Report

Clickbait Detection

Similarity-Aware Deep Attentive Model

Project in Statistical Methods in AI

Monsoon'22

Team

Naval Surange (2020113018)

Harshita Gupta (2020101078)

Dhruv Mittal (2020113017)

Siddharth Jain (2020113014)

Abstract

Clickbaits lead to articles that are either misleading or non-informative, making their detection essential. Most recent work handles the clickbait detection problem with deep learning approaches to extract features from the metadata of the content. In this work, we explore the relationship between misleading titles and the target content as an important clue for clickbait detection. We propose a deep similarity-aware attentive model to capture and represent such similarities with better expressiveness. We evaluate our model on two benchmark datasets and compare it with other baseline methods.

Keywords: Clickbait, Recurrent Neural Networks, Bidirectional-GRU, Attention, Global similarity

Contents

1. Introduction

Clickbait is a text or a thumbnail link that is designed to attract attention and entice users to follow that link and read, view, or listen to the linked piece of online content, being typically deceptive, sensationalized, or otherwise misleading. "You will never believe what happened when..." and "This is the biggest mistake you can make..." are two representative titles of clickbait.

1.1 Objectives

- We implement a deep attentive similarity model which is capable of capturing both global and local similarities of the pair of inputs. The model represents local similarities as vectors to combine them with other features for future prediction easily.
- We introduce the ways of either using only similarity information or combining the similarity with other features to detect clickbait. We further employ an attention-based bidirectional Gated Recurrent Units (GRU) model to obtain robust representations of textual inputs.
- We evaluate our framework on two benchmark datasets of clickbait detection. The experimental results demonstrate its effectiveness in detecting clickbait against the baseline methods.

2. Related Work

- Potthast considered the features from both titles and the linked web page, including linguistic information (e.g., the mean word length and sentiment polarity) and side information (e.g., the writer of the titles) and fed the features into traditional classifiers such as logistic regression, Naive Bayes, and random forest and attained the accuracy of around 80
- Zhou's classifier which won first place in the clickbait challenge is a Recurrent Neural Network based framework that considers the context of words, more specifically, a hybrid of bidirectional Gated Recurrent Unit (GRU) and attention model.
- Maria's classifier takes both image and text representations into consideration and different deep learning methods, like Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM).
- Kumal et al. used Siamese Networks for measuring the text and visual similarities and combined the similarities as the input of several fully connected layers.
- Biyani et al. made a few attempts that used the similarities information with several features, including n-grams and metrics for evaluating the informality. They then fed those features into a gradient boost decision tree (GBDT) classifier.

- Zheng et al. transformed the titles into word embeddings and then used text-Convolutional Neural Networks as the classifier.

3. Conceptual discussion

3.1 Recurrent Neural Networks

A recurrent neural network is a type of artificial neural network commonly used in speech recognition and natural language processing. RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

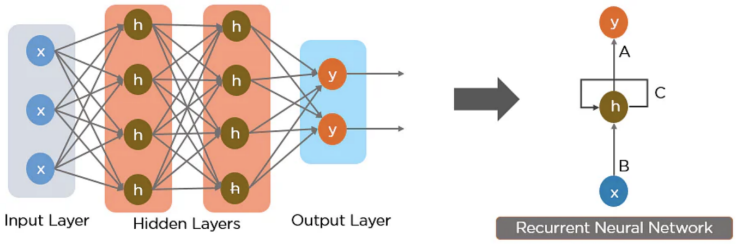


Figure 1: Recurrent Neural Network

RNNs suffer from the problem of vanishing gradients. The gradients carry information used in the RNN, and when the gradient becomes too small, the parameter updates become insignificant. This makes the learning of long data sequences difficult. Hence, we can say that RNNs have short term memory. To tackle this issue LSTM was introduced.

3.2 Long Short-Term Memory

LSTMs are a special kind of RNN capable of learning long-term dependencies by remembering information for long periods is the default behavior.

LSTMs use a series of 'gates' which control how the information in a sequence of data comes into, is stored in and leaves the network. There are three gates in a typical LSTM; forget gate, input gate and output gate. These gates can be thought of as filters and are each their own neural network.

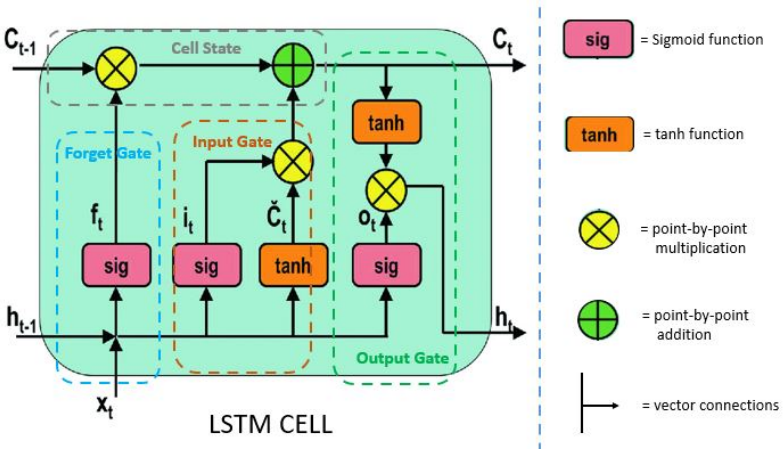


Figure 2: A basic LSTM cell

3.3 Gated Recurrent Unit

GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output.

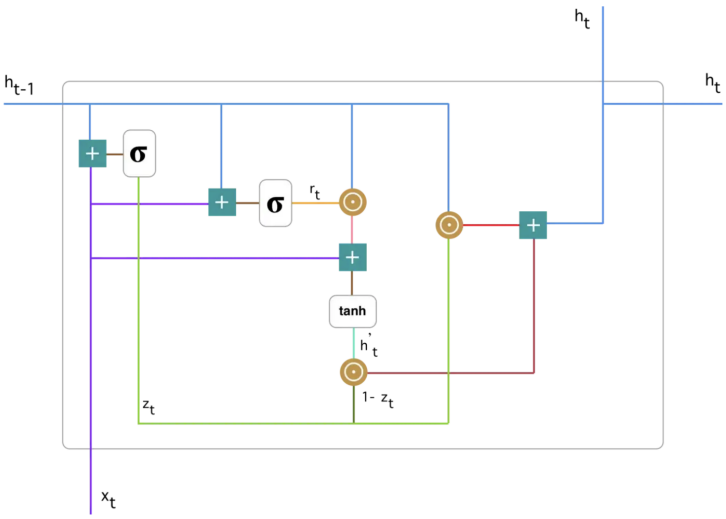


Figure 3: A basic GRU unit

GRUs are able to store and filter the information using their update and reset gates. That eliminates the vanishing gradient problem since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network. If carefully trained, they can perform extremely well even in complex scenarios.

In GRU, there is no explicit memory unit. Memory unit is combined along with the network. There is no forget gate and update gate in GRU. They are both combined

together and thus the number of parameters are reduced. When comparing GRU with LSTM, it performs good but may have a slight dip in the accuracy. But still we have less number of trainable parameters which makes it advantageous to use.

3.4 Bidirectional GRU

In a bidirectional RNN, we consider 2 separate sequences. One from right to left and the other in the reverse order. Consider the word sequence "I love mango juice". The forward layer would feed the sequence as such. But, the Backward Layer would feed the sequence in the reverse order "juice mango love I". Now, the outputs would be generated by concatenating the word sequences at each time and generating weights accordingly.

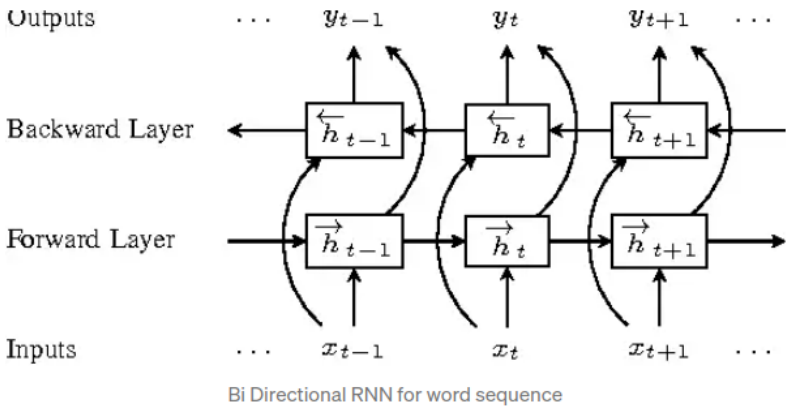


Figure 4: Bi directional RNN for word sequence

3.5 Deep Semantic Similarity Model

DSSM is a latent semantic model. It uses deep neural networks to learn the latent representations. DSSM first maps the input features x to the latent semantic space l,

$$
\begin{aligned}
layer_1 &= W_1 x \\
layer_i &= f(W_i layer_{i-1} + b_i), i = 2, ..., N-1 \\
l &= f(W_N layer_{N-1} + b_N)
\end{aligned}
$$

where $layer_i$ is the i^{th} intermediate hidden layer, W_i is the i^{th} weight matrix, b_i is the i^{th} bias matrix, and f is the activation function, e.g., sigmoid function.

From this, the semantic relevance score between say query Q and document D is measured as:

$$R(Q, D) = cosine(l_Q, l_D) = \frac{l_Q^T l_D}{||l_Q||\ ||l_D||}$$

Learning the DSSM is equivalent to maximising this similarity score for matching documents from the entire collection. A typical improvement over DSSM is to change from deep neural networks to Convolutional DSSM or LSTM-DSSM. In our work, we follow this idea of calculating the similarities in the latent space but in a different way

(attention-based bidirectional GRU). We regard this semantic relevance score as global similarity and additionally, learn a local similarities vector for prediction.

3.6 Attention mechanism in Deep Learning

Attention mechanism was originally designed in the context of Seq2Seq Models.

A seq2seq model is normally composed of an encoder-decoder architecture, where the encoder reads the input sequence and summarizes the information in something called as the internal state vectors (in case of LSTM these are called as the hidden state and cell state vectors). We discard the outputs of the encoder and only preserve the internal states. Decoder's initial states are initialized to the final states of the Encoder. Using these initial states, decoder starts generating the output sequence.

A fixed-length context vector design reflects the inability of the system to retain longer sequences. Often it forgets the earlier elements of the input sequence once it has processed the complete sequence. The attention mechanism was created to resolve this problem of long dependencies.

The core idea here is each time the model predicts an output word, it only uses parts of the input where the most relevant information is concentrated instead of the entire sequence. In simpler words, it only pays attention to some input words.

Attention is an interface connecting the encoder and decoder that provides the decoder with information from every encoder hidden state. With this framework, the model is able to selectively focus on valuable parts of the input sequence and hence, learn the association between them.

This helps the model to cope efficiently with long input sentences.

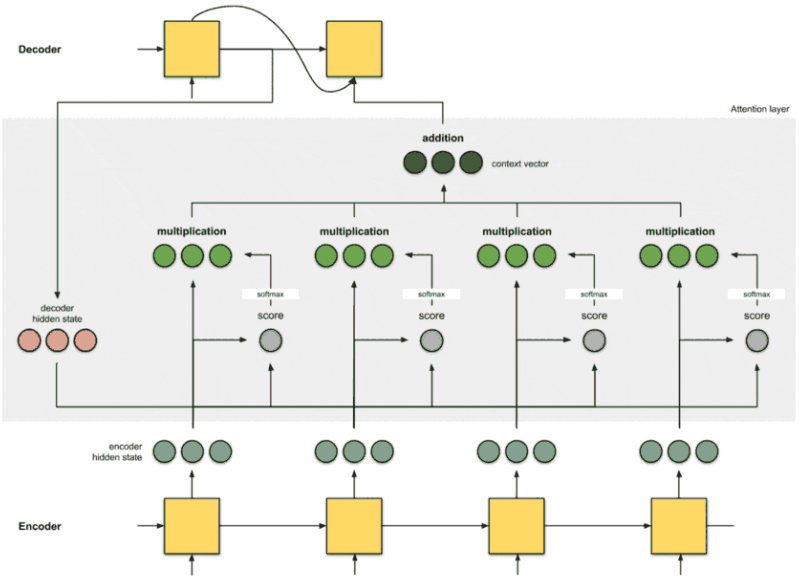


Figure 5: Attention interface

The idea is to keep the decoder as it is, and we just replace sequential RNN/LSTM

with bidirectional RNN/LSTM in the encoder.

The implementation of an attention layer can be broken down into 6 steps:

- Prepare hidden states - In the above example, we have 4 encoder hidden states and the current decoder hidden state.
- Obtain a score for every encoder hidden state - The score is obtained by a alignment score function. In our example, it is the dot product between the encoder and decoder hidden states.
- Run all the scores through a softmax layer - We put the scores to a softmax layer so that the softmax scores add up to 1.
- Multiply each encoder hidden state by its softmax score - By multiplying each encoder hidden state with its softmax score , we obtain the alignment vector or the annotation vector
- Sum the alignment vectors - The alignment vectors are summed up to produce the context vector. (aggregated information)
- Feed the context vector into the decoder

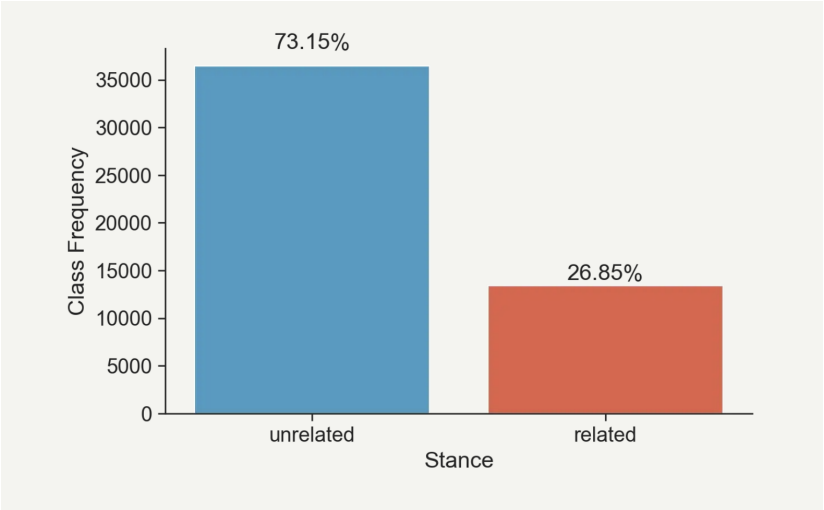
4. Data

Here we use two datasets for evaluating the model.

4.1 Fake News Challenge

FNC dataset is from the Fake News Challenge in 2017. The data describe pairs of titles and bodies and are labeled as 'agree', 'disagree', 'discuss' and 'unrelated'. We regard data with label 'unrelated' as clickbait. The dataset contains 49,972 pairs of titles and bodies for training and 25,413 pairs for the testing.

Given that this is a multi-class classification problem, we should first observe the distribution of the unrelated, agree, disagree, discuss labels. In the following figures, on the x-axis we have the specific class and on the y-axis we have the count of each class represented by the length of each bar. Above each bar is the percentage of the examples having the corresponding class label in the training dataset.



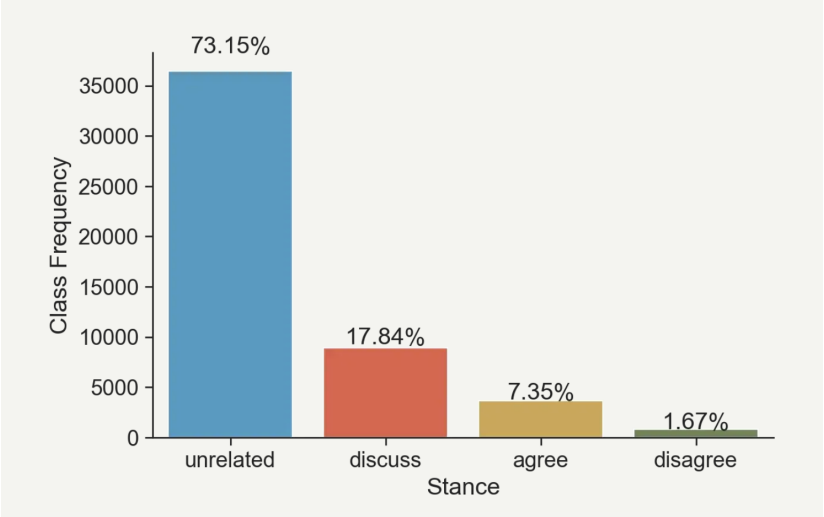


Figure 6: FNC Training Data Class Label Distribution

The processed FNC dataset has an average of 8 words in the titles and 200 words in the bodies.

4.2 Clickbait Challenge

Clickbait Challenge is a benchmark dataset for clickbait detection that released in 2017. The dataset contains over 20,000 labelled pairs of posts for training and validation. A higher score in 0 to 1 stands for the higher probability of a post being clickbait. A post with a mean clickbait score over 0.5 is considered to be clickbait.All posts were annotated on a 4-point scale [not click baiting (0.0), slightly click baiting (0.33), considerably click baiting (0.66), heavily click baiting (1.0)] by five annotators from Amazon Mechanical Turk.

Every data point consists of a JSON-object that looks like this:

```
{
"id": "608999590243741697",
"postTimestamp": "Thu Jun 11 14:09:51 +0000 2015",
"postText": ["Some people are such food snobs"],
"postMedia": ["608999590243741697.png"],
"targetTitle": "Some people are such food snobs",
"targetDescription": "You'll never guess one...",
"targetKeywords": "food, foodfront, food waste...",
"targetParagraphs": [
"What a drag it is, eating kale that isn't ...",
"A new study, published this Wednesday by ...",
...],
"targetCaptions": ["(Flikr/USDA)"]
}
```


Classifiers have to output a clickbait score in the range [0,1], where a value of 1.0 denotes that a post is heavily click baiting.


```
{"id": "608999590243741697", "clickbaitScore": 1.0}
```


Figure 7: Input and Output format of Clickbait challenge dataset

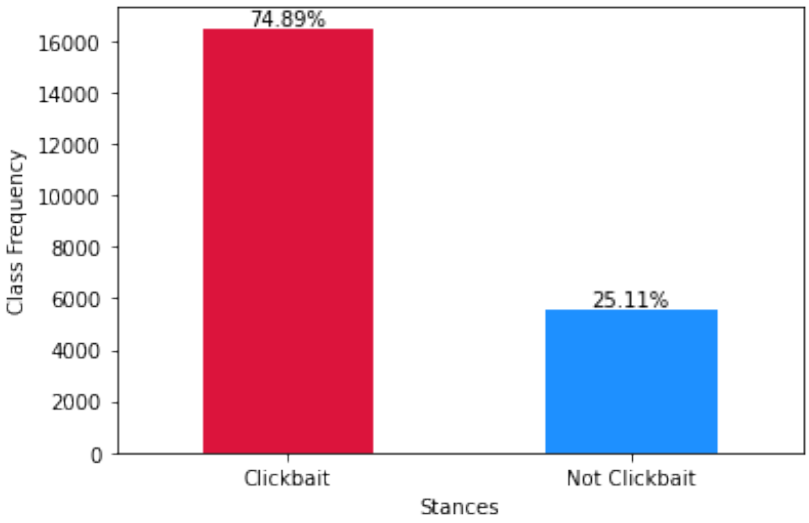


Figure 8: Clickbait Data Class Label Distribution

The Clickbait challenge dataset available online has 100 GB worth of data in the form of images which first needs to be converted into texts and then pre-processed by removing stop words and stemming. The processed Click-bait Challenge dataset has an average of 10 words in the titles and 50 words in the bodies

5. Methodology

5.1 Data Preprocessing

We first preprocess the data by tokenizing, removing the stop words and perform stemming. Stop words are common words that are very common in the language and are filtered out since their contribution is insignificant. This helps in reducing the size of the data and speeds up encoding and subsequent steps. We use NLTK library to remove the stop words.
Stemming is a technique used to extract the base form of the words by removing affixes from them. It is done to normalize the text. We choose to do stemming over lemmatization as the improvement in performance due to lemmatization is not worth the extra computation required. NLTK library is used to perform stemming.

5.2 Sequence Encoding

After the data has been preprocessed, we must encode the words in order to convert them to a computer-understandable form. We first convert each word in our dataset to a single and unique integer such that there is a one-to-one mapping between them, this integer is called an index of the word. Next, we truncate all the sequences that contain more than 200 words and pad all the sequences which have less than 200 words so that there is an equal number of words(or index) in every sequence.

5.3 Vectorization

The vectorization of encoded data is handled by an Embedding layer provided by the Keras API, The Layer is configured to convert every word index to an embedding vector of size 100, This is done for both headings and bodies of both FNC and ClickBait datasets.

5.4 Learning Latent Representations

Here we consider transform the titles H and bodies B into the latent representations: L_H and L_B, where $L_H, L_B \in \mathbb{R}^M$. We apply the attention-based bidirectional GRU using TensorFlow, to obtain hidden representations by using a gating mechanism to track the state of sequences without using separate memory cells. Given a $b_i, i \in [1, N]$, $w_{i,t}$ is the set of word embedding vectors, where $t \in [1, T_i], T_i$ is the number of words in body i which, here, will be 50. We use the bidirectional GRU to get annotations of words by summarizing information from both directions of a word. We get $w'_{i,t}$ by concatenating the forward hidden state and the backward hidden state. And an attention mechanism is used to extract words that are important to the sentence and aggregate the representation of those words to get the latent representation L_{b_i}. Finally, we get the latent representation of the bodies L_B. In a similar way, we can obtain the hidden representation L_H.

5.5 Learning the Similarities

To obtain the global similarities we calculate the cosine similarities between L_H and L_B.

$$r(H, B) = \frac{L_H^T L_B}{||L_H||\,||L_B||}$$

A higher value of $r(H, B)$ stands for a higher level of consistency between the titles and bodies. Intuitively, we want to maximize this similarity score between the matching titles and minimize the similarity score between the mismatching pairs. For using only global similarity to predict the clickbait, we use an in-built function and optimize the loss function using Adam optimizer. We learn the local similarities for a better matching representation. We set the local block size as $\mu, \mu < M$, and we move from left to right with $\nu, \nu < (M - \mu)$ strides to the next local block. Then we have $K = \left[\frac{M-\mu}{\nu}\right]$ local blocks, that is, the latent L_H can be represented as $L_H = [L_{H,1}^T, L_{H,2}^T, ..., L_{H,K}^T]^T$ and so as the L_B. Thus, the local similarities is calculated by

$$LS(H, B) = (r(L_{H,1}, L_{B,1}), ..., r(L_{H,K}, L_{B,K}))^T$$

We apply the self-attention mechanism to select the most useful similarities.

$$A = softmax(V_a tanh(W_a LS(H, B)^T))$$

$$P = softmax(W_P(A \times LS(H, B)) + b_P)$$

where $W_a \in \mathbb{R}^K$ and $V_a \in \mathbb{R}^{K \times K}$ are two weight matrices, A is the attention matrix for the local similarity and W_P and b_P are weights and biases. The prediction for the clickbait is $\hat{y} = argmax_y P$. For using only local similarities to predict the clickbaits, we choose the combination of cross entropy and regularization as the loss function and optimize it using Adam optimization method.

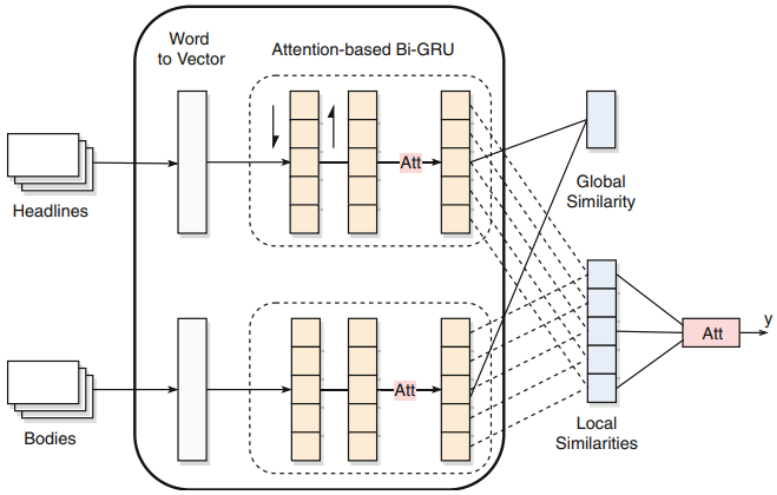


Figure 9: shows the learning of the Latent Representations along with learning of the global similarity and local similarities.

5.6 Learning for Prediction

We will introduce the classification method which combines the features with the similarities. To combine the information from the raw text with the similarities, we adopt an attentive way for the final prediction. We first use fully connected layers to map the hidden representations L_H and L_B into layers with K dimension.

$$L'_H = f(W_H L_H + b_H)$$

$$L'_B = f(W_B L_B + b_B)$$

We concatenate L'_H and L'_B with LS(H,B) to obtain L' and apply self-attention on it to get the combination layer L''. The combination layer is then fed into multilayer perceptrons and we get the $P = softmax(W_P L'' + b_P)$. Then the prediction is $\hat{y} = argmax_y P$. Similarly, we set the loss as the combination of cross entropy and L2-norm of the parameters, and we learn the parameters with Adam optimization.

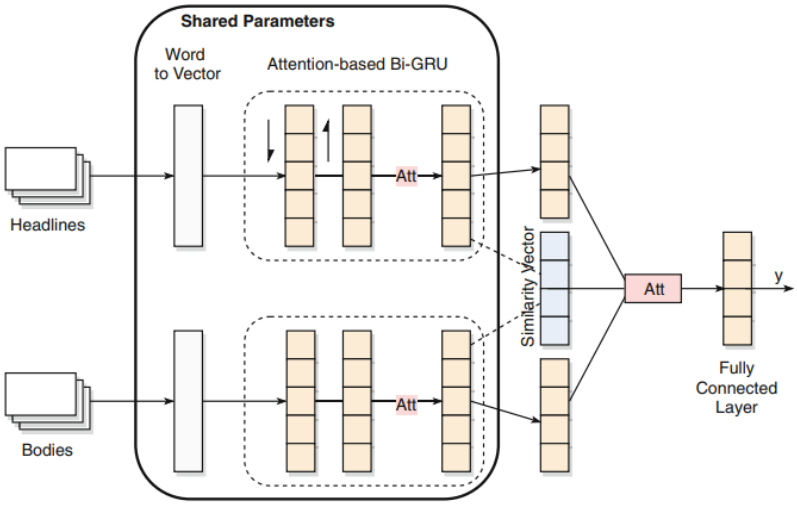


Figure 10: shows learning for Prediction

6. Results and Analysis

6.1 Experimental Setting

```
Train Heading Shape:  (21824, 200)
Train Bodies Shape:  (21824, 200)
Train Labels Shape:  (21824, 1)
----------------------------------
Test Heading Shape:  (18944, 200)
Test Bodies Shape:  (18944, 200)
Test Labels Shape:  (18944, 1)
```

Figure 11: Number of training and test samples in the Clickbait Challenge dataset

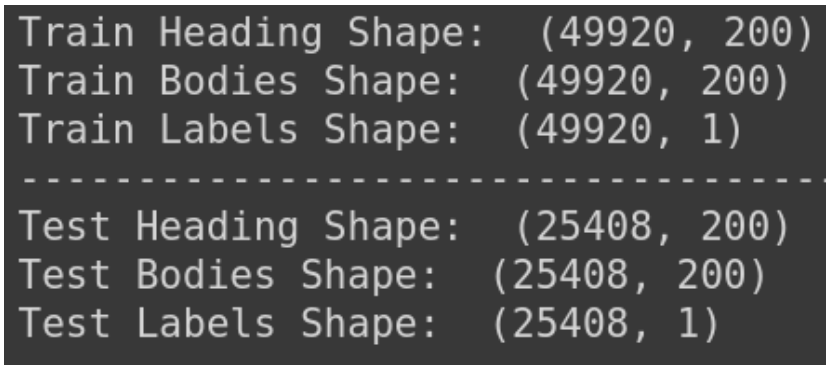


Figure 12: Number of training and test samples in the FNC dataset

We initialize the weight and bias parameters with random variables.

- Word embedding dimension = 100
- Hidden size = 50
- Number of epochs = 5
- Block size = 50
- Learning rate = 0.001
- Lasso regularization, rate = 0.05

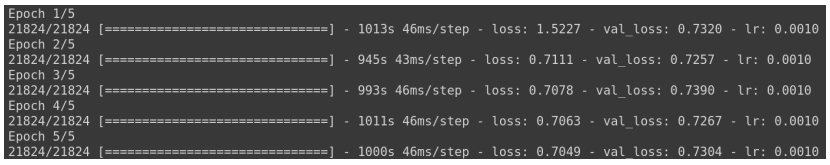


Figure 13: Training the model for Clickbait Challenge dataset

The evaluation is conducted with four commonly used metrics: accuracy, recall, precision, and F1 score.

Accuracy	Precision	Recall	F1-score
0.762	0.58	0.76	0.66

Table 1: Classification report on the Clickbait Challenge dataset

Accuracy	Precision	Recall	F1-score
0.722	0.51	0.72	0.61

Table 2: Classification report on the FNC dataset

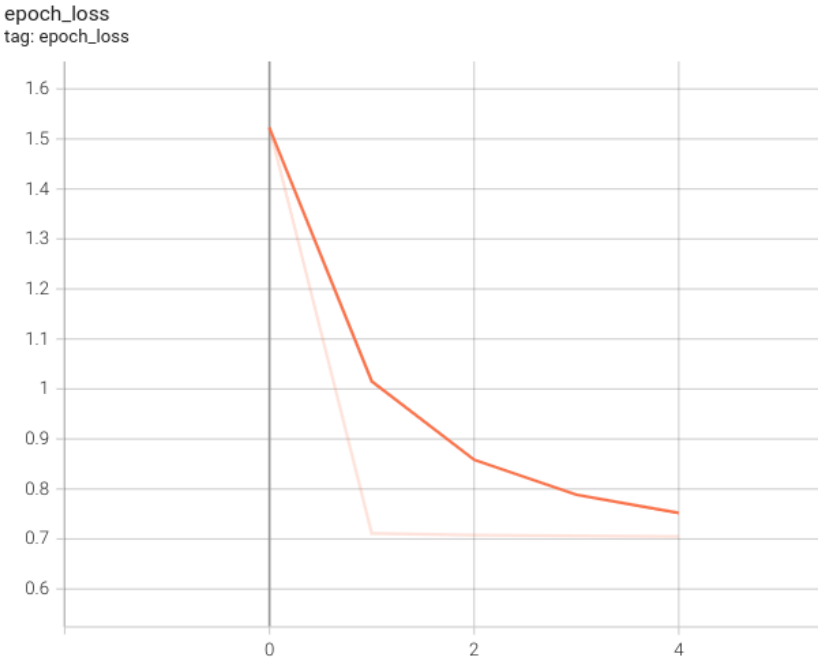


Figure 14: Epoch loss for Clickbait Challenge dataset

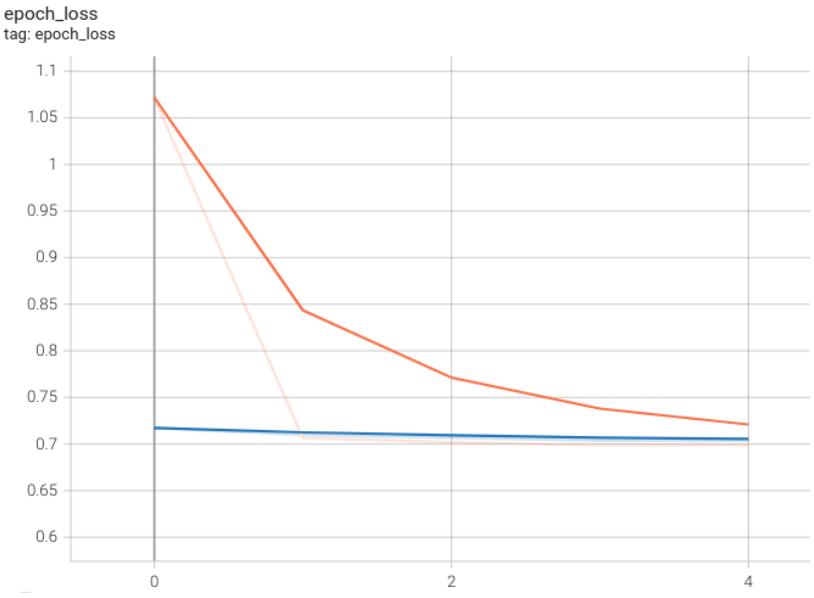


Figure 15: Epoch loss for FNC dataset

6.2 Comparisons

Table 1. Comparison results

Methods	Clickbait Challenge				FNC dataset			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Huang et al. [6]	0.817	0.655	0.661	0.658	0.747	0.894	0.740	0.811
Shen et al. [14]	0.833	0.683	0.643	0.662	0.756	0.959	0.762	0.853
Kumar et al. [7]	0.826	0.699	0.474	0.565	0.859	0.920	0.877	0.907
Zheng et al. [19]	0.844	0.654	0.653	0.653	0.789	0.852	0.845	0.857
Glenski et al. [5]	0.827	0.642	0.621	0.631	0.868	0.925	0.884	0.913
Zhou et al. [20]	0.856	0.719	0.650	0.683	0.879	0.924	0.897	0.919
LSD	0.847	0.697	0.675	0.686	0.885	0.928	0.901	0.923
LSDA	0.860	0.722	0.699	0.710	0.894	0.933	0.912	0.928

It is observed that both the CNN- and RNN-based models perform better than traditional deep neural networks. The attention-based bidirectional GRU shows superior performance in dealing with textual information. Both the similarity information and the attention mechanism helps with the final prediction.

7. Limitations and Challenges

7.1 Computational Challenges

The Clickbait challenge dataset available on their official website was very large with about 100 GB size. The training and testing dataset for both FNC and Clickbait challenge has more than 20,000 labelled pairs of titles and bodies.

As a result preprocessing, vectorization and training the data took around few hours. Additionally, debugging the code required us to run the above parts of the code again which was computationally expensive.

7.2 Extracting Relevant Information from Datasets

For clickbait challenge, we had images in the dataset which made it difficult to download the whole data because of large size. The format was given in a JSONL File format and only some information was relevant.

Similarly, for FNC dataset bodies and headings were given in different files which required us to map the headings to their corresponding bodies as multiple headings had same body.

7.3 Construction of Model

The model required the merging of two bidirectional GRUs such that each of them run parallelly but the output of both affect each other's loss functions. This design is not very intuitive and no relevant sources can be found for such a design making it very difficult to come up with the idea to implement it.

7.4 New to TensorFlow

Implementation of RNN or any form of Neural Network was completely new to us because of which understanding and getting used to the functions and libraries in the allotted time of the project component has been a challenge. We faced several errors which had little to no documentation causing us to spend considerable amount of time in debugging.

8. Conclusion

In this work, we solve the problem of clickbait detection from the similarity perspective, as opposed to the traditional feature engineering which lack the properties in representing the matching information between titles and targeted bodies. We have presented a local similarity-aware deep attentive model that learns global similarity, local similarities and raw input features to make predictions in an attentive manner. The model yields competitive results against other baseline methods on two real world datasets.

9. Work Distribution

- Preprocess FNC data - Siddharth
- Preprocess Clickbait dataset - Siddharth
- Implement vectorization - Harshita
- Encodings for FNC data - Dhruv, Siddharth
- Encodings for Clickbait data - Naval, Siddharth
- Map encodings for stances and bodies - Harshita
- Implement Bidirectional GRU layer - Naval, Dhruv
- Implement Attention with context layer - Naval, Harshita
- Global Similarity - Harshita
- Local Similarity - Harshita, Siddharth
- Train model (Learning the similarities) - Naval, Dhruv
- Preprocessing and encoding test data - Dhruv
- Testing the model - Naval, Dhruv
- Comparison with baseline methods - Harshita, Siddharth
- Report - Harshita, Siddharth, Dhruv
- Presentation - Harshita, Siddharth, Dhruv

10. References

19 Zheng, H.T., Chen, J.Y., Yao, X., Sangaiah, A.K., Jiang, Y., Zhao, C.Z.: Clickbait convolutional neural network. Symmetry 10(5), 138 (2018)

5 Glenski, M., Ayton, E., Arendt, D., Volkova, S.: Fishing for clickbaits in social images and texts with linguistically-infused neural network models. arXiv preprint arXiv:1710.06390 (2017)

20 Zhou, Y.: Clickbait detection in tweets using self-attentive network. arXiv preprint arXiv:1710.05364 (2017)

6 Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: International Conference on Information Knowledge Management, pp. 2333–2338. ACM (2013)

14 Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: ACM International Conference on Conference on Information and Knowledge Management, pp. 101–110. ACM (2014)

7 Kumar, V., Khattar, D., Gairola, S., Kumar Lal, Y., Varma, V.: Identifying clickbait: a multi-strategy approach using neural networks. In: The 41st International ACM SIGIR Conference on Research Development in Information Retrieval, pp. 1225–1228. ACM (2018)

8 LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436 (2015)

18 Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489 (2016)

12 Potthast, M., K¨opsel, S., Stein, B., Hagen, M.: Clickbait detection. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 810–817. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1 72

2 Biyani, P., Tsioutsiouliklis, K., Blackmer, J.: 8 amazing secrets for getting more clicks: detecting clickbaits in news streams using article informality. In: AAAI, pp. 94–100 (2016)