and K.-F. Wong, "Rumor detection on twitter with tree-structured recursive neural networks." Association for Computational Linguistics, 2018.

[7] T. Chen, X. Li, H. Yin, and J. Zhang, "Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 40–52.

[8] Z. Jin, J. Cao, H. Guo, Y. Zhang, and J. Luo, "Multimodal fusion with recurrent neural networks for rumor detection on microblogs," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 795–816. [Online]. Available: https://doi.org/10.1145/3123266.3123454