Stance Detection using Recurrent Neural Networks

Gagan Khanijau
UC Irvine
gkhanija@uci.edu

Manikanta Harsha Nadimpalli
UC Irvine
mnadimpa@uci.edu

Anirudh Rachuri
UC Irvine
arachuri@uci.edu

Abstract

The idea of stance detection was inspired from the fake news challenge-1 which was organized last year. On the data of news headline and body text pairs our aim was to decide whether the texts are related or unrelated. If related, we further wanted to identify whether they can be classified as 'agree', 'disagree', or 'discuss'. In order to solve the problem of stance detection we have explored various neural and recurrent model like multi-layer perceptron model, convolutional neural network model and Bi-directional LSTM model and their variations with feature engineering using natural language processing techniques. In this paper we present a detailed description of our models and experiments tried with them.

1 Introduction

Team Name: The_Hidden_Layer

Fake news (defined as deliberate misinformation or hoax) emerged as a defining topic in recent times and it has been widely cited as a major contributing factor to the outcome of the 2016 US presidential election. The Fake News Challenge(FNC-1) was organised in response to this issue with the stated objective of using machine learning to robustly identify fake news.

FNC-1, in particular, is focussed on one particular aspect of identifying fake news viz., stance detection. Stance detection involves estimating the relative perception(or stance) of two pieces of text relative to a topic, claim or issue. The goal of FNC-1 is to first predict if an article related to a headline and then if related, to estimate the stance of the article relative to the headline(i.e agree, disagree or discuss).

The FNC-1 dataset is derived from the emergent stance classification dataset provided by Ferreira and Vlachos(1). The original has many to many mappings, FNC-1 created a split to avoid the leaky mappings across splits. There are 1683 articles and 49972 headlines. So, using the FNC dataset we try to determine the relation of the article to a headline and further predict its stance. The data contains headline, body and stance which can take these 4 values[unrelated, discuss, agree, disagree].

2 Related Work

We reviewed various work done by winning teams and also some other teams who took part in the FNC-1 Stance detection.

Approaches by winners:

1. Model based on an weighted average between gradient-boosted decision trees and a deep convolutional neural network. We agree that boosting with complex architectures can certainly exploit the potential of machine learning and perform well, but we wanted to deal it with more context based models. We aimed to explore basic text specific approaches with language processing techniques which can perform at par with this approach.(11)

2. Multi-layer perceptron with concatenation of feature vectors of headline, article and joint features as input.(12)

3. Single hidden layer network with TFs of headline, article and TF-IDF cosine similarity as input features. This was a basic and effective approach, but the authors have not explored the task of feature enhancement using NLP techniques. Along with the exploration of language features, we also extended this model to the two-step classification approach which we have listed later in section 3.1.(14)

Other approaches:

- In another work, the authors encode the heading and the body text using two different bidirectional recurrent neural networks and the final states of the RNN cells are encoded as features for a feed forward network classifier. Further an attention mechanism is used in which the output vector of each body word is compared to the headline in order to determine an attention weight for each word. These attention weights are then concatenated with the final states of headline RNN as features for the classifier.(10)

- One of the approaches we reviewed(8), not related to the FNC-1, worked on some different fake news detection. The authors tried to use convolutional neural networks with max pooling over word vectors using GloVe(9).

3 Approach

We have worked on the following three models:

3.1 Feed forward Neural Networks:

Feed forward Neural nets are one of the most prominent learning techniques which have been effective not only in text related problems, but also in various other machine learning tasks. It takes in a n-dimensional input vector which is fed to a hidden layer and sums up to produce an output. The neural networks provide a flexibility of encoding the words in various vector forms and adding any different number of features(using NLP techniques) to capture maximum information about the text. Thus, it is quite intuitive to use this model and experiment on this using various feature engineering approaches using language processing techniques.

The input in our case was the term frequency of most commonly used words across the corpus in the headline and body. We actually preferred to try more language related features rather than changing the model in this case. We used n-grams, stemming and similarity of dependency parse trees in order to incorporate various hand tuned features in this model to improve the performance of the model. These experiments have been described in detail in section 4.

3.2 Bidirectional RNN Encoder:

RNNs are a class of neural networks that works on sequences of input of variable length which is perfectly suitable for the current task at hand. They persist information as they pass along the sequence in the form of hidden states and pass this information along to the next set of inputs. Bidirectional RNNs have been used in sequence-to-sequence prediction problems such as machine translation by encoding a sentence in the source language by encoding a sentence in the source language in the hidden states of the RNN and decoding the encoding into the target language using a decoder(another RNN). Extending this approach for the current we encode both the headline and the body using Bidirectional RNNs and the final states of the RNN cells are concatenated as features for a separate classifier network. The

LSTM Long Short Term Memory networks and a special kind of RNNs capable of learning long-term dependencies. An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. In the context of natural language it captures context of words occurring previously and this context can then be used while predicting the current word under consideration.

GRU Gated recurrent Unit(GRU) combines the forget and input gates illustrated below into a single update gate[1]. Along with that, it also merges the cell state and hidden state and makes some other changes. The resulting model is thus much simpler and computationally cheaper and the results as similar to those obtained with LSTM

As part of this process we have used both the units to model the network

3.3 Convolutional Neural Network

A convolutional neural network or CNN is a class of deep, feed-forward neural networks. A CNN generally comprises of a number of convolutional and pooling layers optionally followed by fully connected layers. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. Pooling layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. CNNs were responsible for major breakthroughs in Image Classification and are the core of most Computer Vision systems today. The general notion is that CNNs are not a good fit for NLP tasks. But in truth, they perform quite well for NLP tasks. Typically, word embeddings are passed as input to CNNs. 1D convolution layer convolves on the embedding dimen-

sions and more importantly, CNNs are efficient in terms of representing *n*-grams. Pooling layer intuitively pools over *n*-grams of text and pooling size is analogous to *n*. Moreover, CNNs are very fast to train and can exploit GPUs. Hence, we felt CNN is a good option to explore for the task at our hand.

4 Experiments

4.1 Feed Forward Neural Networks - Bag of Words Model and Vectorization Techniques

We wanted to begin with the vanilla model of neural networks, with a single hidden layer network with the features as vectorization of words in headline and article body combined together, fed to a feed forward learning model. This reminded us to our first task in the course where we experimented with various word vectors on a logistic regression model where we tried in the experiments for count vectorization using ngram-model, Stemming(Snowball- NLTK), tf-idf weighting etc. We tried using the same input model to a neural network with single hidden layer with four outputs. The model was a word vector of headline and body text vector created by using most common n-grams(max range 2) stemmed by Snowball English NLTK Stemmer (15) which was fed to the a single hidden layer network with 100 neurons and output layer was the four classes. We tried varying the input size by controlling the number of features(or ngrams in this case). We also experimented with other different approaches of vectorization all the members tried previously but found this to be the best. A similar work was done by one of the winners who used the bag of words term frequency and cosine similarity as input vector. We also reproduced their results to contrast with our model.

Our model performed best on 6500 vector size each for body and headline and gave an FNC-1 score of 9580 which is better than all the winners. Also, the winner who used the similar model had a fnc score of 9521. The confusion matrix is shown in Table 1 on the right:

The variation of FNC score and accuracy using variation in the size of the input vocabulary after incorporating bi-grams into the word list is shown in Figure 2 and Figure 3 respectively. The model perfomed best when a vocabulary size of 6500 was considered.

Two step Classification: In order to cater the biasness of the data towards instances of pair belonging to unrelated class, we also tried to extend this model to two step classification task by just replicating the above model into two separate neural architectures. The first network just predicts whether the headline and pair are related. In case the pair is related, the same input was again fed on another model to predict the actual related class. The second network was just trained on the instances only when the samples were related to each other and belonged to one of the three

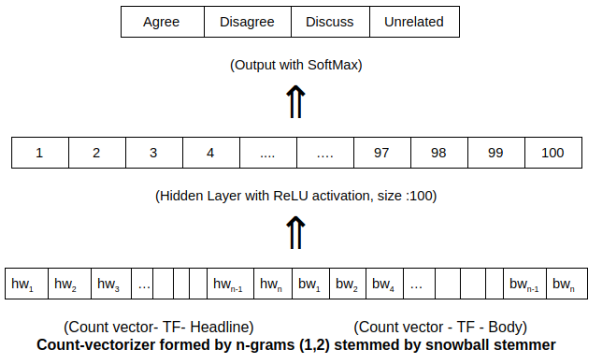

Figure 1: Feed forward model with word vectorization

$Gold \rightarrow$ / $Pred \downarrow$	**Agree**	**Disagree**	**Discuss**	**unrelated**
Agree	857	11	928	107
Disagree	158	50	373	116
Discuss	512	18	3685	249
Un-related	87	14	323	17925

Table 1: Confusion matrix for predictions of feed forward model.

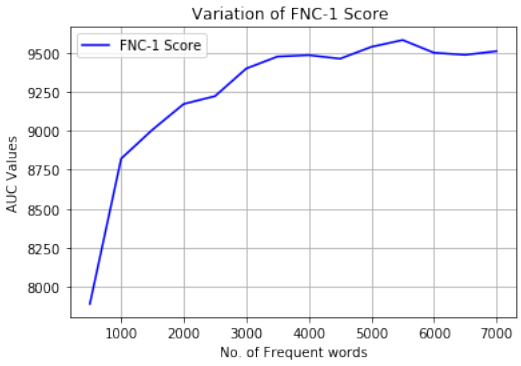

Figure 2: Variation of FNC-1 Score with n-gram vector

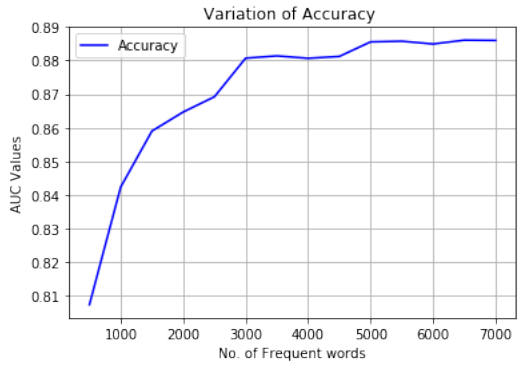

Figure 3: Variation of accuracy with n-gram vector

classes. This model gave an FNC score just about 9500. The possible reasons for the lack of significant improvement of this model might be because in the test step, the second network was only used if the first one predicted the related class which can result in few bad instances in case of false negatives(in case related gold sample is marked as unrelated).

Dependency Similarity Measure: Our basic intention in this model was not to complicate the architecture, instead we wanted to engineer more features which can give a better representation of the text. Since the model was performing good, we further went on to enhance this by adding more features. We felt the need for a similarity measure that captures the similarity much better than cosine similarity measure. We built a novel similarity measure for the pair of headline and body called "Dependency Similarity Measure" based on their dependency parsed trees. Intuitively, if you find the same pair of head and modifier in both the dependency trees, it is more probable that headline and body are related. Body text is generally long and the essence of the content is presented in the initial few sentences. So, we only considered the first five sentences of body to calculate the similarity score. We used Stanford Dependency Parser(13) for this. We increment the score by **0.7** for every head-modifier pair that is commonly found in both the trees and by **0.3** more if the nature of dependency is also same. The score is normalized by the number of words that are present in both headline and body. Going by this calculation, we got a score of *0.7371* for the same headline and body - *"This time around, they're moving even faster"*. Due to inefficiencies in our implementation and time constraints, we could evaluate this measure only on 2000 training and 2000 test data. FNC score increased from **631.5** to **639.5** after the addition of dependency similarity score as an additional feature for the input headline-body pair. Tuning the calculation of the score may result in much better performance.

4.2 Bi directional RNNs using LSTM

This process as explained before consists of using two separate Bi-directional recurrent neural networks to encode the headline and the body text and the final states of the RNN cells are concatenated as features for the classifier network.

The model is implemented as follows: The raw texts from the headline and the body are first tokenized using the nltk package and further embedded using 50-dimensional word vectors using GloVe 6B. The embedding parameters are marked non-trainable in our experiments since the given dataset in the task is not large enough. Out-of vocabulary text has been ignored but could be initialised with some random noise. Headline texts are padded with a size of 50 and the body texts are padded with a size of 200. Larger texts are truncated at the padding size.

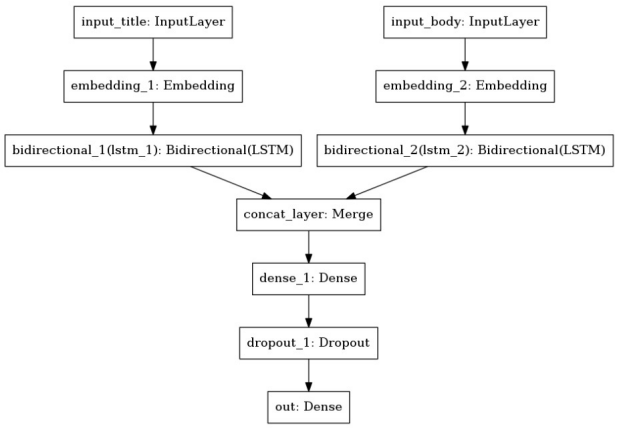

Figure 4: Bidirectional LSTM Architecture

For the bidirectional RNN described above an LSTM cell is used with a hidden layer consisting of 200 units. For the classifier, a 1-hidden layer forward NN with 100 dimensions on hidden layer is used. The final output layer is a 4-class softmax representing probability of 4 classes we are predicting here. A dropout with keep probability 0.9 is applied only on the hidden layer of classifier. The loss used is cross entropy, optimised using the Adam optimizer. The model was trained for 10

epochs with 32 batch size.

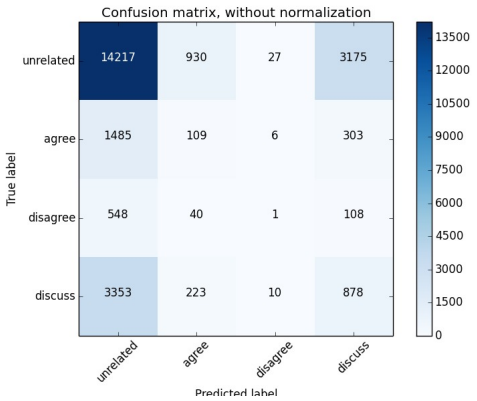

Figure 5: Confusion matrix for bi-directional LSTM

The FNC score obtained was 4714.75. The accuracy obtained was 59.83%.

4.3 Convolutional Neural Network

In general, convolutional neural networks (CNNs) are used for working on images. We tried exploring how these can be applied for text data set. Similar to Bidirectional LSTM Model, input words are converted into 50-dimensional vectors. We treated input word sequence of headline/article as temporal data and embeddings of words as spatial data. In our model, input headline and article are first passed separately to embedding layer. Then they are passed multiple layers of 1D convolution and 1D max pooling. We also sandwiched dropout layers with dropout probability = 0.5 in between these to avoid overfitting. 1D convolution layers is done on spatial data with kernel size as 5. 1D max pooling is done on temporal data with pooling size as 3 and this intuitively is equivalent to 3-gram model of text. After a series of convolutions and max poolings, headline and article are passed to a global max pooling layer to output single encoding for each headline and article. These encodings are concatenated and passed to dense layers before passing to an output 4-class softmax layer. Loss function and optimizer are same as the ones used in Bidirectional LSTM Model.

The FNC score obtained was 4558. The test accuracy obtained was 71.2%.

5 Conclusion & Prospective work

Currently, based on the performed experiments we have listed below the fnc score measure we have achieved with different techniques and the ones listed on the challenge website.

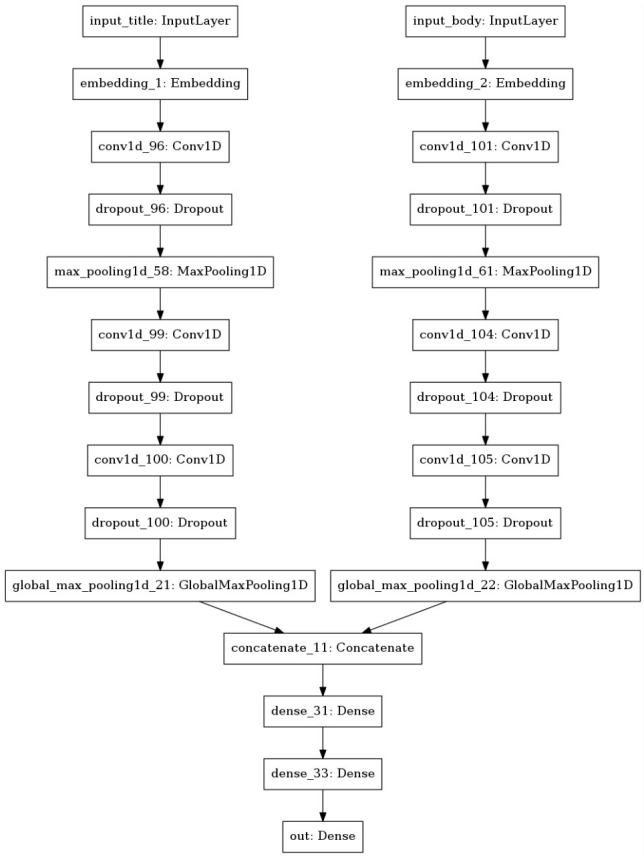

Figure 6: CNN Architecture

Models	**FNC-Score**
Fnc-baseline	8761
MLP with ngram + stemmer	**9580**
MLP with ngram + stemmer + stopwords	9487
Bi directional LSTM	4714
Convolutional Neural Network	4558
MLP: ngram + stem + 2-step	9490
Winner-1	9556

Table 3: FNC Scores with different models

As one can see, we did not use much of hand-tuned features while building our models. Some of such techniques include chargrams, polarity words, word overlap features etc. These features can be incorporated in our models to capture more information and gain better understanding of the input texts to yield much improved results.

We can use Named Entity Recognizer on headline and article. Consider this headline for example - *No, Saudi TV Didn't Blur Out Michelle*

Headline	Body	Prediction
Zombie Cat: Buried Kitty Believed Dead, Meows Back to Life	Hewlett-Packard is officially splitting in two. Following rumors over the weekend, HP is announcing today that it will separate its PC and printer division from its enterprise and services business...	Unrelated
Man saved from bear attack - thanks to his Justin Bieber ringtone	Justin Bieber may not have been able to take on Orlando Bloom, but he sure as hell was able to take on a bear. It all went down when 42-year-old fisherman Igor Vorozhbitsyns attack by a brown bear was interrupted by a Justin Bieber ringtone	Agree
Kushner Proposed Backchannel Communication with Russians	The idea of a permanent back channel was never discussed, according to the source. Instead, only a one-off for a call about Syria was raised in the conversation	Disagree
Steven Sotloff beheading: Islamic State fighters release video showing death of second US journalist	The Islamic State of Syria and Iraq has reportedly released a video showing the beheading of Steven Joel Sotloff, a U.S. journalist being held by the group	Discuss

Table 2: Sample qualitative results

Obama's Face When the President Met King Salman. Using NER(6), we can identify the persons discussed in the headline and if those same persons are discussed in the article, it is very likely that they are related. We can define a similarity score based on the named entity tags for headline and body. For example, increase the similarity score if the same word is tagged as same entity type in both headline and body. Then, include this score as a feature for the Bag of Words model.

References

[1] William Ferreira, Andreas Vlachos. *Emergent: a novel data-set for stance classification*

[2] Richard Davis, Chris Proctor *Fake News, Real Consequences: Recruiting Neural Networks for the Fight Against Fake News*

[3] *FNC:* www.fakenewschallenge.org/

[4] *FNC-1-Baseline:* https://github.com/FakeNewsChallenge/fnc-1-baseline

[5] *Dependency parser - NLP Stanford:* https://nlp.stanford.edu/software/nndep.html

[6] *Named Entity Recognition* https://nlp.stanford.edu/software/CRF-NER.html

[7] *Stance Detection for Fake news recognition* https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760496.pdf

[8] *Identifying bias heavy sentences in news articles* https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2741414.pdf

[9] *Global vectors for Word Representation* https://nlp.stanford.edu/pubs/glove.pdf

[10] *Neural Stance detectors for fake news challenge* https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761936.pdf

[11] *Decision trees with CNN approach* https://github.com/Cisco-Talos/fnc-1

[12] *Multi layer perceptron approach* https://github.com/hanselowski/athene_system

[13] *Stanford Neural Network Dependency Parser* https://nlp.stanford.edu/software/nndep.shtml

[14] *Single hidden layer network with tf-idf cosine similarity* https://github.com/uclmr/fakenewschallenge

[15] *NLTK SnowBall Stemmer* http://www.nltk.org/howto/stem.html