Statistical Modeling Of Stance Detection

by

Borislav Mavrin

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Statistical Machine Learning

Department of Mathematical and Statistical Sciences
University of Alberta

Abstract

In recent years fake news has become a more serious problem. This is mainly due to the popularity of social networks, search engines and news aggregators that propagate fake news. Classifying news as fake is a hard problem. However it is possible to distinguish between fake and real news, by considering how many related tweets agree/disagree with the news. Therefore, in the simplest case the problem can be reduced to identifying whether a given tweet agrees with, disagrees with or is unrelated to the news in question. In general this problem is referred to as 'stance detection'. In machine learning terminology this is a classification problem. This thesis investigates more advanced Natural Language Models, such as matching Long Short Term Memory model and soft attention mechanism applied to stance detection problem. The ideas are tested using a publicly available data set.

Acknowledgements

This work would not have been possible without the help of the people which I would like to thank.

My supervisor Dr. L Kong taught me how to be an independent researcher. He guided me but at the same time gave me room to make decisions on my own. He also made my studies possible through his generous financial support and believed in my ability to handle two concurrent internships.

I would like to thank my co-supervisor Dr. Di Niu for teaching me to be attentive to details and not to give up on hard problems.

I also would like to express my gratitude to the thesis examination committee members: Dr. Ivan Mizera, Dr. Adam Kashlak and Dr. Guozhong Zhu.

My family provided me with invaluable support. My wife listened with patience when I was excited and talked about my research and machine learning.

As Newton said: 'If I have seen further it is by standing on the shoulders of Giants' [1]. My work would not have been possible without recent breakthroughs done by deep learning community.

[1] Isaac Newton (1642-1727) In a letter to Robert Hooke in 1675.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Fake news problem / Motivation

Social networks have become an important source of information in recent years, especially with respect to news. Such networks generate a firehose of data which attracts researchers. Twitter seems to be one of the most popular social networks among researchers. The reason might be a clear and simple structure of the data:

- each tweet has the maximum length of 140 characters
- users can follow other users.

The amount of research papers that use datasets from Twitter is vast and the review of this body of research deserves a lengthy paper in its own. Here I give a very short survey of papers related to the topic of the thesis. Results in [1] provide some descriptive statistics based on a small sample of around 2,000 tweets. A more in-depth graph theoretical analysis based on all Tweets

as of July 2009 is presented in [25].

As it was shown in [36] clustering tweets provides the means of extracting news from Twitter. Breaking news detection was considered to be the next challenge at that time. Recently, a case study published in [29] showed that it is possible to detect breaking news faster than traditional journalism [2].

The veracity of news is another important issue and still remains a challenge. In the case of news detection we know that unsupervised statistical machine learning methods such as clustering allows us to detect news. Simply put, a breaking news is a new cluster. That is the data set of tweets is self-contained in the sense that it contains all the sufficient information for news detection. The main problem in identifying fake news however, is that fake news usually does not contain sufficient information for revealing itself as fake. It needs some external information or some indirect measures of veracity.

An important study of this problem was conducted in [30]. In this paper authors came up with an indirect (but easy to interpret) measure of veracity. They analyzed the dynamics of Twitter activity right after the Chilean earthquake in 2010 [3].

In [30], authors tried to see the difference in propagation of true facts versus false rumours. Researchers came up with the hypothesis that the stance/sentiment of a tweet about the news can measure the veracity of that news. To test the hypothesis authors collected a data set that contains 7 cases of confirmed truths and 7 cases of false rumours. The veracity of each case

[2] Authors also claim their production proprietary system has fake news detection module, although they do not disclose any details.

[3] According to Wikipedia it is considered to be the seventh strongest earthquake in history. NASA also believes that Earth's axis shifted causing a permanent shortening of day.

was validated by reliable sources. An example of a confirmed truth: 'The international airport of Santiago is closed'. For each case researchers manually collected unique tweets. By unique tweet we mean a tweet posted by a user without quotes from other tweets (the opposite of a re-tweet). The number of tweets per one case ranges from 42 to 700. The next step was to classify each tweet into the following categories:

- tweet confirms the news
- tweet denies the news
- tweet asks questions about the news to gather more information
- tweet is indifferent or difficult to classify.

It is worth mentioning that classification was done by humans not by statistical machine learning or classical Natural Language Processing (NLP) methods. That makes results reliable and does not raise questions about the accuracy of classification. The confirmed truths get over 95% of approval [4] whereas false rumours get only 45%. Given the size of the sample and the methodology (classification by humans) the result is significant. Hence, it can be concluded that detection of fake news can be reduced to stance detection of related messages, at least in the case of Titter.

Fake news detection problem is an interesting research problem in Natural Language Processing on its own. In addition, fake news is becoming a serious problem of social networks nowadays. The coverage and the speed of propagation of information in social networks made social networks an important part

[4]The percentage of tweets that confirm the news.

of mass media. However, the news generation process here is not centralized and, therefore, the news is not validated. Identifying the veracity of news by humans is costly. Thus statistical machine learning methods can significantly reduce the cost.

1.2 Pre Neural Networks models / Classical Natural Language Processing methods

Stance detection (SD) as a separate subfield seems to be younger than a closely related field of Recognizing Textual Entailment (RTE). Given a pair of sentences RTE is the classification of the relation between two sentences as:

- sentences contradict each other
- first sentence implies the second
- sentences are unrelated.

SD on the other hand is the classification of a pair of sentences as:

- second sentence confirms/favours the first
- second sentence denies/is against the first
- sentences are unrelated.

It can be seen that the problems are quite close and it is reasonable to conjecture that classification models should have similar approach if not the same.

RTE has been a challenging task and is recognized as an important building block in other NLP systems, like text summarizations and information extraction [11].

A lot of effort was put into explicit feature engineering approaches. Such hand crafted features require expert knowledge in natural language processing. In [26] researchers used features based on:

- overlap of sentences on the level of words
- sentences' alignment
- negation detection
- semantics similarities of parts of sentences that do not overlap

The novelty of [19] lies in the application of soft cardinality [20] in the similarity measure as opposed to a more conventional Jaccard similarity [18] which is based on the mathematical cardinality of a set. When dealing with hand crafted features in Natural Language Problems, a common task is to measure the similarity of sentences or some features derived from sentences.

Definition 1.2.1. Jaccard similarity. For any two sets A and B

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1.1}$$

In [19] authors used soft cardinality instead of mathematical cardinality $|.|$.

Definition 1.2.2. Soft cardinality.

$$|A|_{sim} = \sum_{i=1}^{|A|} \frac{1}{\sum_{j=1}^{|A|} sim(a_i, a_j)^p} \tag{1.2}$$

where

$$A = \{a_1, a_2, \ldots a_{|A|}\}, \quad p \geq 1 \tag{1.3}$$

and $sim(.,.)$ satisfies the following conditions:

$$for \quad i \neq j \quad sim(a_i, a_j) \in [0, 1) \tag{1.4}$$

$$for \quad i = j \quad sim(a_i, a_j) = 1 \tag{1.5}$$

It can be seen that as $p \to \infty$ $|A|_{sim} = |A|$.

The approach taken in [40] is the combination of a large number of hand crafted features. In total researchers used 72 features grouped into following categories:

- Length features. 16 different measures related to the length of sentences.
- Surface text similarity. 10 sentence similarity measures.
- Semantic similarity. 6 similarity measures based on meaning of words.
- Grammatical relationship. 8 measures of similarity on grammatical level.
- Text difference measures. 13 various handcrafted features based not on the overlapping part of sentences but rather on its difference.
- String features. 13 similarity measures on the character level.
- Corpus-based features. N-gram and vector space features, 6 features in total.

Using those features authors applied 5 standard classifiers:

- SVM
- Random Forest
- Gradient Boosting
- kNN
- semi-supervised learning with kNN

The best model achieved high accuracy compared to other models at that time [5]. The work [40] does not have any novel approaches, but rather uses a combination of many existing techniques.

Slightly different approach of Markov Logic Networks applied to RTE was explored in [7]. Authors combined logic-based sentence representations with vector space model in order to learn features derived from both sentence structure and meaning of separate words.

Recent breakthroughs in deep neural NLP led to new approaches in both RTE and SD. In [35] authors applied ideas of conditional RNN encoding and attention mechanism for RTE problem. Another model similar to that in [35] but with an interesting modification of LSTM presented in [39].

The recent model described in [4] achieved state of the art performance in stance detection task of SemEval 2016 [33] data set. That model seems simpler than contemporary models from RTE. It is basically a bidirectional conditional RNN encoder. It should be noted that new data set specifically for stance detection appeared quite recently [12]. It should help advance the research in this field.

[5] around 83 % of 5 fold cross-validated accuracy.

Chapter 2

Methods

2.1 NLP from scratch

It can be seen from Section 2.1 how much effort and prior knowledge is required for hand crafted feature engineering. However, recent advancements in Neural Networks show that automated feature extraction is possible. It is worth quoting a highly cited paper [10] from this field:

> We propose a unified neural network architecture and learning algorithm that can be applied to various natural language processing tasks including: part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This versatility is achieved by trying to avoid task-specific engineering and therefore disregarding a lot of prior knowledge. Instead of exploiting man-made input features carefully optimized for each task, our system learns internal representations on the basis of vast amounts of mostly unlabeled training data.

2.1.1 Statistical language models/distributed representation for words

A lot of tasks in NLP require a good representation of semantics. For instance, a good feature representation of word should preserve its meaning, such as 'closeness' in meaning – i.e. words 'cat' and 'kitten' should be 'closer' in meaning than 'cat' and 'horse'. It turns out that the model that quantifies this idea is the statistical language model. In order to give a formal definition of a statistical language model we need some preliminaries.

Definition 2.1.1. Token. The smallest indivisible lexical unit.

Definition 2.1.2. Text. An ordered set of tokens in a natural language.

A text can be a book, a set of Wikipedia articles and the token can be a character, a word or a whole sentence. In the thesis the token is a word.

Definition 2.1.3. Dictionary. Given a text in natural language, a dictionary is an ordered set of unique tokens contained in text.

Now we can introduce the notion of statistical language model.

Definition 2.1.4. Statistical language model. Given a dictionary D, a statistical language model is the joint probability density over sequences of tokens with elements from the dictionary:

$$\forall\{w_1, w_2, \dots w_T\} \quad s.t. \quad w_t \in D, \quad P(w_1, w_2, \dots w_T) \in \mathbb{R}. \tag{2.1}$$

Simply put a 'good' statistical language model should assign higher probability to natural sentences. For example we expect the following relation to

hold:

P('The quick brown fox jumps over the lazy dog.') ¿ P('The dog quick brown lazy jumps fox over the.')

It is easy to see that by factoring the joint probability statistical language model can be rewritten as:

$$P(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} P(w_t | w_1, w_2, \dots w_{t-1}) \tag{2.2}$$

Given that equivalent formulation a good statistical language model should estimate conditional probabilities in such a way that sampled sentences from the model mimic the sentences from the text that was used for estimation. For example given the sequence of tokens,

'The quick brown fox jumps over the lazy',

a good model will generate 'dog' as the next token, whereas a bad model will generate 'lazy' which won't make the resulting sequence into a sensible sentence.

Estimating conditional probabilities from 2.2 might be difficult, since the longer sequences are less frequent and hence the bias for such estimators will be high. Therefore, a quite common approximation is an n-gram model:

Definition 2.1.5. N-gram language model.

$$\forall \{w_1, w_2, \dots w_T\} \quad s.t. \quad w_t \in D, \quad P(w_1, w_2, \dots w_T) \in \mathbb{R} \tag{2.3}$$

where

$$P(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} P(w_t | w_1, w_2, \dots w_{t-1}) \tag{2.4}$$

with

$$P(w_t|w_1, w_2, \dots w_{t-1}) \approx P(w_t|w_{t-n+1}, \dots w_{t-1}). \tag{2.5}$$

The idea is quite intuitive: consider only n-1 last tokens, that is n-1 order Markov property. However estimation of these probabilities (by counting measure) raises at least three important issues:

1. Model will assign zero probability to sequences longer than n

2. Model will not generalize to sequences it has seen in the training data.

3. Curse of dimensionality. A better model will require larger data sets and the number of n-grams will grow.

A major breakthrough was the distributed representations of words [8]. The main is idea to learn smooth feature representations of words. Smoothness here means the following: slight changes in the meaning of a word in a sentence should cause a slight change in the probability assigned to that sentence.

The model proposed in [8] consists of the following parts:

- Given a sequence of words (for simplicity we can think of a sentence except the last word):

$$w_1, w_2, \dots w_{T-1}, \quad s.t. \quad w_t \in D \tag{2.6}$$

  where D is a dictionary, the model maps each w_t into $\mathbb{R}^m$ with $m \ll |V|$. V is a $|V|$ dimensional vector space such that each element is a standard basis vector (so called 'one hot encoding'). The mapping: $C : V \to \mathbb{R}^m$

is linear, $C(w_t) = w_t C$, where C is a $|V| \times m$ matrix. Matrix C is usually referred to as an embedding matrix.

- The score $y \in \mathbb{R}^{|V|}$ is the output of the one layer feed-forward neural network:

$$y = U \tanh(Hx + d) \tag{2.7}$$

where

$$x = Vec([Cw_1, Cw_2 \dots Cw_{T-1}]) \tag{2.8}$$

is the concatenation of the word feature vectors, H is a $h \times m(T-1)$ matrix, $d \in \mathbb{R}^h$ and U is $|V| \times h$ matrix. tanh is applied element-wise.

The probability distribution is obtained from scores through the softmax:

$$\hat{P}(w_T|w_1, w_2, \dots w_{T-1}) = \frac{e^{y_{w_T}}}{\sum_i e^{y_i}} \tag{2.9}$$

The parameters of the model are C, H, d, U, which are estimated by minimizing the log-likelihood:

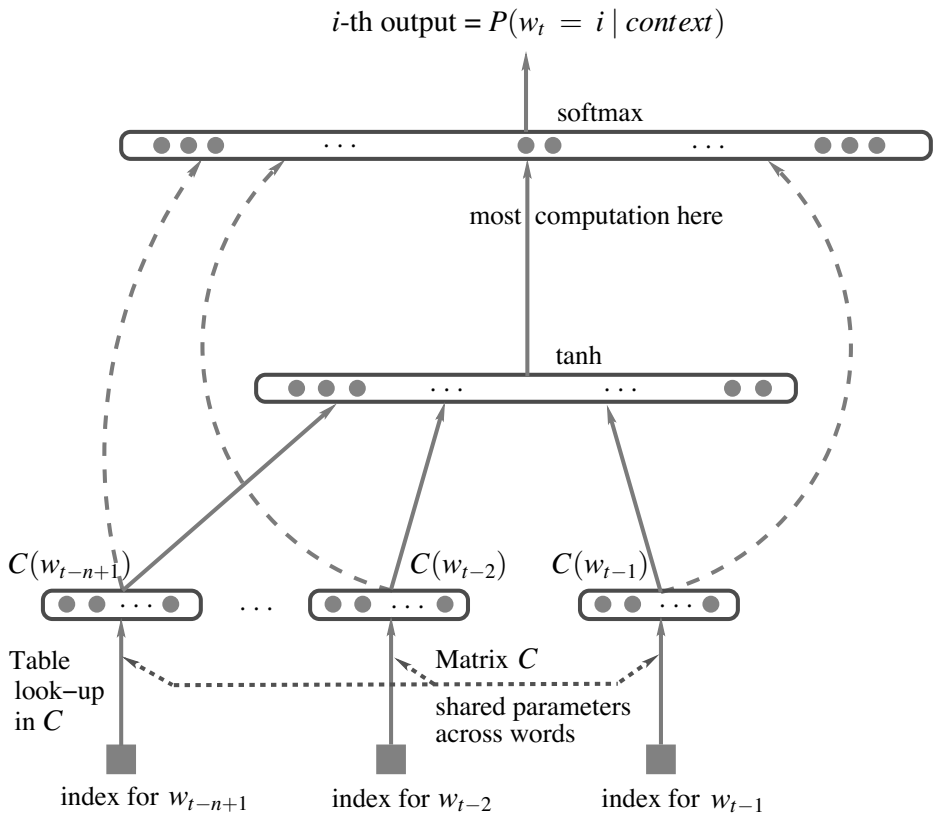
$$L = \frac{1}{T} \sum_i log(\hat{P}(w_T|w_1, w_2, \dots w_{T-1})) \tag{2.10}$$

More schematically model is presented in Figure 2.1 [6].

More elaborate models based on these ideas can be found in [17], [34], [32], [31] and [3]. It is interesting to note that distributed representation of words models are able to produce very rich vector spaces of features. For example, a skip-gram model from [32] estimates the vector space where

[6] Figure by Bengio, Yoshua and Ducharme, Réjean and Vincent, Pascal and Jauvin, Christian from [8].

Figure 2.1: Language model described in [8]



vec('Madrid') - vec('Spain') + vec('France') $\approx$ *vec('Paris').*

It is not a single example, and therefore the structure of such learnt representations is rich.

Similar ideas were extended to models where a token is a whole sentence. A good example of such model is skip-thought vectors model presented in [24].

Distributed representation for words became an important building block in many natural language processing models as a rich feature representation of words. Even though distributed representation for words are estimated for a very specific model, i.e. language model, it turns out that these features are

useful in seemingly unrelated problems like machine translation, sentiment classification etc.

In practice these features are used not as the direct word features but rather as a 'warm start' of the embedding matrix. In order to obtain such a 'warm start' the initial step is to estimate the language model similar to what was described in this section using textual information related to the problem at hand. Very often researchers use word features from pre-trained language model on one billion words provided by Google [21].

2.1.2 RNN/LSTM

Once we have a good feature representation of words the next logical step is to find a good representation of sentences. It can be seen from 2.7 that the word features were combined into a single feature via a linear map. However, the intrinsic structure of the natural language text lies in the time domain. The basic time series model that is able to learn efficient feature representations of sequence of tokens is the Recurrent Neural Network (RNN). The model consists of the following parts:

- Observed states (distribution vector representation of words): $\{x_t\}_{t=1}^{T}$
- Hidden state: $h_t = f(h_{t-1}, x_t)$ with $h_0 = 0$
- Score of the next state: $\hat{y}_t = W^T h_t$
- $\hat{P}(x_{t+1}|x_t, \dots x_1) = softmax(\hat{y})$ or equivalently $\hat{P}(x_{t+1}|h_t) = softmax(\hat{y})$

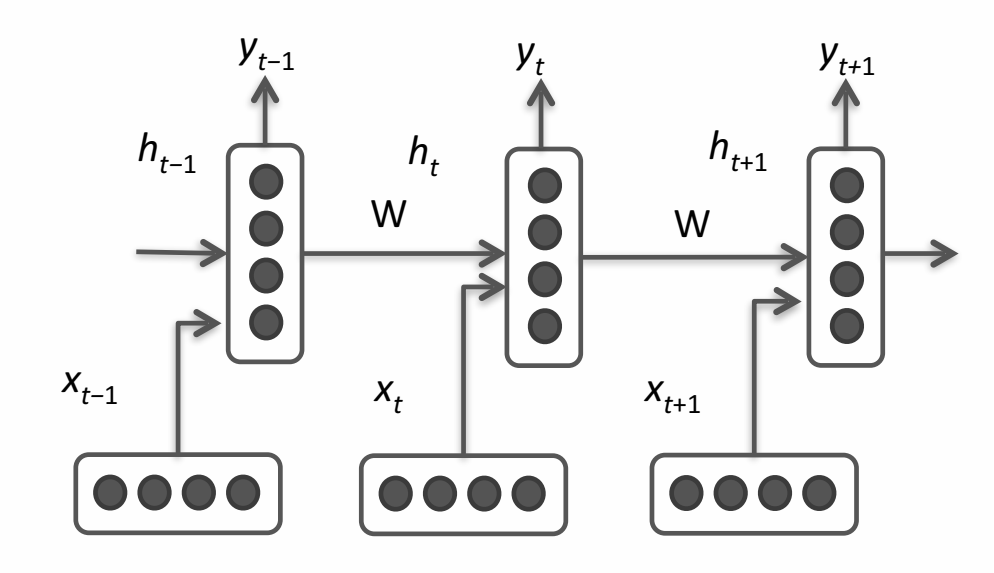
Thus hidden feature representation h_t encodes $x_1, \dots x_t$, i.e. more formally $h_t = f(x_1, x_2, \dots x_t)$. Estimation of such models is performed by minimizing

the log-likelihood:

$$L = \frac{1}{T} \sum_{i} log(\hat{P}(w_T|w_1, w_2, \dots w_{T-1})) \tag{2.11}$$

where $\hat{P}(w_T|w_1, w_2, \dots w_{T-1}) = softmax(\hat{y})$.

Figure 2.2: Recurrent Neural Network



A very important extension of RNN is the bidirectional RNN [37]. Bidirectional RNN consists of two RNNs:

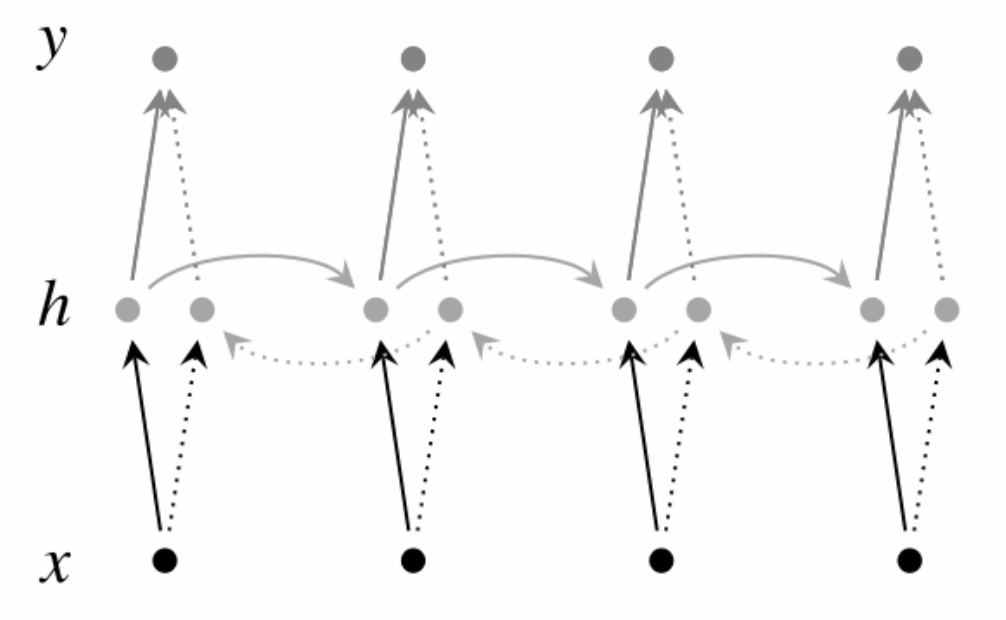
- regular RNN described above
- second RNN that encodes the same sequence but in reversed order. $\{x_t\}_{t=T}^{1}$

Now at each t we obtain two features:

- $\overrightarrow{h}_t$ feature that encodes the sequence $\{x_{t'}\}_{t'=1}^{t}$
- $\overleftarrow{h}_t$ feature that encodes the sequence $\{x_{t'}\}_{t'=t}^{T}$.

Thus we obtain a concatenated feature $[\overrightarrow{h}_t, \overleftarrow{h}_t]$ that encodes the past and the future at each step t, which allows for a better representation of sequences. One of the most significant applications of bidirectional RNNs is the neural machine translation model described in [5].

Figure 2.3: Bidirectional Recurrent Neural Network



Corresponding figures 2.2 and 2.3 [7] illustrate the structure of both models.

RNN suffers from the vanishing/exploding gradient problem which will be discussed in Section 2.1.3. Long Short Term Memory model (LSTM) [16], [13] introduces the 'forgetting' mechanism which allows for moving weight between h_t and x_t.

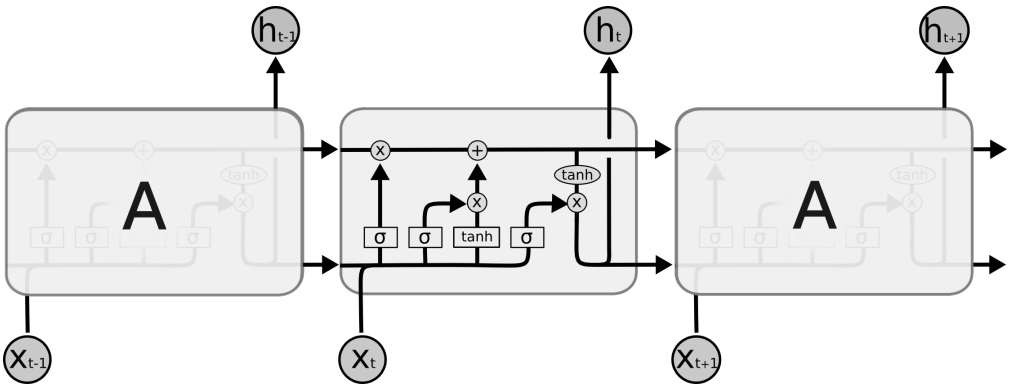
[7] By Christopher Manning and Richard Socher from http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture8.pdf

Definition 2.1.6. LSTM. Given a sequence $\{x_t\}$ LSTM is defined recursively by the following set of equations:

$$\begin{aligned} i_t &= \delta(W^i x_t + U^i h_{t-1}) \\ f_t &= \delta(W^f x_t + U^f h_{t-1}) \\ o_t &= \delta(W^o x_t + U^o h_{t-1}) \\ \tilde{c}_t &= \delta(W^c x_t + U^c h_{t-1}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t). \end{aligned} \tag{2.12}$$

Figure 2.4 [8] illustrates the structure of the LSTM.

Figure 2.4: LSTM



[8] By Chris Ola: h[p://colah.github.io/posts/2015-08-Understanding-LSTMs/

2.1.3 Estimation technique/Backpropagation

The problem in minimizing the log-likelihood is that the function is non convex. In practice, the main algorithm used in minimizing log-likelihood for neural network based NLP models is gradient-descent. The problem in applying gradient descent to neural networks is in the complexity of the functions that represent those models. These functions are basically multi-layered compositions of mappings. Computing derivatives and updating the estimated parameters is a challenging task. The efficient algorithm of computing derivatives was only discovered in 1980s, see for example [27].

It turned out that RNN models pose another estimation problem. Since RNNs are recursive functions, the derivative is the product of the same Jacobian, i.e. Jacobian raised to a power of 30-50. In practice, once the Jacobian becomes small the product converges to zero, or if Jacobian becomes large it diverges to infinity. This is the so-called vanishing/exploding gradient problem discussed in [9] and [14]. The same problem was studied in the framework of dynamical system in [15]. The dynamical system in this case is the gradient descent update mapping:

$$w_{t+1} = w_t - \alpha \nabla f(w_t) \tag{2.13}$$

where f is the objective function and α is the learning rate.

Neural networks are quite sensitive to the learning rate parameter. In practice researchers use adaptive learning rates. One of the most popular algorithms is RMSProp (Root Mean Square Propagation) [38]. The update equations are

$$v_{t+1} = \gamma v_t + (1-\gamma)||\nabla f(w_t)||^2 \tag{2.14}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_{t+1}}}\nabla f(w_t) \tag{2.15}$$

where γ and η are hyper parameters. Another popular algorithm is Adam (Adaptive Moment Estimation) presented in [23].

Another estimation technique widely used in practice is stochastic mini batch gradient descent [28]. The idea is to perform gradient descent update with small random sample of data. The rationale behind it is two-fold:

- First, it is less computationally intensive to perform a gradient descent using only a small batch of data. It is also a good solution for the memory bound problem.
- Second, stochastic update helps to perturb the gradient descent path, which is important in non convex problems.

Non convexity of neural networks comes in the form where we can observe a lot of local minima. However, it was conjectured in [6] in 1989 and recently proved in [22] in 2016. nonexistence of poor local minima. Therefore, the objective in the loss minimization problem is not the global solution, but rather a 'good' local solution, which greatly simplifies the estimation problem.

2.2 Approach/Models implemented

In this project I implemented the following 4 models:

- **Conditional bidirectional RNN encoder** described in [4]. The model

is presented in 2.7. First, bidirectional RNN encodes the name (i.e. 'Donald Trump'). Second bidirectional RNN encodes the tweet (i.e. donald trump has my vote! #donaldtrumpforpresident #donaldtrump) with the initial hidden state produced by the first bidirectional RNN encoder (conditioning). Then the final hidden state is classified as

NONE/FAVOR/AGAINST.

The source code for the model is available in tensorflow 0.6 [2] (the link is in the paper). I ended up rewriting it in tensorflow 1.0.

There is an important part of the model not documented in [4]. By inspecting the code, I discovered that the estimation of the model was done using 3 classes. But during inference authors used simple hand crafted features to classify name-tweet pairs as NONE/RELATED and then used the estimated model to classify RELATED pairs as FAVOR/AGAINST. Therefore, the model was estimated for 3 classes but inference was done only for 2 classes, which does not seem to be theoretically sound. Addition of a third class during estimation brings more complexity to the data and thus affects the properties of the estimator.

This turned out to be a very crucial part of the model. I compared the accuracy on the test set of the model with and without hand crafted feature classifier. The difference in accuracy [9] is significant: 43 % vs 63 % respectively. What could be even more surprising is that the hand crafted features are very simple. For example, given a person's name: 'Donald Trump' and a tweet the problem is to classify the tweet as

[9]Percentage of correctly predicted data points in the test set.

RELATED or NONE [10] (unrelated) with respect to the person's name. The hand crafted feature then: if a tweet contains at least one of the strings from the list {'donald trump', 'trump', 'donald'}, it is considered to be RELATED to 'Donald Trump' and NONE otherwise. I make use of this fact in the experimental design and discuss this issue in the Section 3.2.3.

- **Conditional RNN encoder with neural attention** described in [35]. Its schematic representations are in figures 2.5 and 2.8. First RNN encodes the name ('Donald Trump'). Second RNN encodes the tweet with the initial hidden state produced by the first RNN encoder (conditioning). Let $\{h_j^n\}_{j \in J}$ and $\{h_k^w\}_{k \in K}$ be the encodings of name and tweet respectively. The attention vector is then

$$a_k = \sum_{j=1}^{M} \alpha_{kj} h_j^n, \tag{2.16}$$

where attention weight α_{kj} models the alignment of h_J^n and h_k^w. That is

$$\alpha_{kj} = \frac{\exp(e_{kj})}{\sum_{j'} \exp(e_{kj'})} \tag{2.17}$$

with

$$e_{kj} = W^e \tanh\left(W^n h_j^n + W^w h_k^w + W^a h_{k-1}^a\right). \tag{2.18}$$

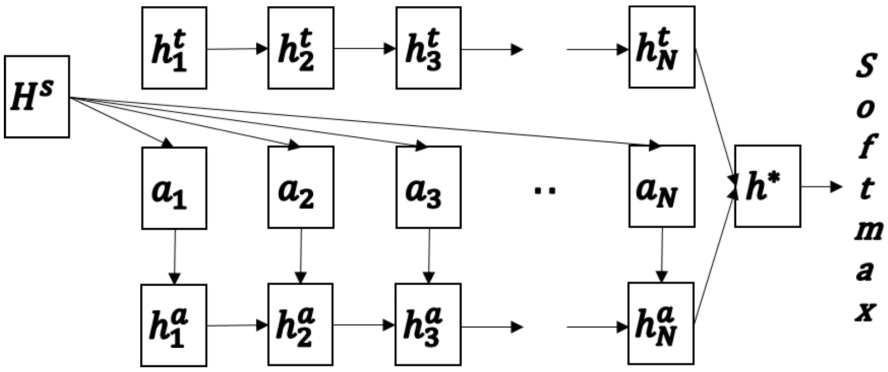
[10] Here I use the same terminology as in [4]. We are detecting stance. Hence, the stance can be related or none (no stance).

Final encoder summarizes attention vectors:

$$h_k^a = a_k + \tanh\left(V^a h_{k-1}^a\right). \tag{2.19}$$

The last layer feature h_N^a is used for classification. I implemented the model in tensorflow 1.0 following the presentation in the paper [39].

Figure 2.5: The figure taken from [39] clarifies the wiring of Conditional RNN encoder with neural attention



- **Matching LSTM (mLSTM) encoder with neural attention** described in [39]. Its schematic representations are in figures 2.6 and 2.9. First RNN encodes the name ('Donald Trump'). Second RNN encodes the tweet without conditioning on the hidden state produced by the first RNN encoder (conditioning). Let h_j^n and h_k^w be the encodings of name and tweet respectively. The attention vector is then

$$a_k = \sum_{j=1}^{M} \alpha_{kj} h_j^n, \tag{2.20}$$

where attention weight α_{kj} models the alignment of h_j^n and h_k^w. That is

$$\alpha_{kj} = \frac{\exp(e_{kj})}{\sum_{j'} \exp(e_{kj'})} \tag{2.21}$$

with

$$e_{kj} = W^e \tanh\left(W^n h_j^n + W^w h_k^w + W^m h_{k-1}^m\right). \tag{2.22}$$

The final encoder summarizes attention vectors using mLSTM (slight modification of LSTM).

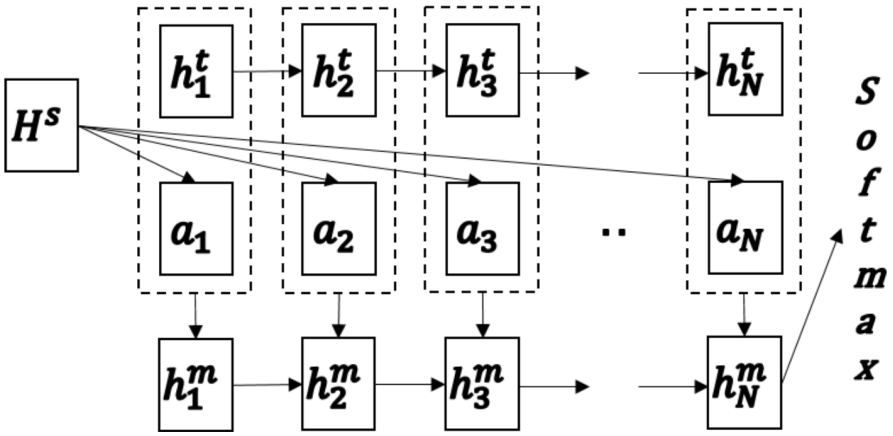
Definition 2.2.1. Matching LSTM. Let $m_k = [a_k, h_k^w]$ then

$$\begin{aligned}
i_k^m &= \delta(W^{mi} m_k + V^{mi} h_{k-1}^m + b^{mi}) \\
f_k^m &= \delta(W^{mf} m_k + V^{mf} h_{k-1}^m + b^{mf}) \\
o_k^m &= \delta(W^{mo} m_k + V^{mo} h_{k-1}^m + b^{mo}) \\
c_k^m &= f_k^m \odot c_{k-1}^m + i_k^m \odot \tanh\left(W^{mc} m_k + V^{mc} h_{k-1}^m + b^{mc}\right) \\
h_k^m &= o_k^m \odot \tanh\left(c_k^m\right)
\end{aligned} \tag{2.23}$$

The last layer feature h_N^m is used for classification. I implemented the model in tensorflow 1.0 following the presentation in the paper [39].

- **2 stage classification** proposed by me. First stage classifier treats the problem as if it has only 2 categories as opposed to 3 as in the original formulation. First stage model classifies name-tweet pairs as 'NONE/RELATED'. The pairs which were marked as 'RELATED' are then passed to the second stage model which classifies name-tweet pairs as 'FAVOR/AGAINST'. In this project I used **Conditional bidirec-**

Figure 2.6: The figure taken from [39] clarifies the wiring of mLSTM encoder with neural attention



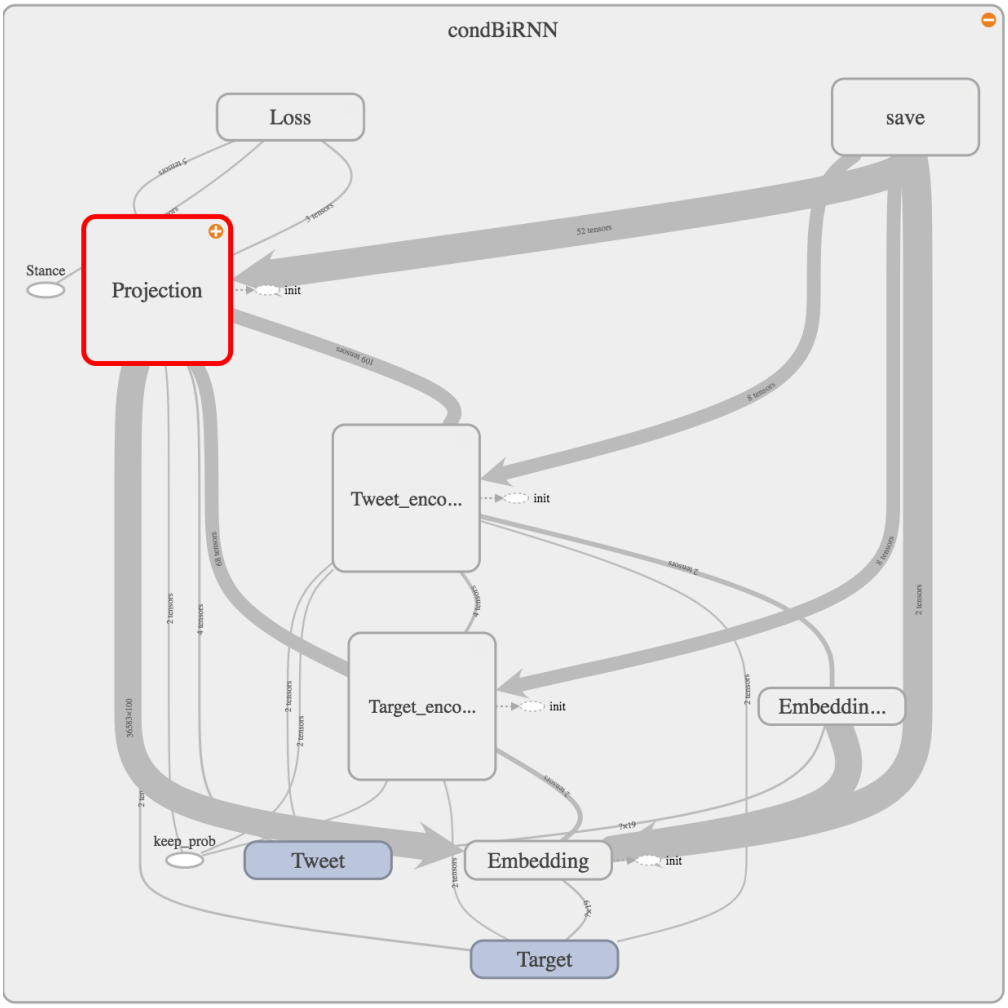
tional RNN encoder as a first and second stage model. In another approach I used very basic hand crafted feature from [4] (decribed in **Conditional bidirectional RNN encoder** subsection) for the first stage model. I implemented the model in tensorflow 1.0.

2.3 Rationale

In this project, I implemented and applied two models from a different subfield of NLP, Recognizing textual entailment (RTE). It seems to me that SD and RTE are very closely related and therefore it serves as a rationale for choosing these models. Furthermore, recent models in RTE incorporate an attention mechanism, which is not present in approaches from SD.

- **Conditional bidirectional RNN encoder** The model was picked since it achieved state of the art performance in the stance detection

Figure 2.7: Conditional bidirectional RNN encoder.



subtask of the SemEval 2016 [33] data set. However, the model that was used seemed a little simplistic. In particular, authors only used a bidirectional RNN encoder. One apparent improvement is the attention mechanism. On the other hand, the performance (around 43%) of the model made me think that there is some room for improvement that might lead to interesting general conclusions. In this project I consider this model as a baseline model.

- **Conditional RNN encoder with neural attention** This is the first model from RTE. As with the previous model, it also uses a conditional RNN encoder, but the main distinction is the addition of attention mechanism.

- **mLSTM encoder with neural attention** This model was picked due to its superior performance on RTE problems compared to **Conditional RNN encoder with neural attention**. It also has a curious modification of LSTM.

- **2 stage classification** The main rationale for implementing this model was the hypothesis that the classes are not equally well separated. I will discuss the hypothesis in detail in the Section 3.2.1.

Figure 2.8: Conditional RNN encoder with neural attention

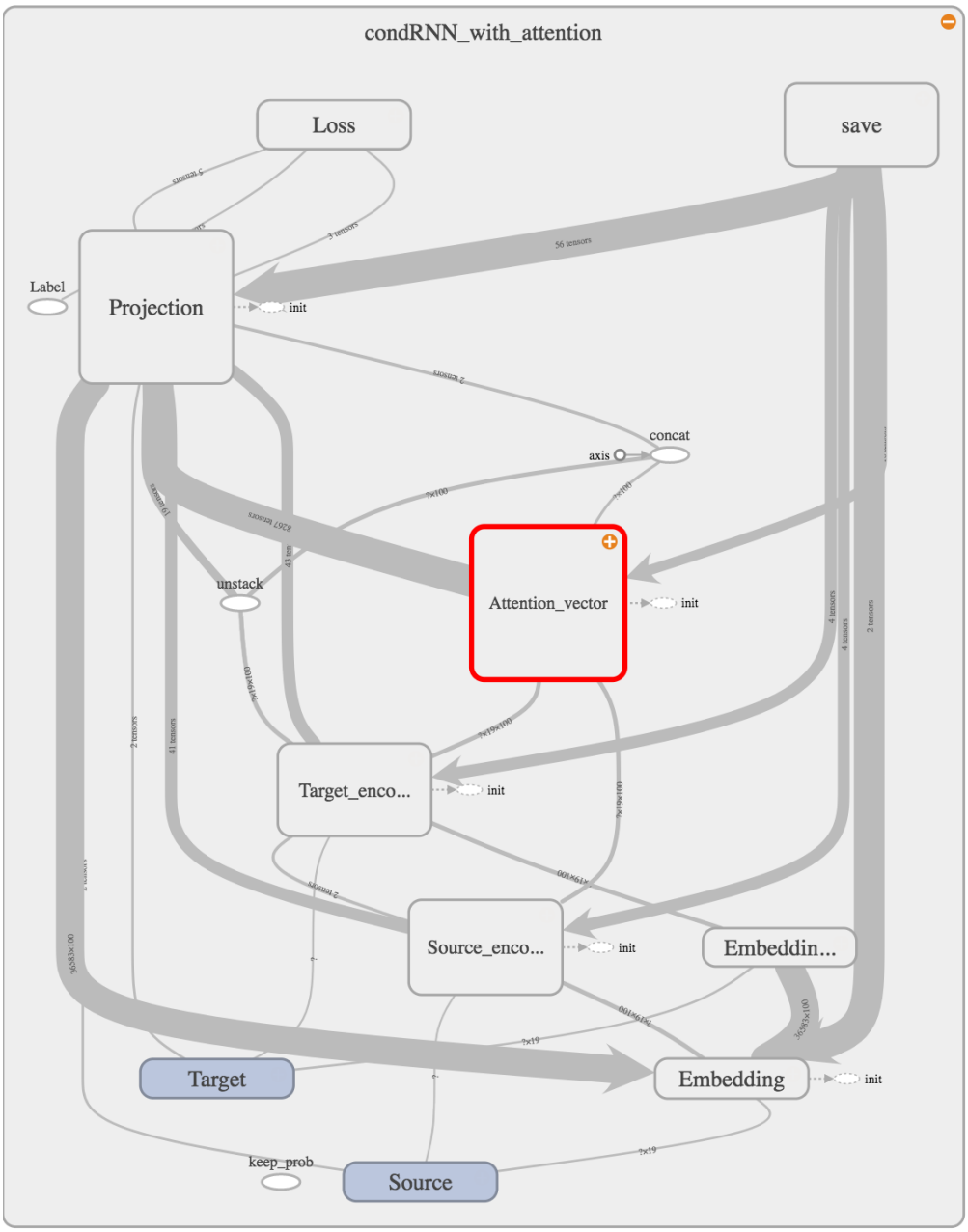
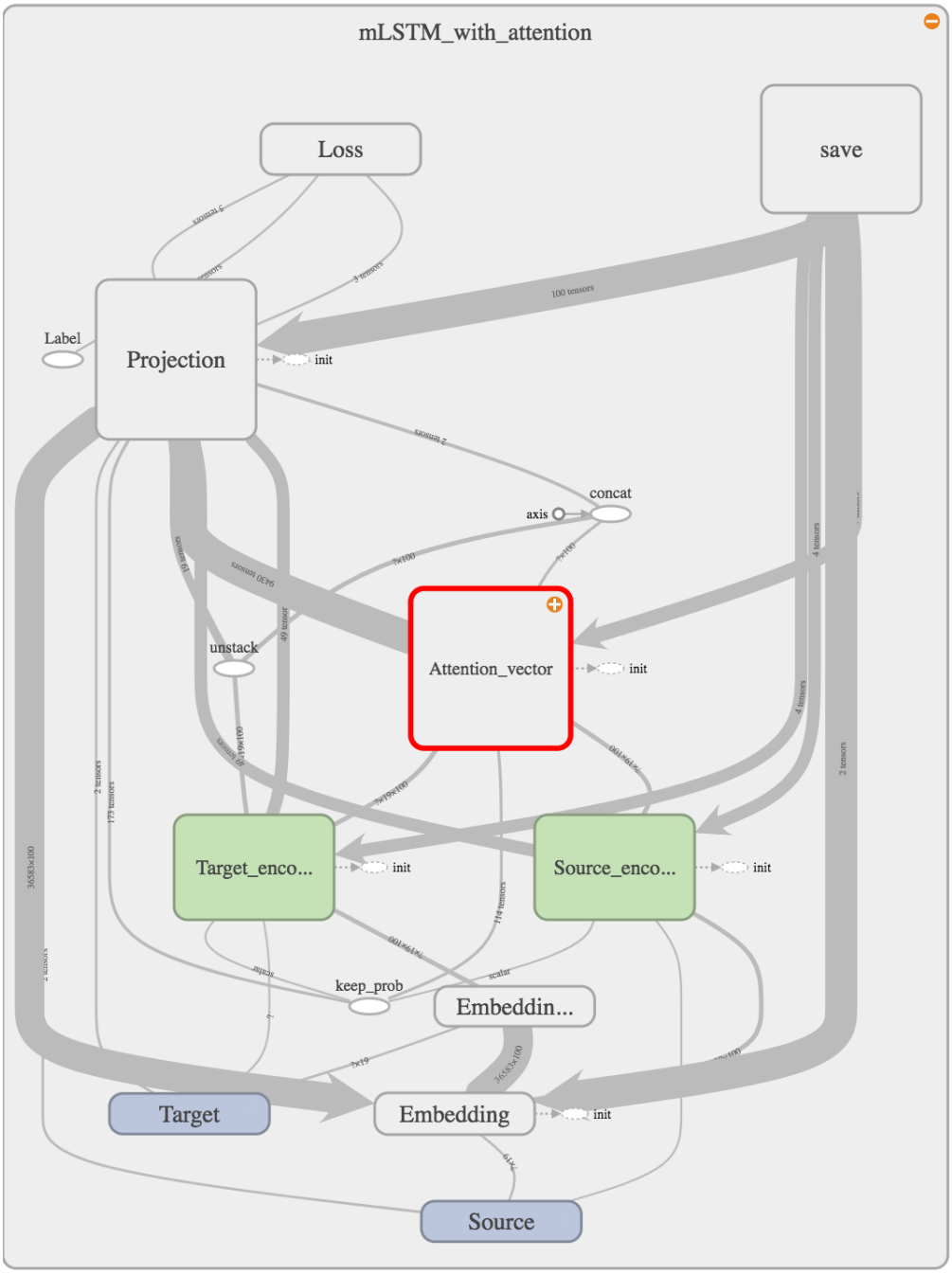


Figure 2.9: Matching LSTM (mLSTM) encoder with neural attention



Chapter 3

Numerical Studies

3.1 Problem

In recent years, fake news has become a more serious problem. This is mainly due to the popularity of social networks, search engines and news aggregators that propagate fake news. Classifying news as fake is a hard problem as was stated in Section 1.1. However, in [30] authors show that it is possible to distinguish between fake and real news by considering how many related tweets agree/disagree with the news. Therefore, in the simplest case the problem can be reduced to identifying whether a given tweet agrees with, disagrees with or is unrelated to with the news in question. In general this problem is referred to as 'stance detection' (SD). In machine learning terminology this is a classification problem.

In this project I addressed a simplified problem. Instead of a piece of news I consider a name. So, given:

- a name: Donald Trump

- and a tweet: donald trump has my vote! #donaldtrumpforpresident #donaldtrump

the problem is to identify the stance: NONE/FAVOR/AGAINST. The dataset used in this project is from stance detection subtask B part of the well know SemEval 2016 [33] data set.

The reason for the simplification is twofold:

- to my knowledge at the time of writing there was no commonly accepted data set for fake news
- simplified problem allows for reduced cost of conducting experiments

The reasons for choosing this particular problem:

- As far as I can judge SD is an 'unsolved' problem. My conclusion is based on the accuracy achieved by state of the art models in this field (around 50%). Therefore, there is room for interesting research which could lead to general results.
- The 'size' of the problem seemed manageable.

3.2 Experiments

3.2.1 Hypotheses

For the purpose of the current project I formulated and tested the following hypotheses:

1. Addition of attention mechanism to RNN encoder allows for a better stance detection.

2. mLSTM in conjunction with attention mechanism allows for a better stance detection.

3. Classes are not equally well separable at the same time. Intuitively, the 3 category problem NONE/FAVOR/AGAINST is in fact two unrelated 2 category problems: NONE/RELATED and FAVOR/AGAINST. And one model might struggle in separating all three classes at the same time. Therefore, the hypothesis states that in order to reduce the complexity of the problem we can separate NONE/RELATED and then refine separation for FAVOR/AGAINST by using the same model.

4. Classes are not equally well separable in 2 stages. Geometrically, the hypothesis states that sub-manifolds of 3 categories are entangled in different ways. That is, the decision boundary between NONE/RELATED is different from the decision boundary between FAVOR/AGAINST. Probabilistically, the problem itself has a specific structure, which allows for the following factorizations:

$$P(FAVOR) = P(FAVOR \mid RELATED)\ P(RELATED),$$
$$P(AGAINST) = P(AGAINST \mid RELATED)\ P(RELATED).$$

Stage 1 models

$$P(RELATED)$$

and stage 2 models

$$P(FAVOR \mid RELATED),\ P(AGAINST \mid RELATED).$$

Here we exploit the fact that only if the name and the tweet are related then we can say what is the type of relationship. Therefore, hypothesis states that two different models are needed in order to separate NONE/RELATED and FAVOR/AGAINST.

5. **Conditional bidirectional RNN encoder** model is not robust to addition of a 'hard to separate' class. As I mentioned in the Section 2.2 the model from [33] was trained on a data set with 3 classes (NONE/FAVOR/AGAINST) and tested on only with 2 classes (FAVOR/AGAINST). The third class was labeled by hand crafted features. Therefore, the idea behind the experiment to see how the model performs if it is trained on a data set with 2 classes (FAVOR/AGAINST) and used to label the test set containing the same 2 classes vs training on 3 classes and testing on 3 classes.

3.2.2 Experimental design

As it was briefly mentioned previously the data set used in all experiments is a stance detection subtask B from SemEval 2016. It should be noted that the training set contains only one name, namely 'Donald Trump' and, more im-

portantly, the training set is not labeled. However, authors of [4] autolabelled it by using hand crafted features (simple regular expressions) which are described in the paper and are publicly available (link is in the paper). Potential implications of this fact will be discussed in the Section 3.2.3.

The data set is split into training (18,760 name-tweet-label triples) and testing sets (707 name-tweet-label triples). It should be noted that the training set is unbalanced with respect to 3 classes. The training set consists of 50% of NONE class, 25% of FAVOR class and 25 % of AGAINST class. For the 3 category classification, I'd rather subsample the NONE subset in order to make the data set balanced. However, in order to keep the results comparable to other published models that make use of the official data set provided by SemEval 2016, that was not altered. Potential implications of that will be also discussed in Section 3.2.3.

1. Addition of attention mechanism to RNN encoder allows for a better stance detection. The experiment design in this case is quite straightforward. I compared the performance of the baseline model **Conditional bidirectional RNN encoder** and a similar model but with attention mechanism **Conditional RNN encoder with neural attention**. The 'training' dataset (18,760 name-tweet-label triples) specified by SemEval 2016 was split into training (90%) and evaluation sets (10%). The models were trained with the same hidden state dimensions and dropout rate for 8 epochs. All these parameters were set as in the original paper [4] describing the baseline model. I used cross entropy loss on the evaluation set as a performance measure.

2. mLSTM in conjunction with attention mechanism allows for a better stance detection. The experiment design in this case is quite straightforward. I compared the performance of the baseline model **Conditional bidirectional RNN encoder** and a similar model but with attention mechanism **mLSTM encoder with neural attention** The 'training' dataset (18,760 name-tweet-label triples) specified by SemEval 2016 was split into training (90%) and evaluation sets (10%). The models were trained with the same hidden state dimensions and dropout rate for 8 epochs. All these parameters were set as in the original paper [4] describing the baseline model. Cross entropy loss on the evaluation set was chosen as a performance measure.

3. Classes are not equally well separable at the same time. The experimental design in this case is slightly different than in two previous experiments. I tested the hypothesis by comparing the performance of the baseline model **Conditional bidirectional RNN encoder** (without hand crafted features) to the 2 stage model, where at each stage I applied the same model **Conditional bidirectional RNN encoder**. The idea behind the experimental design was to see if there is a difference between performing one 3 category classification or two 2 category classification in a sequence.

   The 2-stage classification required splitting the training set into training_stage_1, training_stage_2 and evaluation set into evaluation_stage_1, evaluation_stage_2 sets. Stage 1 model classifies name-tweet pairs into NONE/RELATED and, hence, training_stage_1 set contains the same

sample as the training set but with both FAVOR and AGAINST data points labelled as RELATED. evaluation_stage_1 set is constructed in a similar fashion. Stage 2 model classifies name-tweet pairs into FAVOR/AGAINST and, therefore, training_stage_2 set only includes the name-tweet-label triplets from the training set with labels FAVOR/AGAINST. evaluation_stage_2 set is constructed in a similar fashion. Stage_1 and stage_2 use the same model but trained on different sets: training_stage_1 and training_stage_2 sets. That is same model but with different parameterizations.

In this experiment it is difficult to compare evaluation training loss curves, since 2-stage classification produces two curves: one for stage_1 and one for stage_2 (although they are still very instructive). I used training/evaluation sets to training models and monitor overfitting. Then I used test set to compare accuracy of predictions by both models. Ideally, the data need more splits in order to use test set once in the end. Since all the code was set up for training/evaluation/test split, I ended up using test set twice (in this and next experiments). I will elaborate on that issue in the Section 3.2.3.

4. Classes are not equally well separable in 2 stages. In this experiment, I compare the performance of the 2-stage classifier where both stages use the same model (homogenous) and 2-stage classifier where stage_1 and stage_2 models are different (heterogenous).

   Homogenous model uses the baseline model **Conditional bidirectional RNN encoder** as described in Section 2.2 for both stages. In the het-

erogenous setup I used simple hand crafted features discussed in Section 2.2 as stage_1 model that classifies tweets into RELATED and NONE. For stage_2 model I used the baseline model **Conditional bidirectional RNN encoder** that classifies RLATED tweets into FVOR and AGAINST.

5. **Conditional bidirectional RNN encoder** model is not robust to addition of a 'hard-to-separate' class. In this case the experimental design is quite straightforward: train the baseline model **Conditional bidirectional RNN encoder** on the data set with 3 classes (NONE/FAVOR/AGAINST) and test on data with same 2 classes using hand crafted features to classify NONE/RELATED (original approach from [4]). I replicate the same approach with one modification: the model will be trained on 2 classes only rather than 3.

3.2.3 Results

Here I list the results of experiments for each hypothesis from Section 3.2.1 in the same order:

1. Addition of attention mechanism to RNN encoder allows for a better stance detection.

2. From figures 3.1 and 3.2 it can be seen that mLSTM model with attention is much worse than the baseline model. There is no significant difference between the baseline model and the Conditional RNN encoder with neural attention. It should be noted that loss curves represent the

dynamics of gradient descent. After every epoch the loss is computed using training set and evaluation set. The dynamics of gradient descent shows the convergence properties of each model and potential signs of overfitting. Both training loss curves and evaluation loss curves have monotonically decreasing shapes. That implies that models do not have overfitting problems.

3. Classes are not equally well separable at the same time. By comparing results from numerical experiments for 'baseline without hand crafted features' and '2-stage baseline' it can be concluded that the hypothesis is false (the difference of means is almost within the smallest standard deviation). Therefore, it does not matter if we use the same model in 2 stages with fewer classes or same model with all classes at once.

4. Classes are not equally well separable in 2 stages. By comparing results from numerical experiments for '2-stage baseline' and '2-stage features + baseline' it can be concluded that the hypothesis is true and the difference is very significant (the difference of means is much bigger than the biggest standard deviation). That implies that the decision boundaries between NONE/RELATED and FAVOR/AGAINST are intrinsically different. This conclusion is also supported by results presented in table 2. Here we can observe that baseline model separates FAVOR/AGAINST classes much better than NONE/RELATED. Also simple hand crafted feature classifier separates NONE/RELATED much better than the deep neural NLP (baseline) model.

5. **Conditional bidirectional RNN encoder** model is not robust to ad-

dition of a 'hard to separate' class. By comparing results from numerical experiments for 'baseline with features' and '2-stage features + baseline' it can be concluded that the hypothesis is true although the difference is not that significant (the difference of means is slightly bigger than the biggest standard deviation).

Figure 3.1: Training loss curves

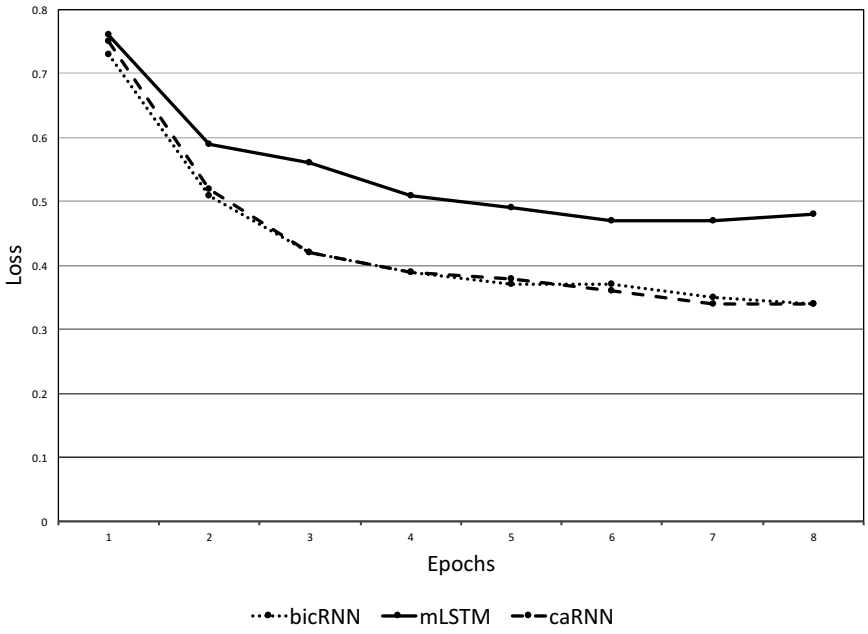


Table 3.1: Test accuracy for all 3 classes (5 repeats)

	mean	standard deviation
baseline without hand crafted features	0.4311	0.003
2-stage baseline	0.4306	0.002
baseline with features	0.6320	0.004
2-stage features + baseline	0.6424	0.007

Figure 3.2: Evaluation loss curves

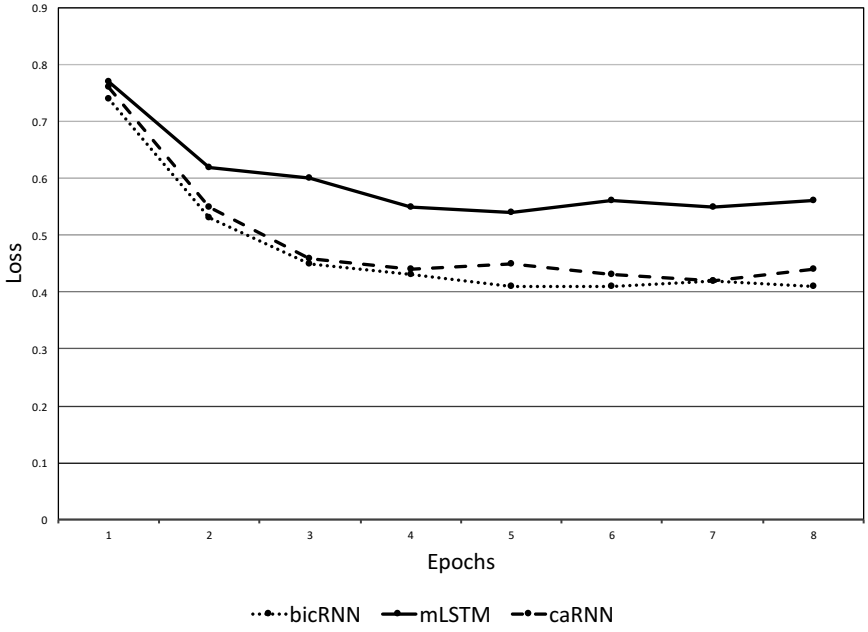


Table 3.2: Test accuracy for 2 classes (5 repeats)

	mean	standard deviation
features	0.7991	0
stage 1 baseline	0.4543	0.003
stage 2 baseline	0.6527	0.04

Chapter 4

Conclusion

4.1 Critical evaluation

Let me first discuss the most general issue, i.e. the issue with the data set. First of all the biggest issue with the training data set is that it is not labelled by humans and, therefore, geometrically sub-manifolds of classes are distorted. That might imply any NLP model trained on that data might not generalize well to the real data sets.

The training data is also unbalanced which can skew the modeled distributions. And a very strong model free baseline is always choose NONE and it will produce 50 % accuracy on training and evaluation sets. At the same time the stage_1 and stage_2 training/evaluation sets turned to be perfectly balanced.

The results of the experiments don't allow us to conclude that attention mechanism is superior to biconditional approach. Besides the data set problems, another issue is that the baseline model has biconditional RNN encoder

and **Conditional RNN encoder with neural attention** has only one directional RNN encoder. For a more fair comparison **Conditional RNN encoder with neural attention** should be implemented with the biconditional RNN encoder, which won't alter other parts of the model.

The hypothesis 'Classes are not equally well separable in 2 stages' might lead to a very general conclusion. The geometry of the 'true' decision boundaries between classes can be very different and thus in order to model those boundaries we might need different models. 2-stage model can be generalized in a straightforward way to n-stage model where at each stage we separate one class from the rest. In this project I hypothesized that separation between NONE/RELATED and FAVOR/AGAINST were different. Similarly, separating CATS/DOGS is different from separating PLANES/FROGS in image classification problems. In general, the assumption of classes being separated by very similar boundaries seems rather strong.

Another point to make is that the baseline model with features from [4] was trained on the dataset with 3 classes and then applied on the data set with 2 classes. It is possible that 3rd 'unused' class distorts the probability distribution modeled by the neural net attracting some weight from other classes and thus seems to me very bad/wasteful/unclear from theoretical point of view. Empirical results support that conclusion. The model trained on 2 classes and test on 2 classes achieves a higher accuracy on the test set.

Overall, my main two contributions are

- Designed and implemented 2 stage classifier.
- Formulated and tested hypotheses in which more advanced natural lan-

guage models are applied to the problem described in 3.1.

4.2 Lessons learned and future work

A very good lesson learnt during this research project is to be a critical thinker. Research papers should be taken with a grain of salt. One of the crucial points in research is reproducibility of results. It was interesting to discover a very crucial hand crafted features based part of the model used but not documented in [4].

From Section 2.2 it can be seen that research should be theoretically sound. This is especially important in todays widely spread applications of statistical machine learning methods.

Another important point is that neural networks is not a one-size-fits-all model. We can see from the experiments that a simple hand crafted feature almost twice more effective than complex neural networks.

Neural nets separate FAVOR/AGAINST more successfully than NONE/ RELATED, it would be interesting to see the performance of the models with attention in classifying FAVOR/AGAINST. It seems that attention mechanism in conjunction with bidirectional RNN might improve the accuracy.

Choosing the right data set is very important. In the future work it is best to choose dataset labeled by humans, balanced and with sufficient sample size.

Bibliography

[1] Twitter study. (technical report). Pear Analytics, 2009.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR 2017 (Oral)*, 2016.

[4] Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*, 2016.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[7] Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J Mooney. Representing meaning with a combination of logical form and vectors. *arXiv preprint arXiv:1505.06816*, 2015.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[10] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[11] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer, 2006.

[12] William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, 2016.

[13] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[14] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[15] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

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

[17] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[18] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[19] Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, 2014.

[20] Sergio Jimenez, Fabio Gonzalez, and Alexander Gelbukh. Text comparison using soft cardinality. In *String Processing and Information Retrieval*, pages 297–302. Springer, 2010.

[21] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[22] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[25] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[26] Alice Lai and Julia Hockenmaier. Illinois-lh: A denotational and distributional approach to semantics. *Proc. SemEval*, 2:5, 2014.

[27] Yann Le Cun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[28] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

[29] Xiaomo Liu, Quanzhi Li, Armineh Nourbakhsh, Rui Fang, Merine Thomas, Kajsa Anderson, Russ Kociuba, Mark Vedder, Steven Pomerville, Ramdev Wudali, et al. Reuters tracer: A large scale system of detecting & verifying real-time news events from twitter. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 207–216. ACM, 2016.

[30] Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. Twitter under crisis: Can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM, 2010.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[33] Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. *Proceedings of SemEval*, 16, 2016.

[34] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[35] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kociskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[36] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[37] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[38] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[39] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[40] Jiang Zhao, Tian Tian Zhu, and Man Lan. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *Proceedings of the SemEval*, pages 271–277, 2014.