# COMP4121 Project

## Multimodal Fake News Online Detection

Tianwei

Z5305298

## Abstract

This report records what I have learned from an experience of joining in Dr. Jiaojiao Jiang’s research on the topic “Multimodal Fake News Online Detection”. The report will start from basic knowledge of machine learning to recent scientific research achievement.

## Table of Content

# Introduction

With the development of network and technology, great changes have taken place in the way people communicate. One can get and exchange information more easily than before. More people learn what is happening in the world online and express his or her own thought. Social media can spread news easily. However, this also makes the spread of false news overwhelming. people often encounter a mix of truth and fake news. This can lead to serious results, such as loss of reputation from the government and confusion of the public. In order to prevent the deterioration of this phenomenon, it is necessary and urgent to have a tool to help society to distinguish fake news.

Currently the methods of detecting fake news are mainly based on the model of machine learning and neural network and have been proved to be effective. With the development of computer science, the model is also being optimized, and the accuracy and efficiency of identifying fake news are also improving. This report will start from the basic of neuron network to its application.

# Basic Knowledge

In this section, we introduce some basic knowledge that are already applied in current fake news detection. We will need the technics in our multimodal online fake news detection.

## Machine Learning

Machine learning is the core of artificial intelligence. The definition is that computers infer the unknown by learning the known conditions, just as human's learning process. For example, how can we pick the "sweetest" mangos in a shop? As a human, one may think that the bigger the mango, the sweeter it is, so he will choose the bigger mangos in the store as much as possible. But after buying them many times, he found that it was not always right. In addition, the redder the mango is, the sweeter it is. So, he began to buy the red and big mangos in the store as much as possible. After buying them many times, he found that the softer the mango, the sweeter it was. So, he began to buy soft, red and big mangoes in the store. This is a series of "learning" process, which is about learning how to buy the sweetest mango. When it comes to machine learning, we give the mango samples and the corresponding sweetness to the computer, and the computer will draw a model. Assume it is fact that the bigger, redder and softer the mango, the sweeter it will be. The model contains the features those sweet mangos have, which are "big, red and soft". When giving mango samples to the computer, the machine can predict the sweetness of mango based on the features. A graphical explanation is provided below.

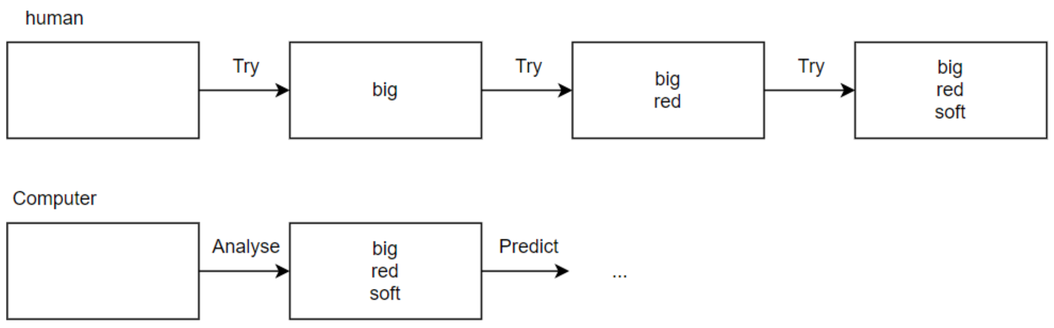


**Figure 1: A graphical explanation of example for machine learning. The blank describes the feature sweet mangos have.**

## Deep Learning

Deep learning is a subfield of machine learning. “Deep” refers to the application of layers. Specifically, deep learning uses Artificial neural network (ANN) to learn complex features from basic features of original input. Applying various kinds of ANNs enables machine to learned different features.

## Artificial Neural Network

Artificial neural network, known as ANN, is a network composed of computing units, known as neurons. An ANN is consisted of three layers, input layer, hidden layer and output layer (Castrounis, 2020). On each neuron there is a specific activate function, which will be introduced in later section. The connection between each two neurons represents a weighted value for the signal passing through the connection, which is called weight. Neural network simulates human memory in this way. Artificial neural network combines the understanding of biological neural network with mathematical statistical model and is realized with the help of mathematical statistical tools. The network itself is usually the approximation of some algorithm or function in nature, or the expression of a logical strategy.

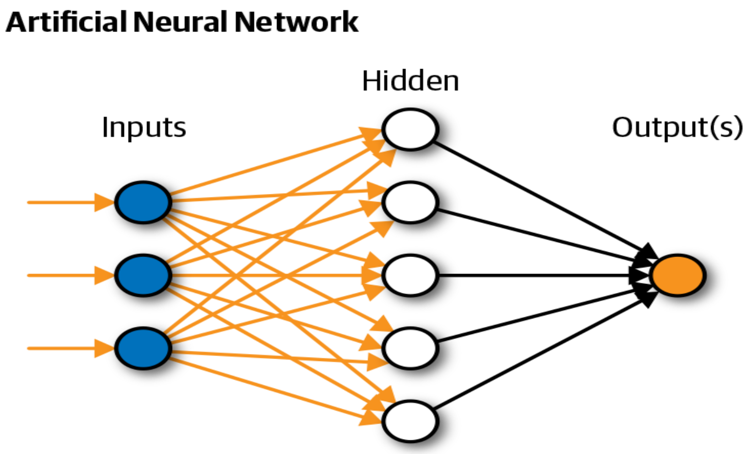


**Figure 2: The structure of Artificial neural network** (Castrounis, 2020)

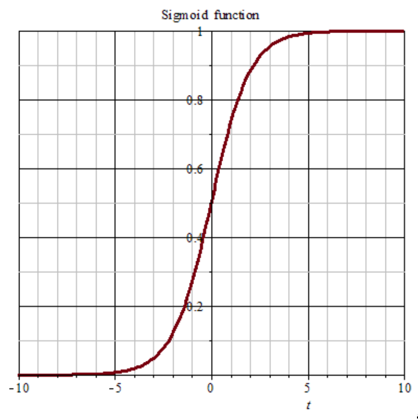
## Activate Function

Activate function is functions in neurons that process the data that is going to be passed from its neuron. That means, every value before passing from one neuron to another neuron need to be process by an activate function. The reason why we need activate functions is that activate functions make the message passing process nonlinear. Assume we do not have activate functions, then the process of message passing from neurons to neurons can be summarized to linear combinations and transformations. Besides, in reality the process of our brain directing us to make decision is not linear. Thus, activate functions aim to help ANN to simulate natural neuron networks.

Nowadays there are various kinds of activate functions. This report will introduce some of them that will be used in later section.

- **Sigmoid**

$$f(z) = \frac{1}{1+e^{-z}}$$



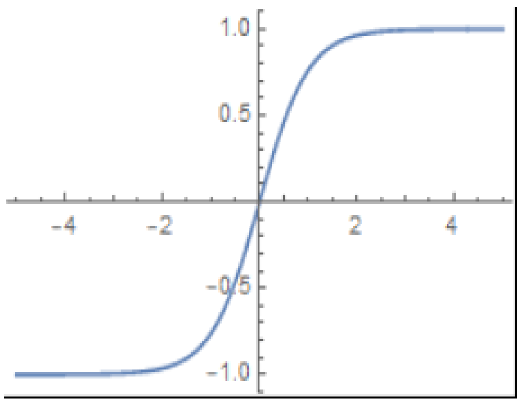


**Figure 3: sigmoid function** **Figure 4: tanh function**

- **Tanh**

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- **Relu**

$$Relu = \max(0, x)$$

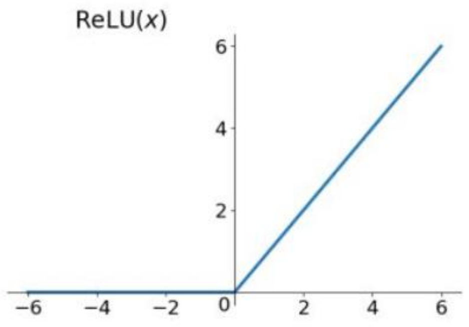


**Figure 5: Relu function**

Every activate function has its advantages and disadvantages. Using different activate functions can produce different result and consume different computation power.

## Convolutional Neural Networks (CNN)

CNN is one of iconic learning algorithms of deep learning. CNN is used to detect “patterns” of images.

A pattern of an image is a feature of the image. It can be specialized in different form. For example, to recognize a hand-write number, a CNN might have to recognize patterns such as horizontal lines, vertical lines, circles semi-circles and so on. To recognize a hand-write 4, the network has to recognize a horizontal line and two vertical lines, or a horizontal line, a vertical line and a slash. To recognize a hand-write 5, the network has to recognize a horizontal line, a vertical line and a semi-circle.

So how to achieve this? CNN does recognition by filters, which is a specified matrix. After doing convolution to the filters and input images, CNN can obtain the information of that pattern. For hand-write numbers, if the input is a hand-write 4, after convoluted with filter for vertical lines, the output will only contain two vertical lines. By this way CNN can learn whatever patterns we want it to learn. Therefore, CNN has some advantages that traditional technologies do not have: good fault tolerance, parallel processing and self-learning ability. It can deal with problems in the case of complex environmental information, unclear background knowledge and unclear reasoning rules. It allows samples to have large defects and distortions, fast running speed, good adaptive performance and high resolution.

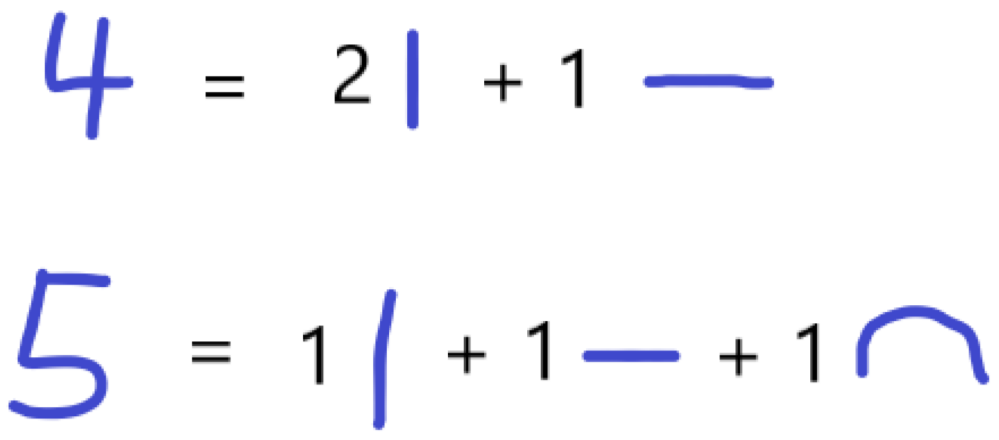


**Figure 6: patterns of hand- write numbers in CNN**

## Pooling

Pooling is an important concept in CNN. Pooling simplifies the input and remain the most feature of input. There are several advantages of pooling.

- **Reduce the size of input**
  After pooling, we compress the input to smaller size with almost same features. In this way we can put less burden on the following layers and save the computational power.
- **translation invariance**
  sometimes we want to adjust our input, such as do translation to the input. Because pooling constantly abstracts the features of the region and does not care about the location, pooling increases the translation invariance to a certain extent.
- **Avoid over-fitting**
  Pooling can help avoid over-fitting. Sometimes we do not want the complete input because of over-fitting. Pooling can help us to delete redundant information in the input.

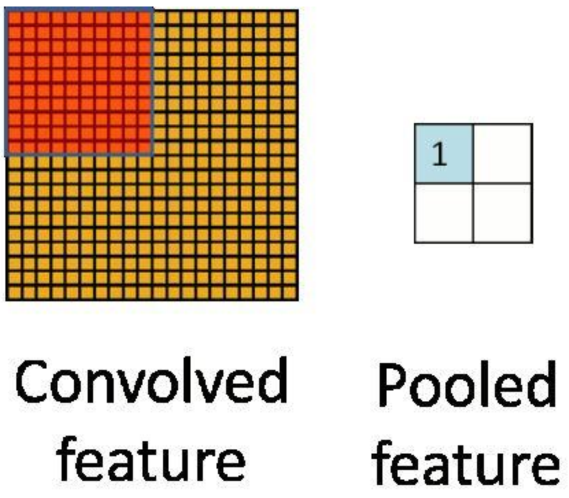


**Figure 7: Pooling overview**

## K-max Pooling

This section will introduce max pooling and k-max pooling. Max pooling keeps the biggest number and discard all other numbers in the region. For example, in the below figure, the filter size is 2 by 2. The biggest number is 19 in the top left block, 86 in the top right block is 86, 97 in the bottom left block and 60 in the bottom right block. Therefore, the output after max pooling process is 19, 86, 97 and 60. The remaining numbers are discarded as they are small, which means not important.

k-max pooling is slightly different from max pooling. While max pooling gets rid of too many points, k-max pooling keeps all values up to k$^{th}$ large number. In this way we can keep more features of the input.

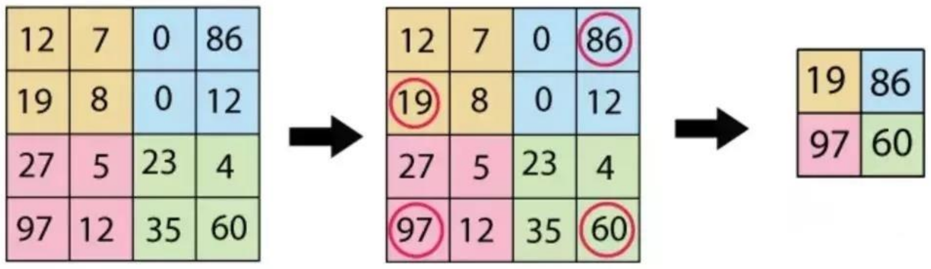


**Figure 8: max pooling**

## BM25

BM25 is an algorithm used to calculate the similarity between query and document in information retrieval field. BM25 mainly consider three aspects:

- **Relevance between each word *t* and document *d***
- **Similarity between word t and query**

- **Weight of each word**

  The general formula of BM25 is as follows:

$$score(Q,d) = \sum_{i}^{n} W_i R(q_i,d) \quad (1)$$

where $Q$ represent a query. $q_i$ is a word $i$ in the query. $d$ is the document. $W_i$ is the weight of word $i$, which is term frequency–inverse document frequency, or IDF in short. We can get $W_i$ by following equation:

$$IDF(q_i) = \log\frac{N - df_i + 0.5}{df_i + 0.5} \quad (2)$$

Where $N$ represents the number of all documents in the index, and $df_i$ is the number of documents containing $q_i$. According to the function of IDF, for a $q_i$, the more documents containing $q_i$, the less important the $q_i$, or the lower its discriminative power is, the smaller the IDF. Therefore, IDF can be used to describe the similarity between $q_i$ and documents.

In terms of $R(q_i,d)$, the design of BM25 is based on an important discovery: the relationship between word frequency and relevance is nonlinear, that is, the relevance score of each word to the document will not exceed a specific threshold. When the number of words reaches a threshold, the impact will not increase linearly, and this threshold is related to the document itself. Therefore, when describing the similarity between words and documents, BM25 is designed as follows:

$$S(q_i,d) = \frac{(k_1+1)tf_{td}}{K + tf_{td}} \quad (3)$$

$$K = k_1\left(1 - b + b\frac{L_d}{L_{avg}}\right) \quad (4)$$

where $tf_{td}$ is the word frequency of word $t$ in document $d$. $L_d$ is the length of document $d$. $L_{avg}$ is the average length of all documents. The variable $k_1$ is a positive parameter used to normalize the range of word frequency of articles. When $k_1 = 0$, it is a binary model, the impact of word frequency become linear. $b$ is another adjustable parameter ($0 < b < 1$). It determines how large the impact of information expressed by document length can be: when $b$ is 1, it completely uses the document length to weigh the weight of words. If $b$ is 0, it does not consider the document length.

When the query is long enough, we also need to describe the weight between the word and the query as follows. This is not required for short queries.

$$S(q_i,Q) = \frac{(k_2+1)tf_{tq}}{k_2 + tf_{tq}} \quad (5)$$

where $tf_{tq}$ represents the word frequency of word $t$ in query, and $k_2$ is an adjustable parameter to correct the word frequency range in query.

Therefore, we can derive the final formula of BM25 as follow:

$$score(Q,d) = \sum_{t\in q}\left[\log\frac{N}{df_t}\right]\frac{(k_1+1)tf_{td}}{k_1\left((1-b)+b\left(\frac{L_d}{L_{avg}}\right)\right)+tf_{td}}\frac{(k_2+1)tf_{tq}}{k_2+tf_{tq}} \quad (6)$$

# Current techniques

Fake news detection system has been developed for a period. Some scientific research achievements are worth learning from, such as Knowledge-driven Multimodal Graph Convolutional Network (KMGCN) model (Wang, Qian, Hu, Fang, & Xu, 2020), EMNLP2020 (Vo & Lee, Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News, 2020), and all kinds of other computational modules (Zhou et al., 2020). In this section, we are going to discuss two techniques in detail, which will be applied in our future research.

## EMNLP2020

In this section, we will discuss about a framework called EMNLP2020. The main idea of this framework is fact-checking articles (FC-articles). FC-articles are articles that comment on the target news, including confirming, supporting, debunking, refuting and so on. If most of FC-articles are against the news, the news is likely false, and vice versa. They are the key to evaluate how reliable a news is. Thus, this model aims to solve one main problem: How to find out and rank the related FC-articles.

### Input

There are many ways to spread news. In previous work (Wang, Qian, Hu, Fang, & Xu, 2020), only text is mostly considered. EMNLP2020 use both text and images of original tweets as input, which probably enhance the performance.

### Framework

To retrieve and rank FC-articles, EMNLP2020 framework has two parts: finding FC-articles and ranking FC-articles. To find FC-articles, a possible way is using BM25. Then the paper designed a MAN (Multimodal Attention Network) to rank the FC-articles. Specifically, we denote the original tweet as $q$ and FC-articles as $d$. Then we derive a function $f(q, d)$ to determine how relevant $q$ and $d$ is.

#### Multimodal Attention Network (MAN)

MAN has four components: (1) projection layers, (2) textual matching layer (3) visual matching layer and (4) unifying textual and visual information. The whole framework is presented in the form of diagram in figure 9.

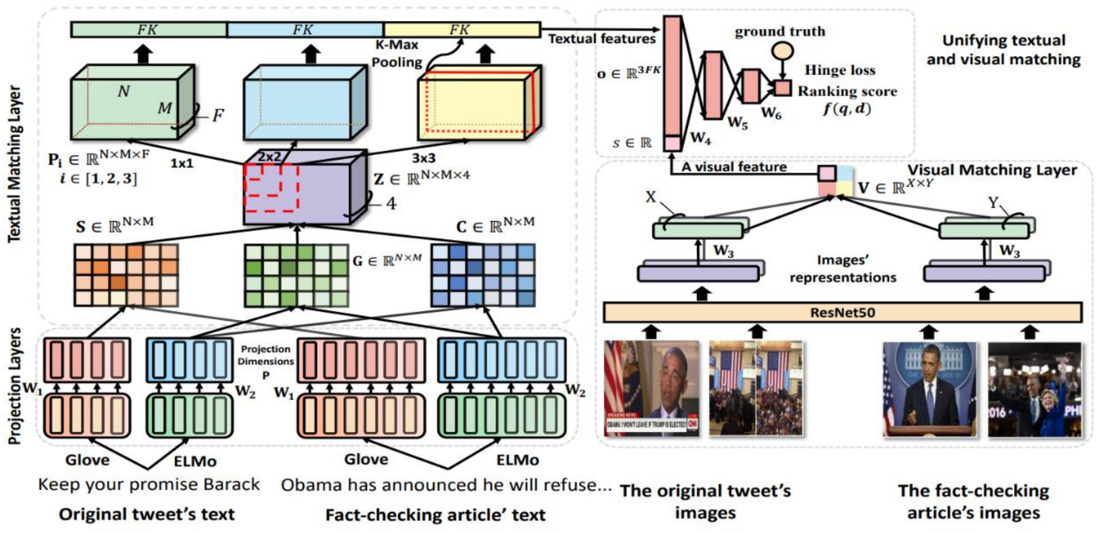

Textual Matching Layer
Projection Layers
FK
K-Max Pooling
Textual features
$o \in \mathbb{R}^{3FK}$
$s \in \mathbb{R}$
ground truth
Hinge loss
Ranking score $f(q,d)$
Unifying textual and visual matching
$P_i \in \mathbb{R}^{N\times M\times F}$, $i \in [1,2,3]$
$Z \in \mathbb{R}^{N\times M\times 4}$
$S \in \mathbb{R}^{N\times M}$
$G \in \mathbb{R}^{N\times M}$
$C \in \mathbb{R}^{N\times M}$
A visual feature
Visual Matching Layer
$V \in \mathbb{R}^{X\times Y}$
Images' representations
ResNet50
Projection Dimensions P
Glove
ELMo
Keep your promise Barack
Obama has announced he will refuse...
Original tweet's text
Fact-checking article' text
The original tweet's images
The fact-checking article's images

**Figure 9: MAN model overview**

## Projection Layers

Projection layers consist of two embedding layers: Glove embeddings (Pennington, Socher, & Manning, 2014) for relationship between word and word, and ELMo embeddings for relationship of context of words.

### Glove Embeddings

Glove embeddings is a mapping model that map words into word vectors that keeps fine-grained semantic and syntactic regularities to a great extent. It focuses on preserving the feature of words. Word in original tweet (denoted as $w_i^q$ for word $i$) and potential FC-articles (denoted as $w_j^d$ for word j) are mapped into vectors $\mathbf{t} \in \mathbb{R}^{\mathbf{300}}$ by fixed Glove embeddings. Then the vectors are projected into another vector $\mathbf{g} \in \mathbb{R}^P$ by following equation.

$$\mathbf{g} = tanh(\mathbf{W_1} \cdot \mathbf{t} + \mathbf{b_1}) \quad (7)$$

where $\mathbf{W_1} \in \mathbb{R}^{P\times 300}$ , $\mathbf{b_1} \in \mathbb{R}^P$. $P$ is projection dimensions. After going through the linear layer, we denote $\mathbf{g}_i^q \in \mathbb{R}^P$ and $\mathbf{g}_j^d \in \mathbb{R}^P$ as representations of word $w_i^q$ and word $w_j^d$, respectively.

### ELMo Embeddings

To make up for Glove embeddings' shortcoming, ELMo embeddings is applied to reflex context of words in queries and articles. ELMo embeddings is a word context representation which models both complex characteristics of word use (e.g., syntax and semantics), and how these uses (i.e., to model polysemy). Words are similarly mapped and projected as how Glove does. $w_i^d$ and $w_j^q$ are mapped into vector $\boldsymbol{l} \in \mathbb{R}^{\mathbf{1024}}$ by fixed ELMo embeddings, then be projected into $\mathbf{h} \in \mathbb{R}^P$ by following equation.

$$\mathbf{h} = tanh(\mathbf{W_2} \cdot \boldsymbol{l} + \mathbf{b_2}) \quad (8)$$

where $\mathbf{W_2} \in \mathbb{R}^{P\times 1024}$ , $\mathbf{b_2} \in \mathbb{R}^P$. $P$ is projection dimensions. After going through the linear layer, we denote $\mathbf{h}_i^q \in \mathbb{R}^P$ and $\mathbf{h}_j^d \in \mathbb{R}^P$ as contextual representations of word $w_i^q$ and word $w_j^d$, respectively.

## Textual Matching Layer

This layer contains three feature matrices: Glove embeddings interactions matrix, attended interaction matrix and contextual word embedding interactions matrix. Then we input them to convolution neural networks (CNNs) for feature extraction.

### Glove Embeddings Interactions

To represent relationships between words accurately, we apply cosine similarity to vector projections. To evaluate the relevance between original tweet and an article, we calculate the number of overlapping words or similar words between them. Specifically, for word $i$ from original tweet and word $j$ from potential AC-article, we derive a matrix $\mathbf{S} \in \mathbb{R}^{N\times M}$ as shown in Eq.3.

$$\mathbf{S}_{ij} = \frac{\mathbf{g}_i^{\mathrm{qT}} \cdot \mathbf{g}_j^{\mathrm{d}}}{||\mathbf{g}_i^{\mathrm{qT}}|| \times ||\mathbf{g}_j^{\mathrm{d}}||}, \mathrm{i} = 1 \ldots \mathrm{N}, \mathrm{j} = 1 \ldots \mathrm{M} \quad (9)$$

Generally speaking, the more similar word $i$ and word $j$ is, the smaller the angle between $\mathrm{g}_i^q$ and $\mathrm{g}_j^{\mathrm{d}}$ is, the bigger $\mathbf{S}_{ij}$ is. Then matrix $\mathbf{S}$ is used as an input of CNNs to determine whether two articles are relevant in a higher level.

### Attended Interaction Matrix

This matrix is designed to reduce the errors caused by the different semantics of words in different contexts. Thus, we need to calculate $\mathbf{G}_{ij}$ by following equation, that is how dissimilar $w_i^d$ and $w_j^q$ are in terms of their context.

$$\mathbf{G}_{ij} = 2 \times \sigma(-||\mathbf{h}_i^q - \mathbf{h}_j^d||), \mathrm{i} = 1 \ldots \mathrm{N}, \quad \mathrm{j} = 1 \ldots \mathrm{M} \quad (10)$$

where $\sigma(.)$ is a sigmoid function. $\mathbf{G}_{ij}$ is Euclidean distance between words' contextual representations, which is $\mathbf{h}$. Since Euclidean distance is non negative, $\sigma(-||\mathbf{h}_i^q - \mathbf{h}_j^d||)$ will be in $(0, 0.5]$ and $\mathbf{G}_{ij}$ will be in $(0, 1]$. We correct $\mathbf{S}_{ij}$ with $\mathbf{G}_{ij}$ by following equation:

$$\mathbf{A}_{ij} = \mathbf{S}_{ij} \times \mathbf{G}_{ij}, \mathrm{i} = 1 \ldots \mathrm{N}, \mathrm{j} = 1 \ldots \mathrm{M} \quad (11)$$

If the semantic meaning of $w_i^q$ and $w_j^d$ in their context are similar, the Euclidean distance between $\mathbf{h}_i^q$ and $\mathbf{h}_j^d$ is small, and $\mathbf{G}_{ij}$ would be close to 1, which means $\mathbf{S}_{ij}$ is close to $\mathbf{A}_{ij}$. However, in the opposite situation, $\mathbf{G}_{ij}$ would be close to 0, which $\mathbf{A}_{ij}$ is close to 0. In this way we decrease the effect of $\mathbf{A}_{ij}$ to the whole matrix when the semantics of two words are very different in different contexts.

## Contextual Word Embedding Interactions

Matrix $\mathbf{C}$ is designed to evaluate the contextual relevance between original tweet and an article. Similarly, we use cosine similarity between context of words as criteria. Besides, Matrix $\mathbf{C}$ can detect typo in text. In reality, Glove embeddings failed to detect typo while ELMo embeddings are able to do so.

$$\mathbf{C}_{ij} = \frac{\mathbf{h}_i^{\mathrm{qT}} . \mathbf{h}_j^{\mathrm{d}}}{||\mathbf{h}_i^{\mathrm{qT}}|| \times ||\mathbf{h}_j^{\mathrm{d}}||}, \mathrm{i} = 1 \ldots \mathrm{N}, \mathrm{j} = 1 \ldots \mathrm{M} \quad (12)$$

Similar to $\mathbf{S}$, the more similar the context of word $i$ and word $j$ is, the smaller the angle between $\mathrm{g}_i^q$ and $\mathrm{g}_j^{\mathrm{d}}$ is, the bigger $\mathbf{C}_{ij}$ is. Then matrix $\mathbf{C}$ is used as an input of CNNs to determine whether two articles are relevant in a higher level.

## Textual Feature Extraction

To extract textual feature, we stack matrices we obtained in previous step. We stack matrices $\mathbf{S}, \mathbf{A}, \mathbf{C}$ and $\mathbf{S} - \mathbf{C}$ to generate a tensor $\mathbf{Z} \in \mathbb{R}^{N \times M \times 4}$ by following equation:

$$\mathbf{Z} = [\mathbf{S} \oplus \mathbf{A} \oplus \mathbf{C} \oplus (\mathbf{S} - \mathbf{C})] \quad (13)$$

where $\oplus$ denotes matrix stacking. Then we extract features by applying CNNs to $\mathbf{Z}$ and get $\mathbf{P}_i \in \mathbb{R}^{N \times M \times F}, i \in \{1 \ldots n\}$ as result, where $F$ is the number of filters applied in CNNs.

Then we do a k-max pooling to our output and produce vector $\mathbf{o}_{i,j}$ as shown following:

$$\mathbf{o}_{i,j} = \mathrm{kmax}(\mathbf{P}_i[:,:,j]), \mathrm{i} = 1..\mathrm{n}, \mathrm{j} = 1 \ldots \mathrm{F} \quad (14)$$

where $\mathbf{P}_i[:,:,j]$ represents $j^{th}$ column of $\mathbf{P}_i$. Consequently, we obtain result $\mathbf{o} \in \mathbb{R}^{nFK}$ by combining $\mathbf{o}_{i,j}$ as follow:

$$\mathbf{o} = [\mathbf{o}_{1,1}; \ldots; \mathbf{o}_{i,j}; \ldots; \mathbf{o}_{n,F}] \in \mathbb{R}^{nFK} \quad (15)$$

Now we have finish extracting textual features. Next, we will extract features from images and figures.

## Visual Matching Layer

Like text, there are three steps to process images: projecting images into a vector space, calculating similarities of images, and extracting features from similarity matrices.
ResNet50 (He, Zhang, Ren, & Sun, 2016), is used to map images into a vector space, which is an optimized, deeper image learning framework that performs more accurate than traditional image deep learning networks. We denote images $i$ from original tweet as $v_i^q$, and images $j$ from potential FC-articles as $v_j^d$. They are mapped into vector $\mathbf{v} \in \mathbb{R}^H$, then be projected into $\mathbf{m} \in \mathbb{R}^T$ by following equation:

$$\mathbf{m} = \mathbf{W_3} \cdot \mathbf{v} + \mathbf{b_3} \quad (16)$$

where $\mathbf{W}_3 \in \mathbb{R}^{T\times H}$, and $\mathbf{b}_3 \in \mathbb{R}^T$. $T$ and $H$ are chosen as 300 and 2048 in this case. We denote the projection results of $v_i^q$ and $v_j^d$ as $\mathbf{m}_i^q \in \mathbb{R}^T$ and $\mathbf{m}_j^d \in \mathbb{R}^T$.

As we did previously, we consider the tweet and potential FC-article are relevant if they have similar images. The similarity of two image is measured by cosine similarity of their projection into $\mathbf{m}$. Therefore, we calculate matrix $\mathbf{V} \in \mathbb{R}^{X\times Y}$ for any pairs of images from original tweet and potential FC-articles as follows:

$$\mathbf{V}_{ij} = \frac{\mathbf{m}_i^{\mathrm{qT}} . \mathbf{m}_j^{\mathrm{d}}}{||\mathbf{m}_i^{\mathrm{qT}}|| \times ||\mathbf{m}_j^{\mathrm{d}}||}, \mathrm{i} = 1 \ldots \mathrm{X}, \mathrm{j} = 1 \ldots \mathrm{Y} \quad (17)$$

To extract image features, we pool the biggest $\mathbf{V}$ from the matrix as our image feature as follows:

$$s = max(\mathbf{V}), \text{where } s \in \mathbb{R} \quad (18)$$

Note that we assume $s$ as $-1$ for those articles without images.

## Unifying Textual and Visual Information

Now we can derive function $f(q,d)$ as follows, which determines relevance between original tweet and FC-articles. We combine textual feature and visual feature by scaling up vector $\mathbf{o}$ by $s$. We denote this product as $[\mathbf{o}; s]$.

$$f(q,d) = \mathbf{W}_6 \cdot relu\big(\mathbf{W}_5 \cdot relu(\mathbf{W}_4 \cdot [\mathbf{o}; s])\big) \quad (19)$$

where $\mathbf{W}_4 \in \mathbb{R}^{128\times(nFK+1)}$, $\mathbf{W}_5 \in \mathbb{R}^{64\times128}$ and $\mathbf{W}_6 \in \mathbb{R}^{1\times64}$. If we take relevant articles (denoted as $d^+$) and non-relevant articles (denoted as $d^-$) into consideration, our loss function is shown as follow:

$$\mathcal{L}(q,d^+,d^-) = max(0,1 - f(q,d^+) + f(q,d^-)) \quad (20)$$

## Experiment

### Datasets

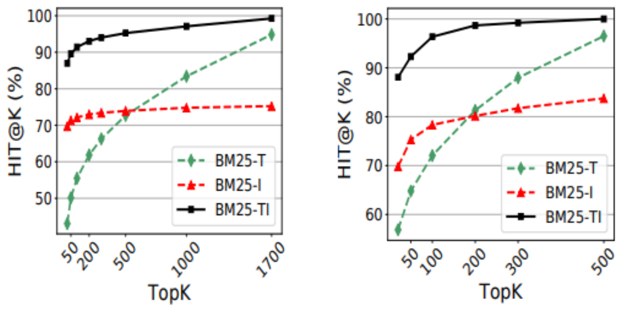
The paper chose a dataset from (Vo & Lee, Learning from Fact-checkers, 2019) as original tweets and their replies from two fact-checking websites, snopes.com and politifact.com, as FC-articles as datasets of the experiment. After filtering from original tweets and FC-articles, the researchers formed two sub datasets, Snopes and Politifact, which contains article pairs that the FC-articles come from the website with the corresponding name of its dataset.

### Experiment

The paper compared their framework with 9 state-of-the-art neural ranking baselines. Basic retrieval and MAN were tested separately.

### Basic retrieval

There are 3 cases to test BM25: text from tweets (BM25-T), images from tweets (BM25-I), text and images from tweets (BM25-TI). In the experiment, BM25-TI performed much better than BM25-T and BM25-I because in addition to limited plain text, a large amount of information is expressed through images as well. Thus, basic retrieval should be based on text and images from tweets.



(a) Snopes (b) Politifact

**Figure 10: result of basic retrieval**

## MAN And Variants

Apart from MAN, there are two variants to be tested, one that only contains text as input, called Contextual Text Matching (CTM), and one that only use image as input, called Visual Matching Network (VMN). To test then, the paper designed two scenarios (SC1 and SC2): SC1 does not consider the text in images, while SC2 consider it.

In scenario 1, our models performed very stable, that outperformed all baselines. On specific topics, CTM and VMN outperformed baselines largely with specific input topic, and MAN stably outperform baselines on average.

In scenario 2, MAN and CTM performed well compared with baselines. However, the performance of MAN varies greatly from different datasets. This might be caused by extensive use of textual signals in SC2. The signals affect the performance of MAN obviously. After extending training data in SC1 to SC2, MAN matched relevant articles more accurate than baselines.

### Overview

Generally speaking, MAN and its variants have different degrees of improvement. Specifically, CTM does not work as well as MAN when dealing with original tweets without images.

## Conclusion

This paper provides a novel framework that does fake news detection by ranking FC-articles according to both text and images. What can be learned from this paper is the idea of detecting fake news by FC-articles, how similarity of text and images is measured and how images can be combined with text. What can be improved is that it would be more efficient if the framework can deal with mobile social network. Whenever we want to update our model with latest tweets and FC-articles, we need to run the learning process again, which extremely waste computational power. In next section we will introduce a model that can handle mobile network.

## A Simplified and Dynamic Graph Neural Network (SDG)

In this section we will introduce a simplified and dynamic graph neural network, or SDG in short, which is a dynamic and efficient graph neural network that can easily deal with mobile network.

## General Idea

To achieve building a simplified and dynamic GNN, the model needs to be able to keep tracking changes of the graph. In the paper, this is designed based on personalized PageRank tracking process. Specifically, other nodes use $k$-layer GNN model and $k$-step random walk distribution starting from the selected seed node. In this way whenever there occurs a change in the graph it can quickly and accurately track the stationary distribution of random walk, rather than solving it from scratch.

## Framework

### Overview

SDG model contains an input layer, a dynamic propagation step, and an output layer. The author of (Xu et al., 2018) have shown that in a $k$-layer GNN model, the impact of a node $V_j$ on itself is propotional to the $k$-step random walk distribution on node $V_i$ starting from the seed node $V_j$, where the random walk distribution converges to stationary distribution as $k$ tends to infinity. Based on that, to dynamically propagate the neighborhood information between nodes instead of stacking layers, the predicted matrix is configured as follows:

$$Z = \text{softmax}(\boldsymbol{PH}) \quad (21)$$

where $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is the dynamic propagation matrix, $\boldsymbol{H} \in R^{n\times c}$ is the hidden node feature matrix extracted from the input node features $\boldsymbol{X} \in R^{n\times d}$ through the model-agnostic neural network $f_\theta$ , i.e., $\boldsymbol{H} = f_\theta(X)$. In next section we will explain how to get $\boldsymbol{P}$ and $\boldsymbol{H}$.

### Dynamic Propagation Scheme

The key to realize dynamic propagation is the dynamic propagation matrix $\boldsymbol{P}$, which is defined as each row, denoted as $P(i,:)$, consisted of the stationary distribution of random walks starting from the node $V_i$. $\boldsymbol{P}$ can be represented as follows:

$$P(i,:)^\top = \alpha \boldsymbol{M} P(i,:)^\top + (1 - \alpha) r \quad (22)$$

where $\boldsymbol{M} = \boldsymbol{AD^{-1}} \in \mathbb{R}^{n\times n}$ denotes the column-stochastic transition matrix, $\alpha \in [0,1]$ denotes the teleportation probability, and $\boldsymbol{r} \in R^n$ denotes the personalized vector with $r(i)$ = 1 and other entries equal to 0. For simplicity, the transpose notation T for the following $P(i,:)$ will not be marked out.

When the graph changes, $P(i,:)$ can easily do response to the changes as each row of $\boldsymbol{P}$ is independent. Specifically, when the graph changes from $\boldsymbol{M}$ to $\boldsymbol{M'}$, we push out $\boldsymbol{P}$ from the changed rows to unchanged rows, then add the pushed out distribution back to the previous distribution to form the new matrix $\boldsymbol{P'}$. Above mathmetical process are showned as follows:

$$P(i,:)_{pushout} = \alpha(\boldsymbol{M'} - \boldsymbol{M})P(i,:) \quad (23)$$

and

$$P\,(i,:)' = P(i,:) + \sum_{k=0}^{\infty}(\alpha M')^k P(i,:)_{pushout} \quad (24)$$

where $P(i,:)_{pushout}$ denotes the distribution score that needs to be pushed out on the residual graph, and $P(i,:)'$ denotes the tracked new distribution. This push out procedure is proved in the paper to be converged as $k \to \infty$. The time compexity of finding $P'$ with error bound $\frac{\epsilon}{1-\alpha}$ is $O(\frac{mn}{q}\log_a \frac{\epsilon}{2})$, where $m$ is the number of non-zero entries of $M'$,and $n$ is the number of nodes in the graph, which is much faster than solving by brute force, which is $O(n^3)$.

## Model-Agnostic Neural Networks

Matrix $\boldsymbol{H}$ is derived from $f_\theta$, which can be various training process such as CNNs, and $\boldsymbol{X}$, which is the features extracted from input graph. Thus, $\boldsymbol{H}$ can be derived by following equation:

$$\boldsymbol{H}(i,:) = f_\theta(\boldsymbol{X}(i,:)) \quad (25)$$

where $X(i,:)$ denotes the input node features of node $V_i$, and $H(i,:)$ denotes the hidden node feature of node $V_i$ extracted by $f_\theta$. Note that we process information propagation and graph feature extraction seperately since dynamic propagation matrix $\boldsymbol{P}$ is independent to the training process $f_\theta$, which means our SDG is able to be scaled up or down and interpret the impact of nodes.

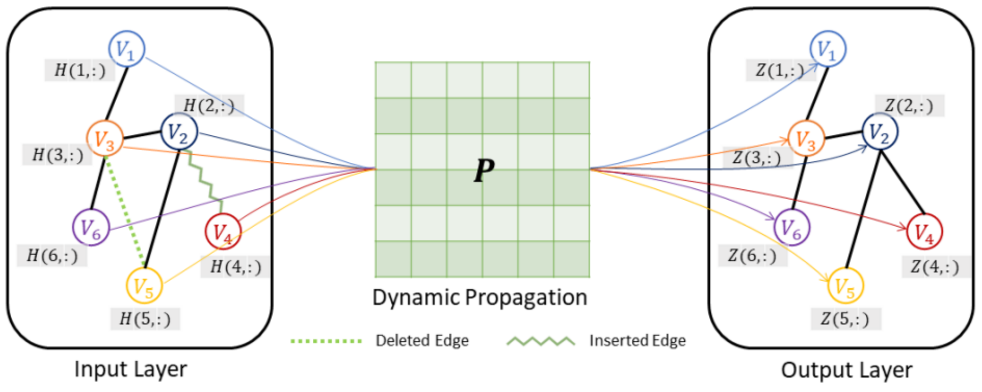


**Figure 11: structure of SDG**

## Algorithm

Generally speaking, the SDG algorithm is that keep running $f_\theta$ to train matrix $\boldsymbol{H}$

and $\boldsymbol{Z}$ until stable first, then to any changes to the graph, adjust $f_\theta$ according to changes between $\boldsymbol{P}$ and $\boldsymbol{P}'$. Specifically, the pseudo code of how SDG model works is as follow:

```
Input:
    Graph $G$ with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$,
label matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$, and any possible updates $\boldsymbol{A}' \in \mathbb{R}^{n \times n}$ and $\boldsymbol{X}' \in \mathbb{R}^{n \times d}$.
Output:
Predictions $Z \in \mathbb{R}^{n \times c}$ and updated predictions $Z \in \mathbb{R}^{n \times c}$.
1:  Compute the initial dynamic propagation matrix $\boldsymbol{P}$.
2:  while not converge do
3:      Train $f_\theta$ through Eq. 1 and Eq. 2 to obtain qualified $\boldsymbol{H}$ and $\boldsymbol{Z}$.
4:  end while
5:  if graph structure and/or node features change then
6:      Update $\boldsymbol{P}$ into $\boldsymbol{P}'$ through Eq. 4 and Eq. 5.
7:  end if
8:  while not converge do
9:      Fine-tune $f_\theta$ on any new matrices $\boldsymbol{A}'$ and/or $\boldsymbol{X}'$.
10: end while
```

## Experiment

### Data Sets and Baseline Algorithms

The paper used three citation datasets in reality and aimed to classify them by algorithms. The graph is formed by regarding papers as nodes, and citation between papers as edges.

The paper chose PPNP and APPNP, which are two simplified graph neural network models, as baseline algorithms. Besides, a varient of SDG, named SDG-S that ignore the dynamic propagation scheme, ie., set $k$ to be 0, was designed as control group.

### Experiment

The performace of algorithms are evaluated from two aspects: speed and accuracy. From the experimental data, it is obvious to see SDG and SDG-S run faster than the two baselines. Besides, the experiment showed that when the change of graph is relevantly small, SDG and SDG-S have performance. As $k$ and $\alpha$ increase, the accuracy of classification increase, the model had a slightly better performace. This is might because the model-agnostic neural networks SDG used compensate the problem that the algorithm is not accurate enough.

## Conclusion

SDG is a dynamic and simplified neural network model that outperformed traditional GNNs in terms of ways to deal with updating graphs and message passing in graphs. It reduces the consumption of computational power and increase the speed and accuracy when dealing with dynamic networks. This technique can be applied in fake news detection field as social network is mobile.

# Future Work

So far, the research has reached this place. More works need to be done in the future. Next I should go through the code provided by the two paper mentioned in section 3 and come up with our own model.

# Reference

Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Rao, J., Liu, L., Tay, Y., Yang, W., Shi, P. and Lin, J., 2019. Bridging the Gap between Relevance Matching and Semantic Matching for Short Text Similarity Modeling. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. and Sun, M., 2020. Graph neural networks: A review of methods and applications. AI Open, 1, pp.57-81.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. I., & Jegelka, S. (2018, July). Representation learning on graphs with jumping knowledge networks. In International Conference on Machine Learning (pp. 5453-5462). PMLR.

Vo, N. and Lee, K., 2019. Learning from Fact-checkers. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval,.

Vo, N. and Lee, K., 2020. Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),.