A Study on Deep Learning for Fake News Detection

Pham Trung Tin

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
March, 2018

Master's Thesis

A Study on Deep Learning for Fake News Detection

1610157 Pham Trung Tin

Supervisor : Nguyen Minh Le
Main Examiner : Nguyen Minh Le
Examiners : Satoshi Tojo
Kiyoaki Shirai
Shinobu Hasegawa

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science

February, 2018

Abstract

Fake news detection is the task of classifying news according to its veracity. In a simple setting, it is a binary classification task, whereas in a more difficult setting it is a fine-grained classification task. Fake news detection is one of the hottest research topics in artificial intelligent recently. Due to the availability of the Internet and the readiness of sharing information through social media, it is easy to make fake news and propagate it worldwide. When being spread widely, the fake news may cause substantial adverse impact to many aspects of life. Consequently, an amount of research has been conducting recently to spot fake news. Despite receiving significant attention from the crowd, fake news detection did not gain much improvement for a while due to the insufficiency of fake news data. In fact, it was not until 2017 with the introduction of Liar dataset, has fake news detection shown some noticeable performance.

Most of the current methods applied to tackle fake news detection are based on deep learning on the ground that deep learning models have been achieving state-of-the-art results in a wide range of artificial intelligence such as natural language processing, computer vision, speech processing. Concretely, Convolutional Neural Network(CNN) and Long Short-Term memory network(LSTM) were used in fake new detection and yielded remarkable results. Having said that, fake news detection is still in its infancy with the accuracy of fine-grained being under 50%. In an attempt to improve the performance of fake news detection, we explore a range of techniques focusing on attention-based neural networks. In this thesis, our contributions are: (1) the implementation of single attention with an intermediate representation, (2) the modeling of two-way relationship between main text and connected information, (3) the proposal of memory network in word level and character level.

First of all, we present the simplest approach called single attention neural network for fake news detection. In this model, the statement is merely considered a sequence of words, and different types of side information are also considered a kind of sequential data. The side information is then represented as a unique vector. For clarity, we simply sum over all types of side information. This vector then act as an attention factor over the main text. Our experiment demonstrates that the single attention neural network surpasses the more complicated hybrid CNN which employs two CNNs and one LSTM to drive intermediate representations.

Our second approach stems from the question that whether the main text has reverse effect on the given dataset. To clarify this doubt, we try to model the two-way interaction between the text and its connected information. First of all, side information is processed as described in the single attention neural network. In the reverse fashion, the text is also twisted into a unique vector to act as an attention component. Similar to the case of single attention, we take summation of all vectors for words in the statement. Our experiment show that the mutual relationship between text and its connected information is relevant to detecting fake news. In fact, the performance increases more nearly 10%.

Still, there is a gap to arrive at the same level of the state-of-the-art.

Our next implementation is the word level memory network. In this module, we try to construct a memory comprising a number of cells. The structure of the network includes two types of memories which are input and output memory. These memories allow for learning different representations from the given input. In the attempt to mimic the operation of a memory, we also allow update of memory cells. Moreover, we do not treat all types of side information as a whole. Rather, they are examined separately. Our experimental results show that our memory network helps improve the performance of the state-of-the-art. The results also reveal that different types of side information do not contribute equally to the task, and the mixture of all side type information may not be useful.

Our final effort in detecting fake news is the proposal of character level memory network. In this model, we examine another way to construct memory cell. In fact, we build each memory cell from all character of words, instead of words themselves. In doing so, we take advantage of a weighting encoding to fill value for a memory cell from characters of a word. Unfortunately, our experiments exhibit that the encoding scheme from character level does not further enhance the result.

Overall, we introduce a number of approaches to fake new detection task. Those can provide a view attention-based neural network approach for further advances. The most crucial contribution is that we push the accuracy to a higher level, which surpasses the state-of-the-art.

Keywords: fake news detection, attention mechanisms, attention-based neural network, natural language processing.

Acknowledgment

I would like to express my greatest, sincerest gratitude to my main supervisor, Associate Professor Nguyen Le Minh, for a range of opportunities and aids he has been giving up on me. From the very beginning, Professor Nguyen Le Minh offered me a chance to study in Japan when I was struggling to find one to continue my education in my home country, Vietnam. This led to a wonderful time in my life in terms of both research career and personal life. Regarding doing research, he always gave me insightful, instructive orientations during the course of my master years. In fact, conversations between us yielded numerous wonderful ideas to my main research, and helped me overcome the problem I was confronted. During two years at JAIST, I have learned a variety of valuable knowledge and skills which would follow me for the rest of my life as a researcher. Associate Professor Nguyen Le Minh also gave out sympathy and encouragement whenever I fell into severe depressions due to personal issues. Without his care, it is unlikely that I can get through all difficulties and finish my Master on time. In addition, thanks to a chance to study in Japan that Associate Professor Nguyen Le Minh brought to me, I discovered a number of cultural activities and events. Specifically, I really enjoyed Hanami festival in the spring, Hanabi festival in the summer and Yuki festival in the winter. I also conquered the three sacred mountains in Japan: Mt.Fuji, Mt.Tate, and Mt.Haku. Further, I developed three hobbies my cooking and photography and jogging. Overall, Associate Professor Nguyen Le Minh gave me meaningful advice and valuable knowledge in research, the exploration of Japanese culture, and personal development. From the bottom of my heart lies a big thank and respect to him.

Furthermore, I would like to thank all members of Nguyen lab, with whom I have shared wonderful moments at JAIST. In fact, weekly seminars are the most exciting we have together. On the seminars, labmates give their inquiries and comments on the presented topic, which force presenters to think deeply and critically about what they are doing. Thanks to these weekly activities I can practice presentation skills and critical thinking, which are important parts in doing research. Besides, I am happy to also have picknick and other activities with the lab's members.

I also appreciate Professor Ho Tu Bao for the minor research. Working with him, I learn how to formulate and choose a problem when doing research. He also taught me how about to make a professional, well-structured presentation. I am also grateful to Associate professor Kiyoaki Shirai and Professor Satoshi Tojo for their valuable comments on my research.

I would like to thank my Japanese teacher Kuniko Kitajima for giving her time and effort in teaching me Japanese. Thanks to her, I have some noticeable progress in learning, which is helpful to my personal life in Japan.

Last but not least, thank all of my fellow and international friends who are always with me, and constitutes my life full of happiness and success at JAIST.

Pham Trung Tin
February 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Fake news is a type of news that has no basis in fact, but is presented as being factually accurate[1]. It may has misleading, false, imposter, manipulated, fabricated content, or satire, parody, and false connection with the intent to mislead people. As such, fake news may has substantial impacts on numerous aspects of life.

Politically, fake news could be employed in election campaigns or politic-specific events for or against famous figures. For example, Donald Trump said in a tweet on Tweeter[2] that "*he won the second debate with Hillary Clinton in a landslide in every poll.* Justification from POLITIFACT.COM reveals that not only did Trump not win by a landslide in any of the polls, he didnt win any of the polls[3]. However, this statement was to gain favor for himself during the election campaign.

Economically, fake news may exert devastating effects on the consumption of food and products. Take the fake news that grapefruits could cause cancer for example. This unfounded allegation circulated through a number of Vietnamese newspapers in 2007, resulting in a false perception of the public about the fabricated connection between eating grapefruit and having cancer. This fake news led to a severe drop of grapefruit' price, which plummeted to only 10% of the current one[4]. Farmers could not sell their grapefruits, or sold with a very low price, hence the local economy suffered from hundreds of billion damage. The related newspaper agencies were later charged with getting involved in spreading this fake news.

Socially, fake news may destroy one's esteem and social status or even cause social unrest. A picture of a woman wearing a hijab and talking on the phone in the site of a terror

[1] `https://en.wikipedia.org/wiki/Fake_news`

[2] https://twitter.com

[3] http://www.politifact.com/wisconsin/statements/2016/oct/12/donald-trump/donald-trumps-ridiculous-claim-all-polls-show-he-w/

[4] https://tuoitre.vn/tin-don-an-buoi-bi-ung-thu-lam-thiet-hai-hang-tram-ti-216359.htm

attack in London, Uk enjoyed thousands shares with #BanIslam hashtag and a claim that the woman was indifferent to the suffering of victims around her[5]. This posed negative attentions to the woman and rose hatred towards Islam. The woman later posted a statement that she was devastated witnessing the attack.

1.2 Challenges

Challenges of detecting fake news springs from the fact that it is even difficult for human beings to separate fake news from true news. Concretely,

- Language use is complex in fake news. Rashkin analyzes the difference between language use of fake news and that of true news on three kinds of fake news: satire, hoax, and propaganda. Her work reveals that a wide range of linguistic factors contribute to the formation of fake news such as subjective, intensifying, and hedging words with the intent to introduce vague, obscuring, dramatizing or sensationalizing language. Therefore, applying feature-based approaches will be labor-intensive and time-consuming.

- Fake news usually mixes true stories with false details, which are confusing to be recognized correctly. It is often that fake news maker blend true story with false details to mislead people. For example, the statement "*However, it took $19.5 million in Oregon Lottery funds for the Port of Newport to eventually land the new NOAA Marine Operations Center-Pacific*" is half-true since it combines the true number of $19.5 million and the misleading place where the money went to. In such case, it is easy to get people's attention about trusted parts without noticing the presence of fabricated ones.

- Fake news data is limited. Currently, there is only political fake news dataset published. Domains other than politics are still open to future research.

1.3 Problem definition

Suppose that we are given a training set of statements $S = \{s_1, s_2, \ldots, s_N\}$ and associated side information $U = \{u_1, u_2, \ldots, u_N\}$, where N is the number of statements. Each s_i consists of a sequence of words w_1 w_2, ..., w_n, while each u_i is a single or a set of side information. Our basic goal is to predict whether the statement is fake or true, or more challengingly, to classify it into a fined-grained level of truthfulness.

1.4 Introduction of method

Our work explores a variety of attention mechanism variants in detecting fake news. Particularly, we employ four kinds of attention-based neural networks: single attention,

[5]http://www.bbc.com/news/world-42487425

dual attention, word-level and character-level memory networks as a continuation of using deep learning in detecting fake news. Single Memory neural network focuses on a one-side interaction between a statement and its associated side information while the dual one makes use of two-side interaction. Furthermore, Memory network, a kind of attention-based neural networks, can exhibit the ability to give selective focus on subregions of a given input performed by attention mechanism. Besides, they facilitate the storing of extra information in memory vectors, which is showed to be effective in many tasks such as language modeling, question answering [10]. Character-level memory network with a memory encoding scheme is then implemented as our last model.

1.5 Contributions

Our main contributions in this paper are:

- Exploration of single attention which indicates the effect of side information over the main text.
- Discovery of mutual interaction between texts and side information via a dual attention neural network.
- Proposal of a memory network that is able to store external information helpful for fake news detection.
- Investigation of multiple computations by reading input repetitively in a stacked memory network.
- Production of an accuracy that surpasses that of the current state-of-the-art.
- Exploration of character level memory network which takes advantage of an encoding scheme into memory cells.
- Evaluation of the model for both of 6-label and 2-label classifications.

Our thesis is structured as follows. Chapter 2 will describe some background, while our chapter 3 will concentrate on our approaches. Chapter 4 will present our experiments and results, followed by conclusion in chapter 5.

Chapter 2

Literature Review

2.1 Deep learning models

2.1.1 Neural Networks

Inspiration

Inspired by biological neuron systems, artificial neural networks, neural networks, or feed-forward networks are algorithms that try to mimic the brain and the way it functions. In biological setting (Figure 2.1[1]), one neuron receives a signal from its tree of dendrites, or dendritic tree, and if the signal is strong enough, it will pass through an axon and link to a dendrite from another neuron. Two neurons are actually separate from each other by synaptic gaps and only become connected when the link of an axon from one neuron and dendrite from the others are stimulated.

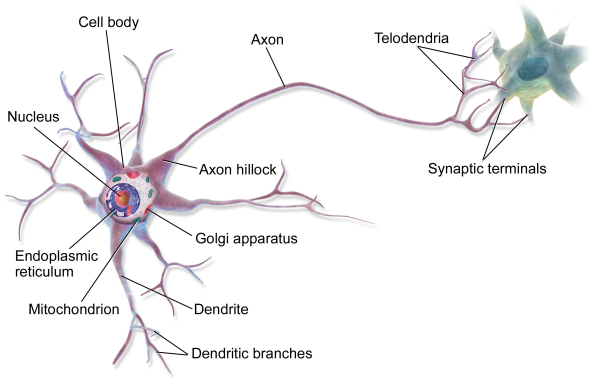


Figure 2.1: Biological neural network

[1] https://en.wikipedia.org/wiki/Biological_neural_network

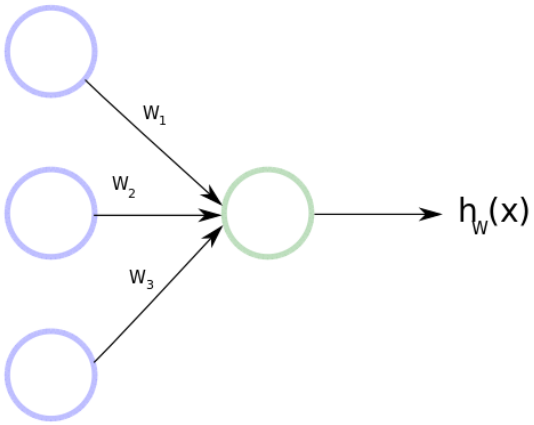


Figure 2.2: A simple artificial neuron network with one output unit

Model representations

How to model this system? Given a binary input $x_1 \in (0, 1)$ representing whether a neuron is fired or not, it gets multiplied by a weight W_1. This part is to model the synaptic connection between two neurons where W_1 is corresponding to the degree of connection; it is bigger if the connection is strong, and smaller otherwise. In other words, it reflects the influence of synaptic connection to the decision whether or not the axon is stimulated. Similarly, we also have x_2, x_3, ..., x_n that get multiplied by W_2, W_3,..., W_n respectively. All of the products are then summed into one unit to depict **collective influence** of those inputs. But whether the input is strong enough to make the neuron fired? To model this, we take summation of all input neurons and put the result through an activation function. If the of output of the activation function greater than 0, the axon is stimulated. Figure 2.2 describes a neuron network with logistic activation function. In this case, activation $h_W(x)$ is computed as:

$$h_\theta(x) = g(W^T x) \tag{2.1}$$

where g(z) is a activation function (logistic function in this example) computed as:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

In the same fashion, multiple connections are modeled by multiple layers with different sets of weights. Suppose that we have a neural network with 3 layers as described in Figure 2.3, activations of the hidden layer (layer 2) are computed as:

$$a_0^{(2)} = g(W_{00}^{(1)} x_0 + W_{01}^{(1)} x_1 + W_{02}^{(1)} x_2 + W_{03}^{(1)} x_3) \tag{2.3}$$

$$a_1^{(2)} = g(W_{10}^{(1)} x_0 + W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3) \tag{2.4}$$

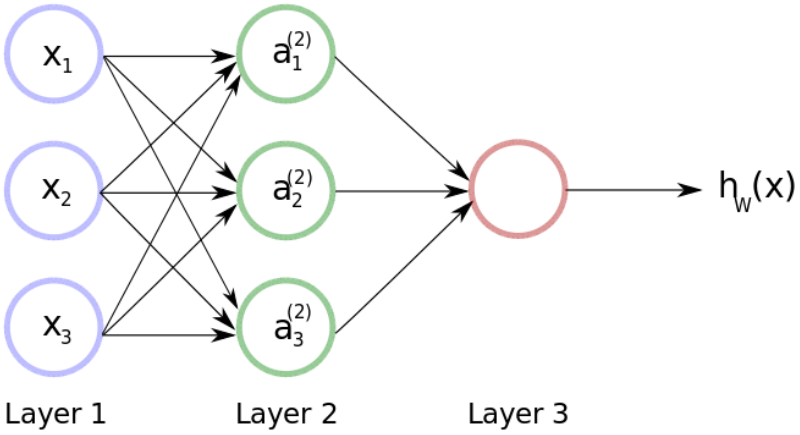


Figure 2.3: Neuron model with logistic activation function. Note that biases x_0 and $a_0^{(2)}$ are omitted in this figure.

$$a_2^{(2)} = g(W_{20}^{(1)}x_0 + W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3) \tag{2.5}$$

$$a_3^{(2)} = g(W_{30}^{(1)}x_0 + W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3) \tag{2.6}$$

In machine learning literature, equations (2.3), (2.4), (2.5) are rewritten in matrix notation. Firstly, weight matrix representing the connection between layer 1 and layer 2 is rewritten as:

$$W^{(1)} = \begin{bmatrix} W_{00}^{(1)} & W_{01}^{(1)} & W_{02}^{(1)} & W_{03}^{(1)} \\ W_{10}^{(1)} & W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{20}^{(1)} & W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{30}^{(1)} & W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \end{bmatrix} \tag{2.7}$$

Then,

$$z^{(2)} = W^{(1)}x \tag{2.8}$$

$$a^2 = \begin{bmatrix} a_0^2 \\ a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} = g(z^{(2)}) \tag{2.9}$$

Finally,

$$z^{(3)} = W^{(2)}a^2 \tag{2.10}$$

$$h_W(x) = a^3 = g(z^{(3)}) \tag{2.11}$$

Neural network for multi-label classification

Suppose we have to perform a 3-label classification task, a neural network in Figure 2.4 can be a possible solution to the problem. Vector output h_W is a 3-dimensional one hot vector.

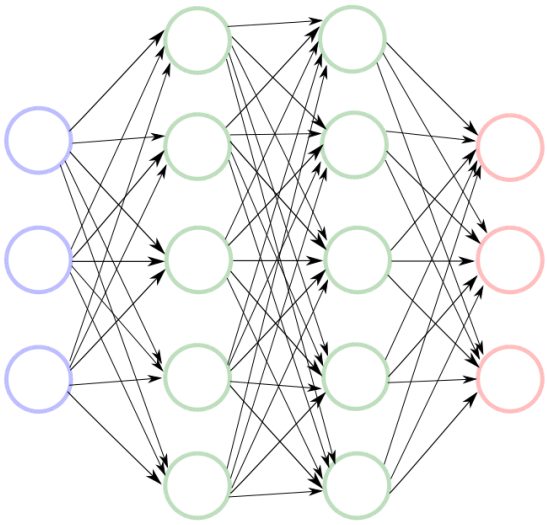


Figure 2.4: Neural networks with 2 hidden layers for 3-label classification

Squared Error Function

Loss function denotes the difference between predicted output $\hat{y}$ from the model and the ground truth y. A naive approach may be applied by taking difference between them or norm 1:

$$L = |y - \hat{y}| \tag{2.12}$$

For mathematic convenience when taking derivatives, square error, or norm 2, is applied as:

$$L = (y - \hat{y})^2 \tag{2.13}$$

However, norm 1 or norm 2 are rarely used. Instead, cross entropy, which will be described in the next section, own some characteristics that make it preferred in neural networks.

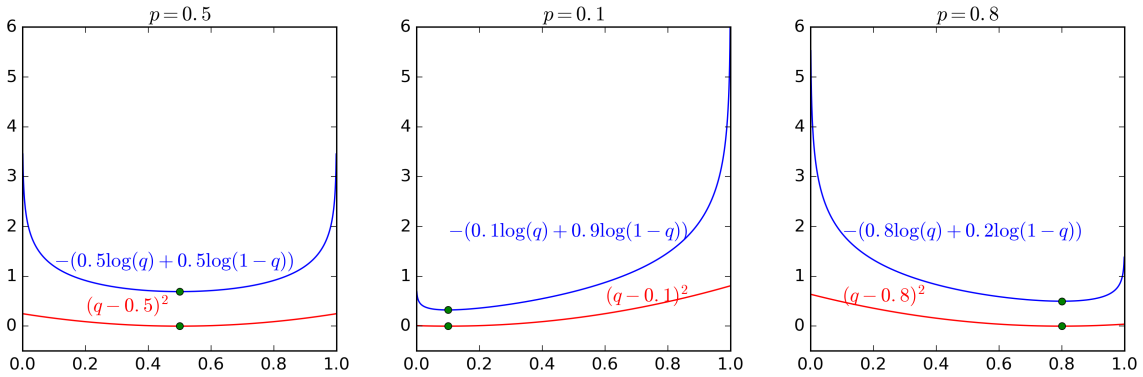
Cross Entropy

Cross entropy between two discrete distributions is defined as:

$$H(y, \hat{y}) = -\sum_{i=1}^{C} y_i log(\hat{y}_i) \tag{2.14}$$

Figure 2.5 demonstrates the difference between cross entropy and square error function. We observe the following phenomena:

- maximum values of both function are obtained when $q = p$ at the green spot.
- cross entropy makes the difference between two distribution p and q become more pronounced than the square function does. Concretely, when p and p are far away



(a) (b) (c)

Figure 2.5: Comparison of cross entropy and square function. Blue lines are for cross entropy, while red lines are for the other function.

from each other, the loss will be bigger in cross entropy, and smaller than in square error.

These two characteristics make it favorable for optimization purpose.

2.1.2 Convolutional Neural Networks

Convolutional Neural Network is a subset of normal neural networks in that it employs multiple connections between neurons of a layer to those of the next one through a set of weight matrix and non-linear activation functions. However, convolutional neural networks have some differences from ordinary one:

- Convolutional neural networks take as input a matrix (eg. a 2-dimensional or 3-dimensional matrix) instead of a vector as in ordinary one.
- Instead of getting a dot product of weight matrix and input, convolutional neural networks calculate convolution between them.

A CNN has two primary components: convolutional and pooling layer. For the ease of explanation, let's assume that the input is a 2-dimensional matrix.

- **Convolutional layer**: a matrix called filter or kernel is used to slide over subparts of the given inputs. The result after sliding is termed *feature map*. In this scene, the input and the filter matrix are considered two functions, and sliding of the filter matrix over the input matrix is to mimic the convolution operation of the two functions. The "speed" of sliding is determined by *stride size*, while the the size of the filter is called *kernel size* or *filter size*. The purpose of this operation is to learn a certain representation from the given input. For a certain problem, therefore, various filters with different filter size are usually applied to extract different features from the given input. This convolutional layer has two prime characteristics: **sparse**

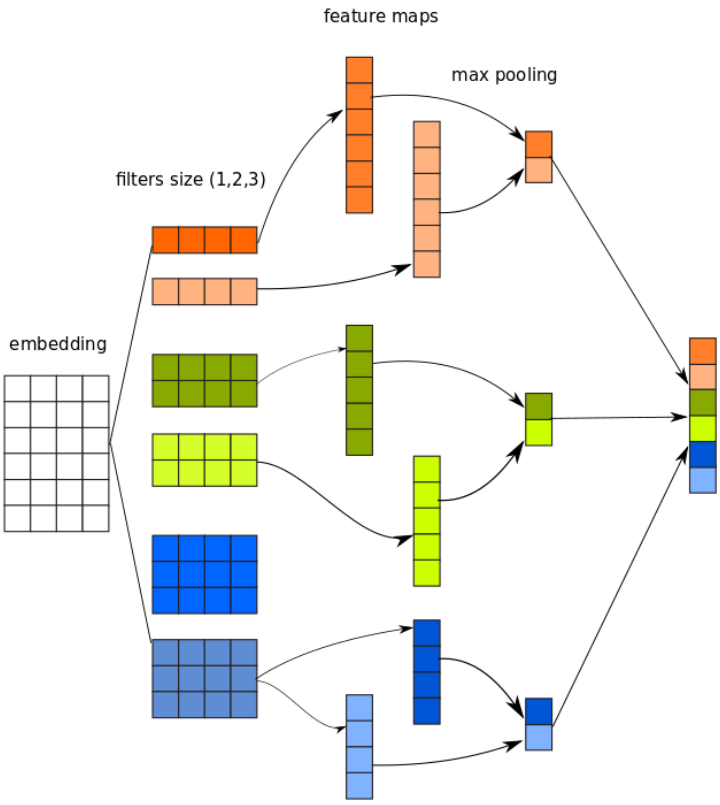


Figure 2.6: Examples of CNN for NLP

connectivity and **share weights**. Unlike in normal neural networks where all units from one layer are connected to all units from the next one, a unit from a layer is just locally connected with some units in the previous one, which is called "sparse connectivity". On the other hand, because of the sliding, the weight of the filter is shared throughout all sub-regions of the given inputs.

- **Pooling layer**: is a kind of subsampling. Given a matrix, the pooling layer is to extract the most prominent (in case of max pooling) value or combine all of the values (average pooling).

One interesting trait of CNN is **compositionality**. One pair of convolutional and pooling layer can be stacked by another pair, which facilitates the learning of complex non-linear functions. Compositionality, sparse connectivity and shared weight allow for learning interesting features form a given input. Despite being famous for computer vision tasks such as object recognition or image captioning, CNN has also been applied in natural language processing and yielded remarkable results. First of all, an input sentence is represented as a matrix, then a number of filters are slid over it. In this case, a 1-dimensional convolution operation is used. The use of different filter sizes such as 1,2,3 is

considered equivalent to as n-gram features in case of natural language processing.

2.1.3 Recurrent Neural Networks (RNN)

RNN is a subclass of neural networks where the same subnetwork (also called cell) is repeated for multiple times to read different inputs. The repetitive structure is illustrated as in Figure 2.7

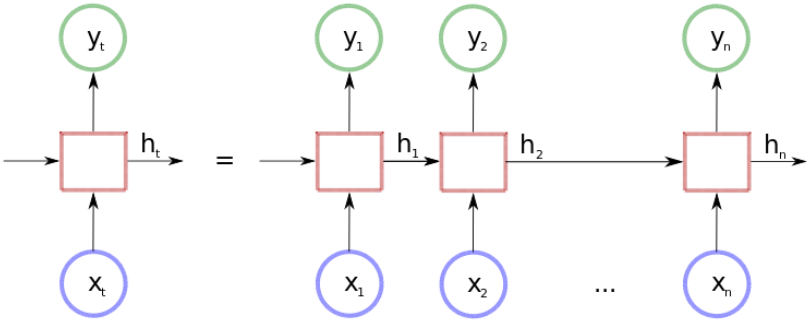


Figure 2.7: An unrolled recurrent neural network

Given input x_t and hidden state of previous step h_{t-1}, new hidden state and output at time step t is computed as:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.15}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{2.16}$$

where:

- x_t is input vector at time step, h_t is hidden layer vector, y_t is output vector at time step t.
- W,U,b are parameter matrices and vector.
- σ_h, σ_y are activation functions

This network is particularly designed to deal with sequential data where inputs are not fed into the networks all at once, but are broken down into small pieces which are later passed into the network cell one after another. Despite being designed to deal mimic and work on sequence nature of some kinds of data, it is proved that RNNs have limitations in capturing long dependencies. As a result, Long Short-Term Memory Network, a modified version of RNN with gating mechanisms, is devised to get over the limitation of traditional RNNs.

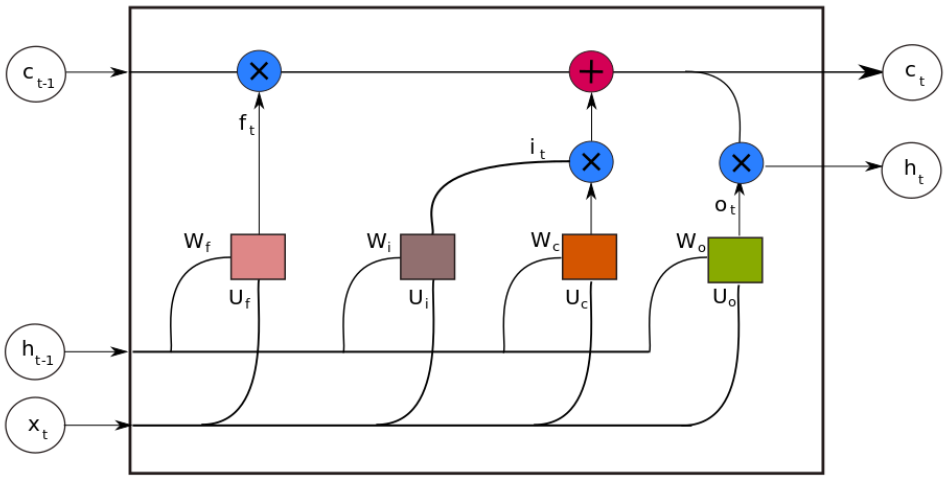


Figure 2.8: Internal structure of Long Short-Term Memory Networks

2.1.4 Long Short-Term memory Networks (LSTMs)

To overcome the drawback of traditional RNNs, 3 gates are added into the cell of the network to facilitate the notion of memory. In fact, a memory is kept and updated when the cell reads inputs at each time step. Figure 2.8 illustrates LSTMs with four gate: forget (f), input (i), memory (c) and output gate (o).
Given an old memory C_{t-1}, the new cell memory C_t is computed as:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2.17}$$

Forget gate: decides which information is to be eliminated from the current memory. Given an input at x_t time step t, it is computed as:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.18}$$

C_{t-1} then gets multiplied with this f_t to transform it with some information removed.
Memory gate: generates new candidate memory. Given an input x_t, it is computed as:

$$\widetilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.19}$$

Input gate: This gate determines how much information of the candidate memory will be injected into the updated one. Given an input x_t, it is computed as:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.20}$$

$\widetilde{C}_t$ then gets multiply by i_t to get the new added memory into the new memory cell.
Output gate: determines how much of the cell memory is extracted out. It is computed as:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.21}$$

the new hidden state is then updated as:

$$h_t = o_t * \sigma_c(C_t) \tag{2.22}$$

It is said that with the presence of internal memory and its capability to get update sequentially, the long dependency problem is addressed.

2.1.5 Attention Mechanism

In this section, we first introduce softmax function and get to attention mechanism in detail later.

Softmax Function

Softmax function is the key to attention mechanism. In fact, it is to derive the probabilities of different regions of a given input. Suppose we have a neural network as depicted in Figure 2.9, softmax function is defined as follows:

$$a_i = \frac{exp(z_i)}{\sum_{j=1}^{C} exp(z_j)}, \forall i = 1, 2, 3..., C \tag{2.23}$$

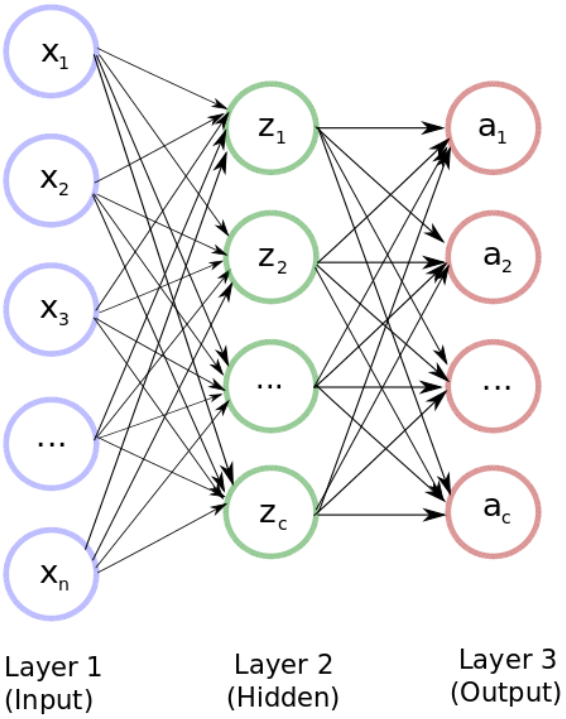


Figure 2.9: Softmax layer

This function possesses some favorable features:

- each z_i is certainly transformed into a positive number by being put into $exp(z_i)$. In other words, all output a_i are positive numbers even if some z_i are negative.
- It is smooth and differentiable. Therefore, it is easy to take derivatives, and hence backpropagation.
- The function is monotonically increasing in that it produces large a_i if z_i, small a_i otherwise.
- sum of all a_i is equal to 1.

Therefore, the output vector $a = [a_1, a_2, ...a_n]$ can be interpreted as a probability vector. Specifically, we can assume that:

$$P(y_k = i|x_k, W) = a_i \tag{2.24}$$

$P(y_k = i|x_k, W)$ is perceived as the probability of data point x falling into class i given parameter matrix W of the model.

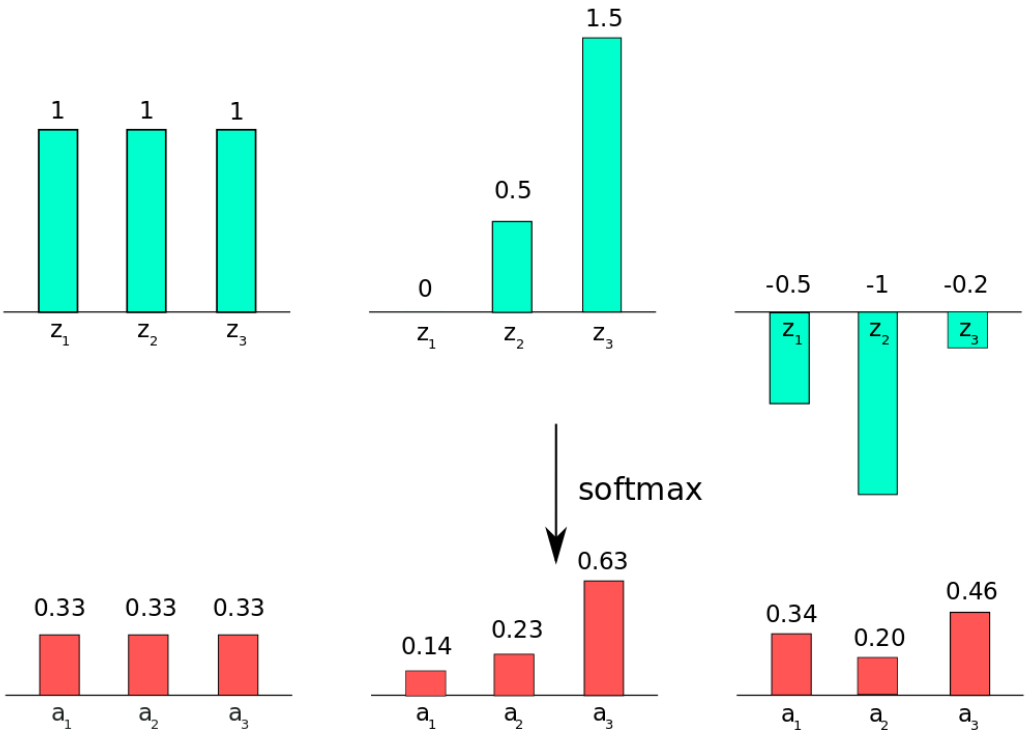


Figure 2.10: Examples of softmax output

Score Function

Score function is to measure the relevance between two input vectors, which may take one of the following factor :

$$score(s_{i-1}, h_j) = \begin{cases} s_{i-1}^T h_j, & \text{doc} \\ s_{i-1}^T W h_j, & \text{general} \\ v^T tanh(W[s_{i-1}^T; h_j]), & \text{concat} \end{cases} \tag{2.25}$$

Depend on the task and the amount of data given, one out of the three forms of the the score function can be applied. With the *doc* version, the relevance is computed directly from the two given vector, hence there is no need for learning parameters, but it can be limited in some cases. On the other hand, the **general** and **concat** versions provide more flexibility with the introduction of weight matrices. However, because of having more parameters to learn, the two functions may be inappropriate when large amount of data is unavailable.

Attention Mechanism

Attention Mechanism is a technique that allows for selective focus on certain parts of a given input. With the intent to overcome the limitation in long dependency faced by RNN-based models, attention mechanism is first applied in machine translation with remarkable results, it is then widely used in numerous state-of-the-arts systems beyond machine translations including machine reading, image captioning, aspect-based sentiment analysis, and natural language inference. To introduce the concept of attention mechanism, we take attention in Machine Translations as an example.

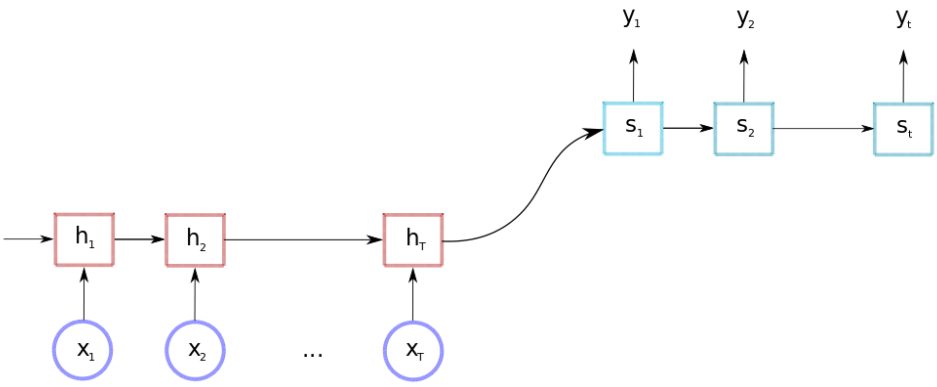


Figure 2.11: Basis Neural Machine Translation

Neural Machine Translation (NMT) employs RNN encoder-decoder framework, which first uses a deep model such as a LSTM to generate a fixed-length vector representing

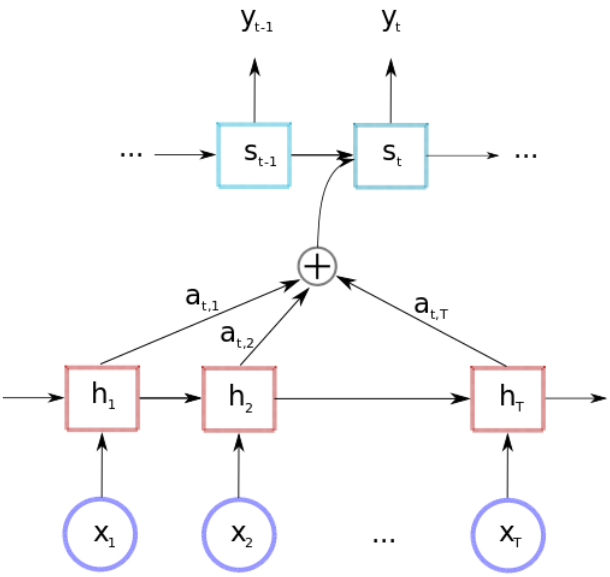


Figure 2.12: Attentive Neural Machine Translation

overall meaning of the source sentence in he encoder phase. Typically, the last hidden state generated by the LSTM will be used to represent the meaning of the sentence, or sentence embedding in a more technical term. Detail of NMT is described in Figure 2.11. Formally, given a sequence of vectors x = (x_1, x_2, ..., x_{T_x}), an RNN is used such that

$$h_t = f(x_t, h_t - 1) \tag{2.26}$$

, where f is a nonlinear function. Then,

$$c = h_{T_x} \tag{2.27}$$

However, this approach suffers from long dependency problem when using RNN-based model for sequential data. To mitigate this problem, attention mechanism is advised to let the model learn selective attention on some parts of the input sentence at each step of output generation in decoder phase. Figure 2.12 demonstrates attention mechanism in neural machine translation.

Weight α_{ij} associated with h_i is computed by:

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.28}$$

where

$$e_{ij} = score(s_{i-1}, h_j) \tag{2.29}$$

, with s_{i-1} is the state of the RNN decoder at the time step $i-1$.

As described in previous section, using softmax function here means that the summation of all α_{ij} in step i will equal to 1. each α_{ij} represents how much word j in the source sentence contribute to the output words y_i. In other words, each generation of y_i will find its own way to computed weight sum of input differently, which in turn give attention to different parts of the given input.

Besides giving solving long dependency problem, attention mechanism is also interesting by its facilitation for easy visualization.

2.2 Related Work on Fake News Detection

The study of news' veracity started in the early 2010s, known as rumor detection. Pioneering works to detect rumor stress on data extracted from social networks due to the ease of propagating information from them. Castillo [1] took advantage of feature-based methods to assess the credibility of tweets on Twitter. Further along the line, Ma [8] extracted useful features to detect rumors. Those approaches achieved certain success, but heavily relied on feature engineering, which is expensive and time-consuming. Consequently, more recent endeavors using deep neural network were performed to get grid of the need of feature engineering. Ma [7] modeled streams of tweets as sequential data, then used Recurrent Neural Network (RNN) for predicting weather the streams were rumors or not. This approach was proved to yield better results than previous feature-based learning and effective at early rumor detection.

Detection of rumor is related to, but different from, that of fake news. While both try to assess the credibility of news, their focused domains and data have little in common. In fact, research on rumor detection examines the trustworthiness of a group of posts related to a piece of news on Tweeter, while fake news detection works on an independent statement. Furthermore, statements to be studied in fake news detection are not only from social networks, but also from other places such as a public speech, a website, or a news advertisement, whereas posts in rumor detection is limited from social networks only.

To attract the crowd's attention towards fake news, a fake news challenge[2] was launched in 2017 based on the argument that support or disagreement between headline and body text are cues for debunking fake news. That year also witnessed a new direction in researching on fake news detection which focuses on political data, thanks to the introduction of Liar Dataset by Wang [12]. In that work, besides presenting a new benchmark dataset for fake news detection, the authors also proposed a hybrid architecture to solve the task. Their model made use of two components. One is a Convolutional Neural Network (CNN), which was to learn representation for text. The other was another CNN

[2] `http://www.fakenewschallenge.org/`

for meta-data representation learning, followed by a Long Short-Term Memory neural network(LSTM)[3]. Two kinds of representations then were passed into a fully-connected layer with softmax activation function to output the final prediction. Although being complicated with many parameters to be optimized, their model perform poorly on the test set, with only 27.4% in accuracy. Figure 2.13 depicts the hybrid CNN extracted from their original paper [12].

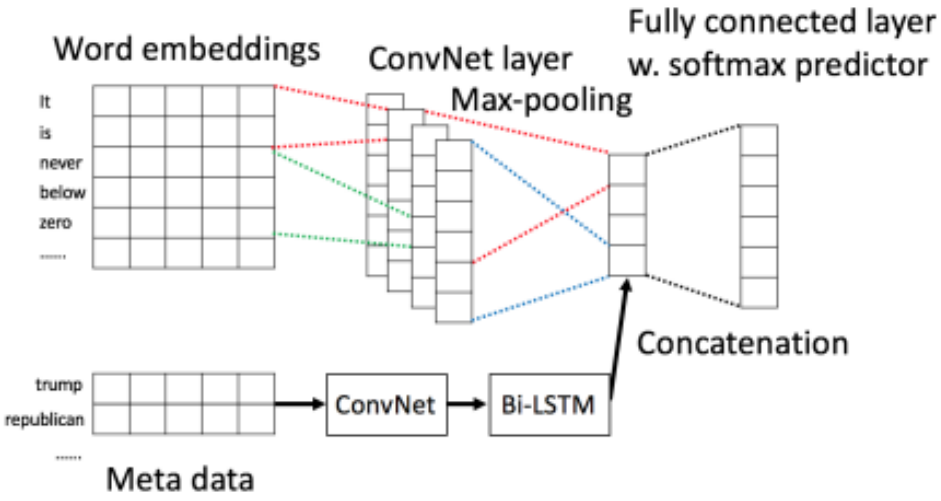


Figure 2.13: Hybrid CNN

Rashkin [9] took a different perspective on detecting fake news by looking at its linguistic characteristics. They employed four types of lexical resources, that are the Linguistic Inquiry and Word Count (LIWC), subjective words with sentiment lexicon, hedging lexicon, and intensifying lexicons crawled from Wiktionary. They tried to examine lexicon's distribution in fakes news and true news so as to discover the difference between the language of truth news and that of fake news. Despite substantial dependence on lexical resources, the performance on political set was even slower than [1], with only 22.0% in accuracy.

Long [6] proposed a hybrid LSTM which exploited two separate LSTMs. Word vectors were fed into the first LSTM, with topic and speaker information being two attention factors. Word vectors were again passed into the second LSTM, and speaker information were also used, but as an additional input rather than an attention factor. The two extracted vectors were then fed into a fully connected layer with softmax function to output the final prediction. Figure 2.14 shows the hybrid LSTM extracted from the original paper of [6]

On the other hand, Volkova [11] works exclusively on data from Tweeter with the main goal is to predict if a news post is suspicious or verified, and classify it into fine-graned

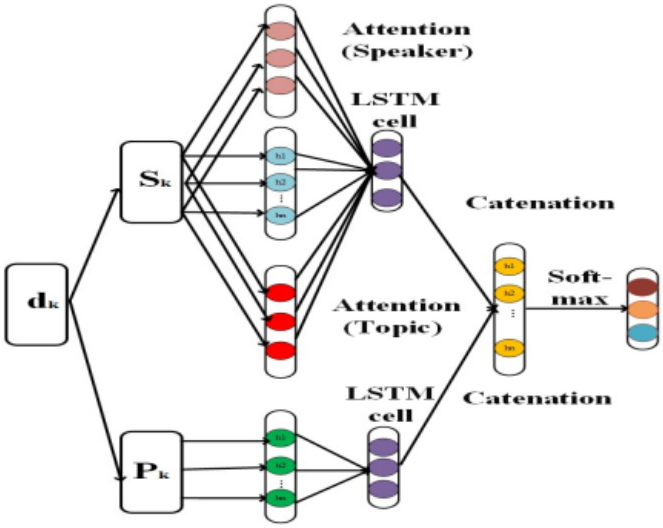


Figure 2.14: Hybrid LSTM

subsets of suspicious news - satire, hoaxes, clickbait and propaganda. The author used linguistic neural networks with linguistic features. The insight from their work is that linguistic feature is relevant for fine-graned classification, whereas syntax and grammar features have little effect.

Chapter 3

Approaches

3.1 Single and Dual attention for fake news detection

3.1.1 Single attention

In this section, we introduce the simplest attention-based approach we take to deal with fake news detection. Inspired by attention mechanism in neural machine translation, image captioning and other tasks, the fake news detection is converted as a problem of determining which subregions of the given statement are the most relevant parts to the task with the help of side information as an attention factor.

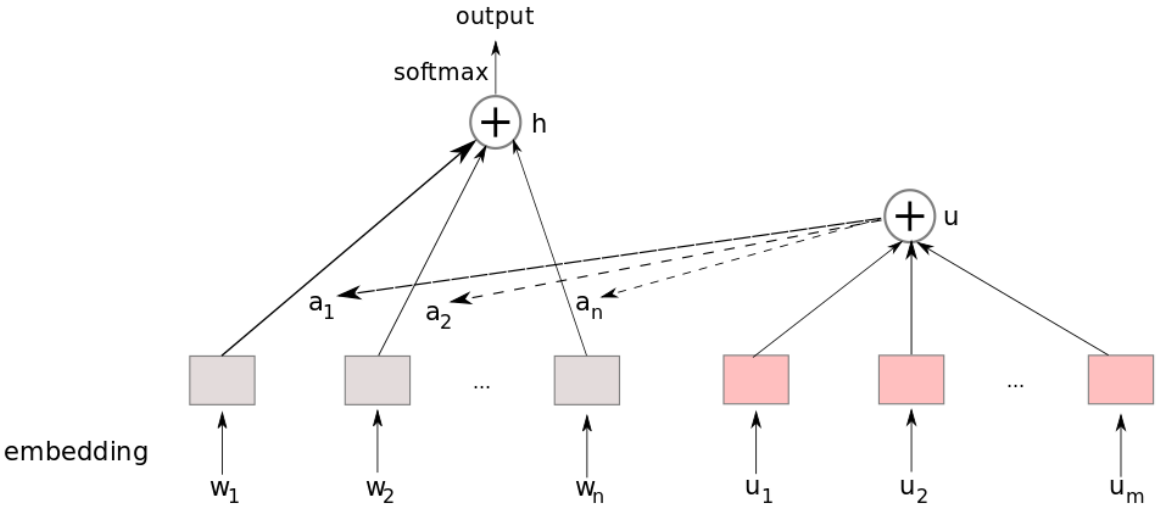


Figure 3.1: Single attention for fake news detection

Regarding input representation, all words and side information are transformed into vectors using embedding matrices. In an attempt to find the impact of side information over

the statement, we convert all side information into a unique vector. In our experiment, we take the easiest approach which simply sums all of the side information vectors. The resulting vector then acts as attention component to help determine which parts of the statement are the most relevant to it, forming a probability vector. We then take the sum of all word vectors in the statement weighted by the probability vector. Figure 3.1 illustrates our single attention.
It is noted that instead of utilizing a deep model such as LSTM or CNN to produce intermediate hidden representation as [12] and [6] did, attention scores are calculated directly from the embedded vectors of words and side information. Hence, there are much fewer parameters to be optimized.

3.1.2 Dual attention

In the previous module, we consider the possible effect of side information on the main text by modeling it as an attention factor. However, it is skeptical that whether the statement also exerts any influence on the side information backwardly. To address this question, we examine the two-way interaction of the statement and its side information. Firstly, the side information acts as the attention component as presented in the previous section. In the reverse direction, the text is represented as an attention factor over the side information. After all, vector outputs from two directions are concatenated to form the final representation. Figure **??** illustrates out dual-attention model.

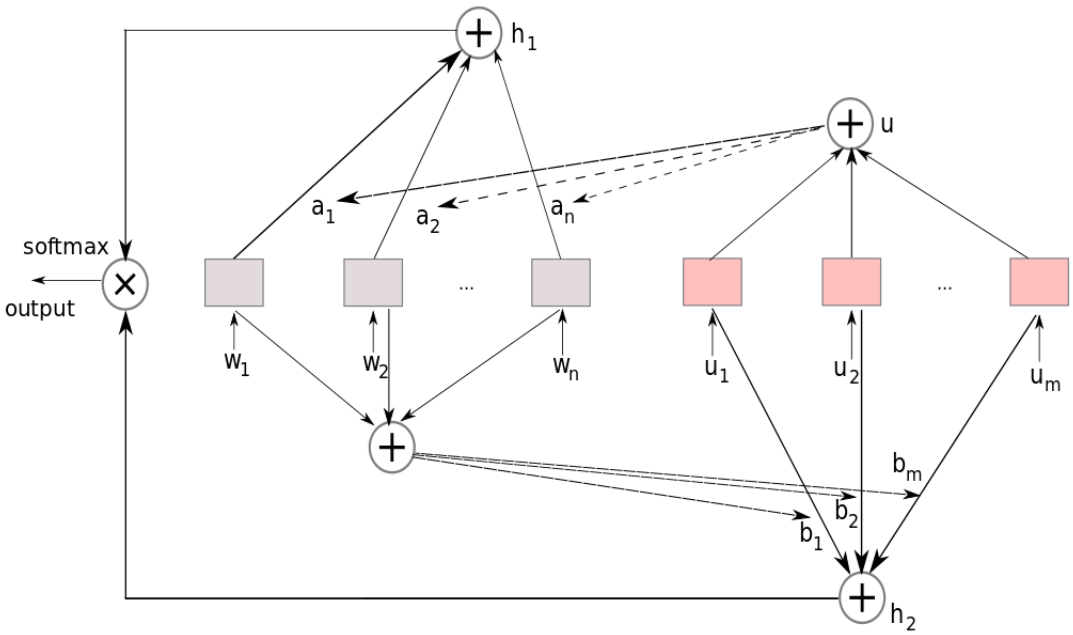


Figure 3.2: Dual-attention model for fake news detection.
"+" symbol means summation while "X" simple mean concatenation

Figure 3.2 demonstrates our dual-attention model. a_1, a_2, .. a_n are attention scores given side information as attention factor. On the backward way, embedded vectors from words are first summed into a unique attention vector. b_1, b_2, .. b_m are attention scores given the attention vector from the statement.

3.2 Word-level Memory Network for fake news detection

Memory network is a kind of attention-based neural network, that shares the core idea of attention mechanism. The original version of memory network using hard attention was introduced by Weston[13]. It was then adjusted by Sukhbaatar[10] with the substitute of hard attention by soft attention so that it can be trained end-to-end with less supervision required. Since then, it has been having many successful applications in a wide range of NLP tasks by virtue of its capability to store external information. In his work, Sukhbaatar demonstrates effective use of memory networks on question answering and language modeling. Das [2] exploited memory networks to perform attention between a considerable number of facts in the mixture of text and knowledge base to solve question answering task. Li [5] used memory networks to find out attitudes towards a set of entities from text. In this section we will describe single layer and multiple layer memory network for word level, and left the char level in the next section.

3.2.1 Single layer Memory Networks for fake news detection

Input memory representation: An embedding matrix A $\in R^{v \times d}$ is used to transform words $\{w_i\}$ in a statement into memory vectors $\{m_i\}$, where v and d are the vocabulary size and embedding size respectively. The associated side information, except for credit history which is already in form of a vector, is also converted into a vector u using another embedding matrix B $\in R^{v' \times d'}$. Unlike the original version of end-to-end memory networks proposed by [10] which computes dot product of w_i and u to find the relevance between them, we employ a different approach by doing aggregation since this allows for difference, hence flexibility in dimensions of embedding matrices A and B.

$$score(m_i, u) = v^T tanh(W_m m_i + W_u u + b) \tag{3.1}$$

Where $W_m \in R^{d \times a}$, $W_u \in R^{v \times a}$, and v $\in R^a$ with a is the dimension of attention vector. Equation 3.1 is interpreted as as a feed-forward network with tanh activation function and W_m, W_u, and b are weight matrices and bias.
Then, softmax function is applied to calculate vector p, normalized matching of w_i and u.

$$p_i = softmax(score(m_i, u)) \tag{3.2}$$

As described in the previous section, this vector is viewed as a probability vector, indicating how much each memory cell contribute to the formation of the output vector.

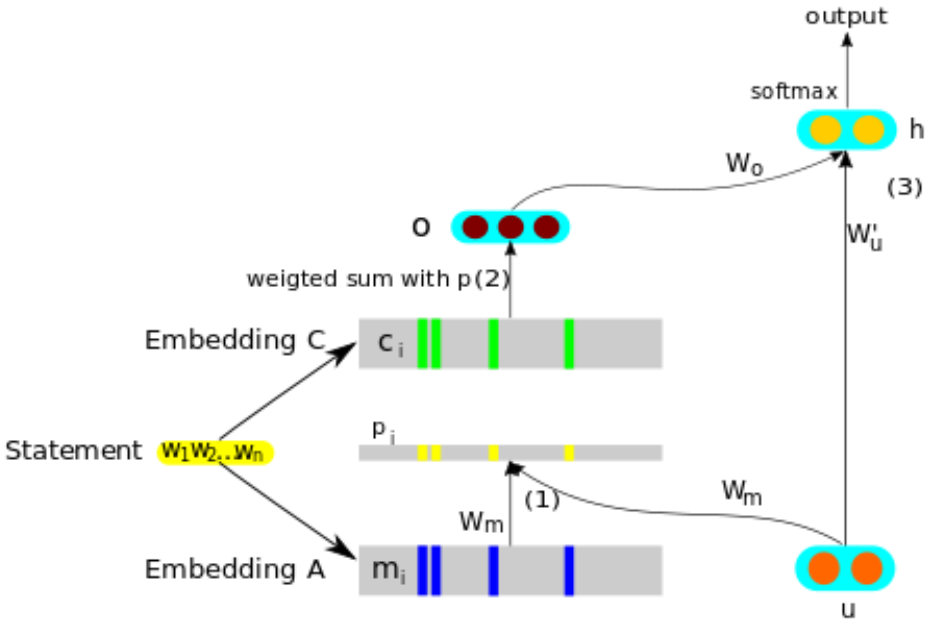


Figure 3.3: Single layer memory networks for fake news detection

Output memory representation: Each w_i is converted into a vector c_i using another embedding matrix C. The output vector o is a weighted sum of c_i by probability vector from the input memory:

$$o = \sum_i p_i c_i \tag{3.3}$$

Generating prediction: since we allow for the difference in dimension of A and C, instead of simply taking sum between o and u we put them into a feed-forward network as follows:

$$h = \sigma(W_o o + W'_u u + b) \tag{3.4}$$

Where $W_o \in R^{d \times d'}$ and $W_u \in R^{d' \times d'}$. Again this is a feed-forward neural network with weight matrices W_o, and W'_u and σ is an activation function, which is rectifier function (**relu**) in our model.
A fully-connected layer (F) is then applied, followed by a softmax layer to generate the final prediction.

$$\hat{y} = softmax(F(h)) \tag{3.5}$$

Cross-entropy is then used as the objective function.

$$L = \sum_i \sum_j y_j log(\hat{y}_j) \tag{3.6}$$

Despite having the similar structure, our proposed memory network is different from end-to-end memory networks by [10] in the receiving input and the way memory vectors are formed. Concretely, the input in [10] is a set of sentences, each of which is transformed into a memory vector using a weighting scheme, while ours is a set of words, each of which is converted into a memory vector directly by looking into an embedding matrix. As such, what we try to learn is external, different representations of words.

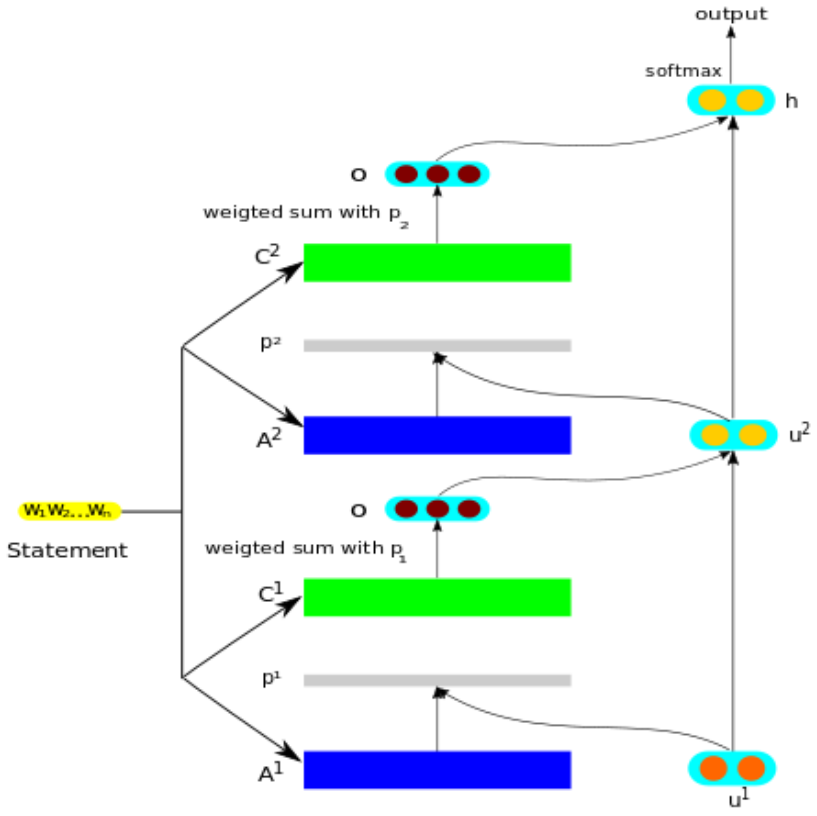


Figure 3.4: Two layer memory networks for fake news detection

3.2.2 Multiple Layer Memory Networks for fake news detection

we extend the model by stacking multiple layers such that output of equation (4) at layer k will be the input u^{k+1} in the next layer.

$$u^{k+1} = \sigma(W_o o^k + W'_u u^k + b)$$

Where W_o, W'_u are weight matrices shared or distinct across layers. W_m and W_u presented in the previous section are also shared or varying through different layers. The rest of the network will be the same of that in single layer. Figure 3.4 illustrates the two layer memory network.
Intuitively, the use of multiple hops or layers is to simulate the update operation of the memory. Suppose say we have two layers. At the first layer, the attention mechanism focus on certain cells in the memory while at the second layer the attention may focus on different cells. The difference in the focus areas indicates that the memory gets update throw multiple computations then the most updated one at the final hop will be extracted to be the final representation.

3.3 Character-level Memory Network for fake news detection

In this module, we aim to explore another variant of memory network with the addition in memory encoding step. In word-level memory network, memory cell is constructed directly from embedding vectors of words. In character-level, however, we experiment with characters independently, then characters of the same words are encoded into a memory cell.

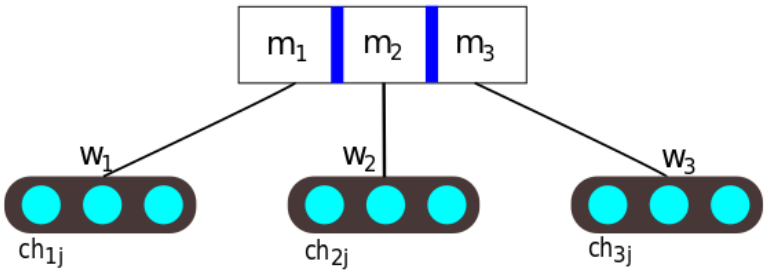


Figure 3.5: 3-cell, char-level memory encoding for fake news detection

Instead of simply taking summation over all characters, we employ position encoding (PE) to take into account the position of characters in a word. Specifically, each memory cell m_i is computed as:

$$m_i = \sum_j l_j * Ax_{ij} \tag{3.7}$$

,where l_j is a column vector computed using the following equation:

$$l_{kj} = (1 - j/J) - (k/d)(1 - 2j/J) \tag{3.8}$$

,with J being the number of characters in a word.

Defined in this way, each value of l denotes a different weight in each position of the given sentence.

Another modification takes place in this module is the score function. Since we aim to examine the effect of all side information other than credit history, there is no need to have different dimensions of memory cell and side information. The dot product is utilized as the score function:

$$score(m_i, u) = m_i * u \tag{3.9}$$

Also, the embedding weight matrix of side information and are the same; both are extracted from with the same vocabulary set. The rest of the model is the same as in that case of word-level setting.

Chapter 4

Evaluation

4.1 Dataset and Preprocessing

We evaluate our model using LIAR dataset crawled from POLITIFACT.COM[1] by Wang[12]. Some examples in the dataset are presented in Table 4.1.

Table 4.1: Examples of statements and side information in the dataset

Statement: Says he won the second debate with Hillary Clinton in a landslide in every poll. **Speaker**: Donald Trump **Topic**: debates, elections, polls **Job Title**: President **State**: New York **Party**: Republican **Credit history**: (63, 114, 51, 37, 61) **Location of speech**: a tweet **Label**: pants-fire
Statement: Each year, 18,000 people die in America because they don't have health care. **Speaker**: Hillary Clinton **Topic**: health-care **Job Title**: Presidential candidate **State**: New York **Party**: Democrat **Credit history**: (40, 29, 69, 76, 7) **Location of speech**: a speech in Des Moines, Iowa. **Label**: true

[1] http://www.politifact.com/

The dataset embodies 12,836 examples divided into separate train, validation, and test set by a ratio of 8:1:1. Each example encompasses a statement and a number of associated side information; that are topic, speaker name, title, party, affiliation, job, speech location and credit history. Credit history holds an account about the numbers of statements a speaker has made in pants-fire, false, barely-true, half-true and mostly-true categories. Table 4.2 shows the distribution of six labels over training, validation, and test set. From the statistics, it can be noticed that the dataset is somewhat imbalanced in that the numbers of pain-fire, barely-true, and true are significantly fewer than those of false, half-true, and mostly-true.

Table 4.2: Distribution of six classes

	pants-fire	**false**	**barely-true**	**half-true**	**mostly-true**	**true**	**total**
Train	**842**	1,998	1,657	2,123	1,966	1,683	10,269
Valid	**116**	263	237	248	251	169	1,284
Test	**92**	250	214	267	249	211	1,283

We also deal with two-label setting. In fact, pants-fire, false and barely-true are grouped into false, while half-true, mostly true, and true are grouped into true. Statistics of the two label is showed in Table 4.3.

Table 4.3: Distribution of two classes

	true	**false**
Train	4,497	6,072
Valid	616	668
Test	556	727

Although the statements in the dataset are politics-related only, they are not merely from democrats and republicans, but also from other non-politic individuals or organizations. Table 4.4 shows details of speaker statistics.

The data is processed as follows:

- For statements, each is tokenized using NLTK[2], then stopwords and punctuation are removed. All money characters are converted into one token, so are percentage and number.

[2] `http://www.nltk.org/`

Table 4.4: Speaker Affiliations

Speaker Affiliations	
Democrats	4,450
Republicans	5,687
None(FB post, Tweets, etc.)	2,185

- For credit history, it is normalized to be a 5-dimensional vector, each value ranges within 0 and 1. Specifically, vector (70, 71, 160, 163, 9) is converted into (0.281, 0.285, 0.642, 0.654, 0.036).
- Topic and location are treated as sequence of words each. After being converted into a embedded vector, the are summed into one unit vector.
- Speaker name, state, and job title are treated as one token before doing embedding transformation.
- Special tokens are introduced to facilitate the computation in our models. Specifically, words absent from the vocabulary set are converted them into <UNK>. Also, since the numbers of words in statements are varying, <PAD> token are added to derive the statement having the same size. The same padding strategies are also applied in case of character-level model.
- Labels are represented as one-hot vector. For example, vector output of paints-fire is (1, 0, 0, 0, 0, 0). In case of 2-label setting, vector output of false and true are (1, 0) and (0, 1) respectively.

4.2 Experimental settings

We compare our memory network (MM) against the following baselines:

- CNN-WangP: a hybrid CNN using side information by [12]. In their model, one CNN is used to captures text representation, and CNN-LSTM is used for side information representation learning. Their CNN based on CNN for text by Kim[4]
- LSTM-L: a hybrid LSTM using two LSTMs by [6]; one takes as input a statement and a type of associated side information, while the other takes as input a statement only, but with topic and speaker information as attention components. The performance of baseline models was displayed in Table 4.5.

For our model, word embeddings were initialized randomly, are learned and gets updated during the training process. After all, embedding matrices with 50 dimension were used. Special tokens such as <PAD> and <UNK> are initialized using a uniform distribution

Table 4.5: Accuracy of baseline models for 6-label classification (%)

Method	**Dev**	**Test**
Majority	20.4	24.7
CNN-WangP	24.7	27.0
LSTM-L	40.7	41.5

in range [0.1, 0.1]. We strictly turned all hyperparameters on dev set and observe the best result based on accuracy score. Details of our configuration is described in Table 4.6.

Table 4.6: Configuration of Experiments

Hypeparameters	**Value**
Batch size	64
Word embedding size	50
side information size	32
Character embedding size	20
Attention size	32
Learning rate	0.25
Max statement length	30
Dropout	0.8
Optimizer	Adadelta[14]

Evaluation metrics: We use accuracy, precision, recall, and f1 as evaluation metrics (tp, fp, fn in the following equations are true positive, false positive and false negative respectively).

- Accuracy is a measure calculated as the ratio of correct predictions over the total number of examples.
- Precision is to measure the percentage of positive predictions that are correct and is defined as:

$$Precision = \frac{tp}{tp + fp} \tag{4.1}$$

- Recall is to measure the percentage of correct predictions the classifier catch and is defined as:

$$Recall = \frac{tp}{tp + fn} \tag{4.2}$$

- F1 is to find the balance of recall and precision and is computed as:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

Fp, tp, fn are false positive, true positive, false negative. Since they are well known in natural languge processing, we skip details of those concepts.

Implementation tools: all the source codes are written in Python 2.7[3] using deep learing library Tensorflow[4].

4.3 Results

4.3.1 Single and Dual attention

Table 4.7 demonstrates the performance of single and dual attention neural networks for the fake news detection task. Despite being far from the state-of-the-art, applying single attention derives better results than the hybrid CNN. In fact, accuracies of our single attention neural network are 4% and 0.2% higher than those of the hybrid CNN in dev set and train set respectively. Another advantage of our model over the hybrid CNN is that the structure of our network is much simpler than that of the hybrid CNN. Remember that the hybrid CNN employs two separate CNNs and a LSTM, while we do not use any deep model as an intermediate layer, therefore our model has significantly fewer parameters to be learned. In short, our single attention model has better results than the hybrid CNN despite its simplicity.

Table 4.7: Single and Dual attention for 6-class setting (%)

	Single		Dual	
	Dev	Test	Dev	Test
Acc	28.7	27.6	40.5	37.3
Pre	38.0	27.5	42.8	39.6
Rec	27.2	25.2	39.4	37.6
F1	26.9	24.3	40.4	38.2

Having better results than the hybrid CNN, the single attention neural network is still far from the dual one. Further along the line, the dual attention neural network helps improve the single one by 11.8% on dev set and 9.7% on test set. The same trend takes place in precision, recall, and f1 score. Specifically, there are 4.8% and 12.1% rise in precision, 12.2% and 12.4% in recall , 13.5% 13.9% and in f1 on dev and test set respectively.

Compared to the state-of-the-art, the dual neural network is still having some gap to reach. Having said that, the results are quite comparable. There are 0.2% and 4.2% improvements needed in the dev and test set respectively to be in pair with the-state-of-the-art.

[3]https://www.python.org/
[4]https://www.tensorflow.org/

The same pattern occurs in the 2-class setting. It is demonstrated from Table 4.8 that the two-way interaction between statements and side information considerably promotes the performance of the classifier. Particularly, accuracies of dual attention neural network are 5.6% and 3.5% higher than the single one in the dev and test set respectively. Obviously, dual neural attention shows its significant contributions in both 6-label and 2-label classification.

Table 4.8: Single and Dual attention for 2-class setting (%)

	Single		Dual	
	Dev	Test	Dev	Test
Acc	63.9	66.6	69.5	70.1
Pre	64.2	66.0	69.8	69.6
Rec	63.5	65.0	69.1	68.8
F1	63.3	65.1	69.0	69.0

4.3.2 Word-level memory network

The effect of each side information

It is indicated from Table 4.9 that credit history is the most informative factor in detecting fake news, which is consistent with the finding by Long [6]. Other types of side information produce values that are far lower than those by credit history. The low values of precision, recall, and f1 scores when using side information other than credit history were because our model failed to give correct predictions for paint-fire, barely-true, and true. This could be explained from imbalance nature of the dataset since the numbers of examples in these categories are fewer than the others. Comparisons between our model and the baselines regarding precision, recall and f1 is impossible because those values are unavailable from those baseline models.

Moreover, our proposed memory network using credit history only (MM+ch) already outperformed hybrid LSTM incorporating all side information with attention by [6] by 6.7% and 2.7% accuracy score on dev and test set respectively. Results from Table 4.10 also confirm the dominant contribution of credit history, when all evaluation scores rose by more than 10%. Speaker name (sp) and party (pa) information came in the second place on dev and test set. Again, values for 2-label classification from previous works are unavailable for comparisons.

Form Table 4.11, it is noticed that incorporating more types of side information boosts the performance. Specifically, when combining credit history with party information, accuracy scores increased by 1.7% and 2.2% on the dev and test set respectively. Likewise, precision

Table 4.9: Single Layer Memory network (MM) using one type of side information for 6-label classification(%). Sp, tp, jb, st, lc, pa, and ch stand for speaker name, topic, speaker job, state, location, party, and credit history respectively.

	MM+sp		MM+tp		MM+jb		MM+st		MM+lc		MM+pa		MM+ch	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Acc	25.2	25.4	24.7	25.7	25.3	24.2	26.1	24.9	25.5	24.6	27.0	24.2	**47.4**	**44.2**
Pre	16.8	22.8	22.4	21.5	17.9	17.1	21.0	20.5	20.7	21.6	13.5	32.1	**54.7**	**53.7**
Rec	20.6	20.7	21.9	22.2	21.3	20.5	22.7	21.4	21.6	20.9	22.7	20.4	**44.3**	**43.5**
F1	15.9	16.6	19.3	19.8	16.4	15.8	20.0	19.0	17.9	17.7	16.8	15.5	**43.1**	**42.1**

Table 4.10: Single Layer Memory networks (MM) using one type of side information for 2-label classification (%)

	MM+sp		MM+tp		MM+jb		MM+st		MM+lc		MM+pa		MM+ch	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Acc	59.4	64.1	62.6	62.4	60.2	63.8	61.8	62.8	62.7	62.3	61.6	62.6	**73.8**	**74.4**
Pre	59.2	63.3	63.0	61.5	60.2	62.9	61.9	61.9	63.0	61.3	61.8	61.6	**74.0**	**74.0**
Rec	59.1	63.2	62.1	60.3	59.9	62.6	61.3	61.3	62.3	60.6	61.1	61.0	**73.6**	**73.7**
F1	59.0	63.2	61.7	60.2	59.8	62.7	61.0	61.3	62.0	60.7	60.9	61.0	**73.6**	**73.8**

scores went up by 2.4% on the dev set and 1.4% on the test set. Recall scores witnessed the same trend with the rise of 2.9% on the dev set and 2.5% on the test set. The F1 score was also improved by 4.6% on the dev set and 3.7% on the test set. State and party information perform somewhat better than speaker and job information. Similarly, in 2-label classification, it seems that credit history delivers the most needed information that adding others (eg. MM+ch+st or MM+ch+pa) did not help. Therefore, results for those cases are not presented either.
Overall, MM+ch+pa produced accuracies higher than that of the state-of-the-art, which are by 8.4% and 4.9% on the dev and test set respectively.

The effect of multi-layer memory networks

Unfortunately, stacking our memory networks with more layers does not further improve the performance in our task. Table 4.12 shows that the performance of using two-layer memory network was lower than that of using one layer on all evaluation metrics when using only credit history (MM2+ch). However, in case of using both credit and party information, our two layer (MM2+ch+pa) and three-layer memory networks (MM3+ch+pa) nearly reached the single layer one (MM+ch+pa). This demonstrated that external learned information is still relevant in this case.

Table 4.11: Single Layer Memory networks with combined side information for 6-class setting (%)

	MM+ch		MM+ch+sp		MM+ch+jb		MM+ch+st		MM+ch+pa	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Acc	47.4	44.2	47.9	45.4	48.4	44.8	**49.1**	45.9	**49.1**	**46.4**
Pre	54.7	53.7	53.0	51.5	53.0	51.0	56.5	52.4	**57.1**	**55.1**
Rec	44.3	43.5	46.1	45.3	46.6	44.7	47.2	45.5	**47.2**	**46.0**
F1	43.1	42.1	46.7	45.6	47.3	44.6	**48.0**	45.5	47.7	**45.8**

Table 4.12: Single-layer (MM) and two-layer (MM2) and three-layer memory network(MM3) for 6-label classification in comparison.

	MM+ch		MM+ch+pa		MM2+ch		MM2+ch+pa		MM3+ch+pa	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Acc	47.3	44.2	**49.1**	**46.7**	40.5	38.7	48.8	46.2	**49.1**	45.9
Pre	50.2	49.7	**56.7**	**55.4**	31.6	26.2	54.9	53.5	54.1	52.6
Rec	44.6	43.6	47.3	**46.3**	34.4	33.2	47.3	46.1	**47.4**	45.9
F1	44.0	42.0	47.7	**46.1**	29.8	28.6	47.6	45.7	**47.7**	45.8

4.3.3 Char-level memory networks for fake news detection

It is shown in Table 4.13 and 4.14 that the char-level memory network performs worse than the word-level one for all settings. Concretely, in 6-label classification, cMM combined with any side information yeild lower results than those from their MM counterpart. The same trend occurs in the 2-label classification task. We speculate that the decoding scheme is not helpful. Rather, it may hinder the classifier from predicting correctly. Due to the lack of time, we do not have results for other side information.

Table 4.13: Char-level Single Layer Memory network (cMM) using one type of side information for 6-label classification(%)

	cMM+sp		cMM+st		cMM+pa	
	Dev	Test	Dev	Test	Dev	Test
Acc	25.6	23.9	23.4	22.1	25.3	26.4
Pre	17.7	15.1	20.1	14.7	21.0	20.2
Rec	21.7	20.2	19.7	18.5	21.2	22.1
F1	18.1	16.5	14.7	13.8	15.9	16.7

Table 4.14: Single-Layer Memory network (MM) using one type of side information for 2-label classification(%)

	cMM+sp		cMM+st		cMM+pa	
	Dev	Test	Dev	Test	Dev	Test
Acc	54.6	58.1	54.9	57.9	56.5	57.5
Pre	56.4	56.7	55.7	56.1	57.9	55.5
Rec	53.2	53.4	53.7	53.8	55.4	53.9
F1	47.1	49.0	49.8	50.8	52.3	51.9

4.4 Discussion

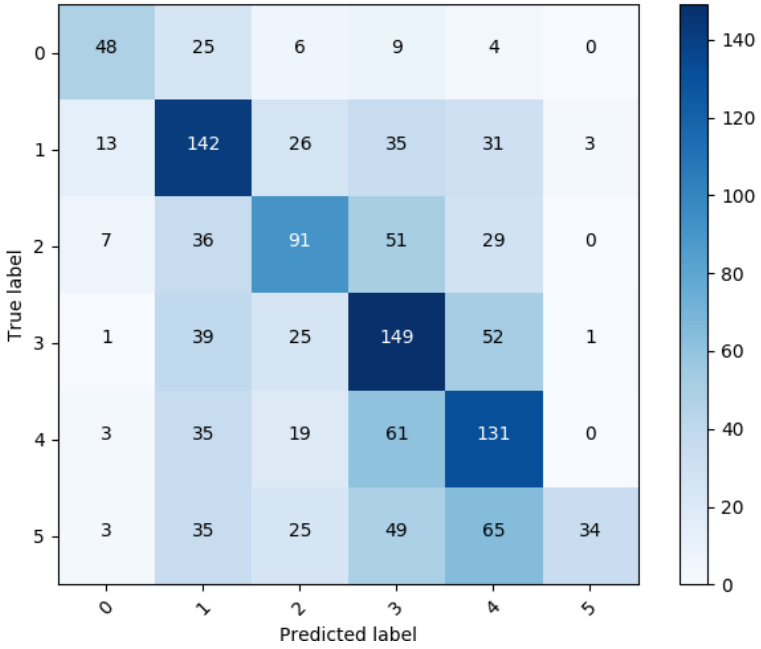


Figure 4.1: Confusion matrix of our best model (MM1+ch+pa)

From our best model (word-level memory network with credit history and party information), we observe several phenomena.

It is difficult to distinguish paint-fire from false. For 6-label classification, our model predicts correctly 48 out of 92 instances with *pants-fire* label, but misclassifies 25 other instances as *false* label, which accounts for 27% of wrong predictions. The reason may come from the language use, especially strong determiners such as every or any. For example, in the statement "*Says he won the second debate with Hillary Clinton in a landslide in every poll.*" Justification from POLITIFACT.COM tells us that not

only did Trump not win by a landslide in any of the polls, he didnt win any of the polls[5]. It is obvious that the statement is false, but it is rated as pants-fire to stress the over exaggeration of the lie. However, our model is unable to recognize that emphasis. Similarly, in the statement "*This town (Wilmington, Ohio) hasn't taken any money from the government,. They don't want any money from the government*", the mention of *any* is so subtle that our model fails to classify it as pants-fire, but as false instead. Figure 4.1 shows the confusion matrix by our best model.

Table 4.15: Examples of correct predictions on 2-label, but wrong in 6-label classification.

ID	Statement	For 6-label		For 2-label	
		Predict	**Truth**	**Predict**	**Truth**
1	Says he won the second debate with Hillary Clinton in a landslide in *every* poll.	false	pants-fire	false	false
2	This town (Wilmington, Ohio) hasn't taken *any* money from the government. They dont want *any* money from the government.	false	pants-fire	false	false
3	The Fed created $1.2 trillion out of nothing, gave it to banks, and some of them foreign banks, so that they could stabilize their operations.	mostly-true	true	true	true
4	Texas families have kept more than $10 billion in their family budgets since we successfully fought to restore Texas sales tax deduction a decade ago.	mostly-true	true	true	true
5	Says the unemployment rate for college graduates is 4.4 percent and over 10 percent for noncollege-educated.	half-true	true	true	true
6	Each year, 18,000 people die in America because they don't have health care.	mostly-true	true	true	true

Likewise, separating true from half-true and mostly-true is challenging. In 2-label setting, our model successfully predicts only 34 out of total 211 examples to be true, and misclassifies 65 and 49 others to be mostly-true and true respectively. We observe

[5]http://www.politifact.com/wisconsin/statements/2016/oct/12/donald-trump/donald-trumps-ridiculous-claim-all-polls-show-he-w/

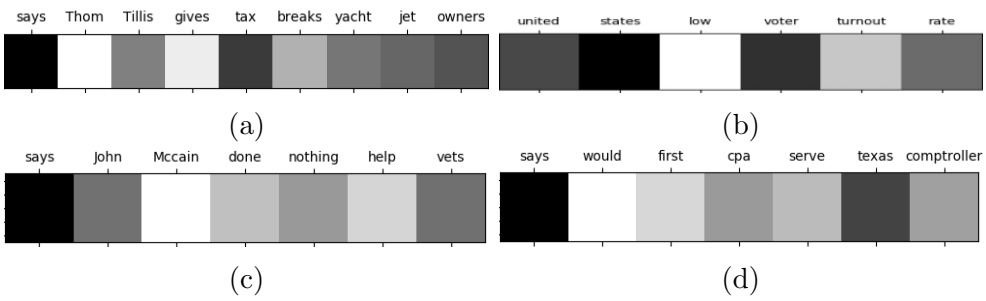


Figure 4.2: Visualizations of attention weights. (a), (b), (c), (d) represent statements of 1,3,2,4 in Table 9 respectively

that reference to numbers, percentages, and money is exploited as a factor to mix false opinion into a true story. In particular, the statement "*However, it took $19.5 million in Oregon Lottery funds for the Port of Newport to eventually land the new NOAA Marine Operations Center-Pacific*" in the training set in which an amount of money also appears is annotated as half-true since the statement mixes true number of $19.5 million and the misleading place where the money went to. On the other hand, our model misclassified statements 3, 4, 5, and 6 in Table 7 which share the same pattern of money or number reference.

Table 4.16: Examples of correct predictions in both 6-label and 2-label setting.

ID	**Statement**	**For 6-label**		**For 2-label**	
		Predict	**Truth**	**Predict**	**Truth**
1	Says Thom Tillis gives tax breaks to yacht and jet owners.	pants-fire	pants-fire	false	false
2	Says John McCain has done nothing to help the vets.	pants-fire	pants-fire	false	false
3	The United States has a low voter turnout rate.	true	true	true	true
4	Says he would be first CPA to serve as Texas comptroller.	true	true	true	true

Figure 4.2 illustrates attention weights the proposed memory model generates for statement 1, 3, 2, and 4 in 4.16. We realize that our model give focus on proper nouns (Thom in Figure 4.2a and McCain in Figure 4.2c and verb *give*. Therefore, it seems that dealing with name entities and verbs is promising for this task. Language use is also a cue for detecting fake news. In fact, Figure 4.2b shows that adjective *low* is given strong attention, while Figure 4.2d reveals that the attention is put largely on model verb *would*.

Chapter 5

Conclusion and Future Work

We provide various approaches towards fake news detection. Of all the proposed models, the word-level memory network obtains the best results in both binary and 6-label setting. Our word-level memory network takes advantage of attention mechanisms to focus on the most relevant subparts of a given input as well as storing external information by the network itself. Experimental results demonstrate that the additionally stored information is helpful for the task. Moreover, dealing with fine-grained labels is difficult as neighboring labels are so confusing to be recognized correctly. Overall, our model outperforms the current state-of-the-art by 8.4% and 4.9% on dev and test set respectively.

Besides, it is recognized from the comparisons between single and dual attention neural networks that mutual interaction between a statement and its connected information is relevant to the task. Also, not all kinds of side information make an equal contribution to detecting fake news. The performance primarily comes from credit history, and is improved more with the presence of other side information.

Future work will take into account other traits of fake news such as linguistic, temporal, or propagating features. For linguistic features, it is viable to consider named entities and part-of-speech. In addition, propagation of fake news via a number of people can be modeled as a graph structure. In this case, deep learning on graph is a good candidate to tackle fake news detection.

Bibliography

[1] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.

[2] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*, 2017.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[5] Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 671–680. ACM, 2017.

[6] Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 252–256, 2017.

[7] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.

[8] Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM, 2015.

[9] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2921–2927, 2017.

[10] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[11] Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653, 2017.

[12] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[13] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[14] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.