Similarity-Aware Deep Attentive Model for Clickbait Detection

Team Name: Ego et al.

Tanmay Bhatt - 2020112017
Abhinav Siddharth - 2020112007
Anjali Singh - 2020102004
Pranav Manu - 2020112019

Index

- Motivation
- Objective
- Existing Work
- Understanding the Proposed Method
- Results
- Limitations of the method
- Conclusion
- Datasets

Motivation

Clickbaits are a type of web links designed to entice users to enter specific web-pages or videos.

We come across clickbait text and images multiple times each day. Usually, such links will lead to non-informative and misleading articles which causes the reader to stray from the topic which they were actually looking for.

Hence, it becomes imperative to come up with methods for automated clickbait detection such that users can be warned against potential clickbait material.

Objective

The objective of our project is to build a deep similarity-aware attentive model for capturing the discriminative information from local and global similarities.

In other words, given the headline and the article body, the model should be able to classify the text as clickbait or not.

Some examples of clickbait text are

- You'll never believe what happened after …
- How to achieve unbelievable results with this 1 weird trick!

Existing Work

Clickbait detection has been a fairly sought after topic in the recent years and hence considerable amount of work has been put forward toward the same

The first few approaches extracted linguistic features from both the header and the body text and passed these features on traditional classifiers like logistic regression, Bayes classifier, random forest, etc.

More later works employ CNNs and RNNs to look for similarities between the header text and the body to make the classification.

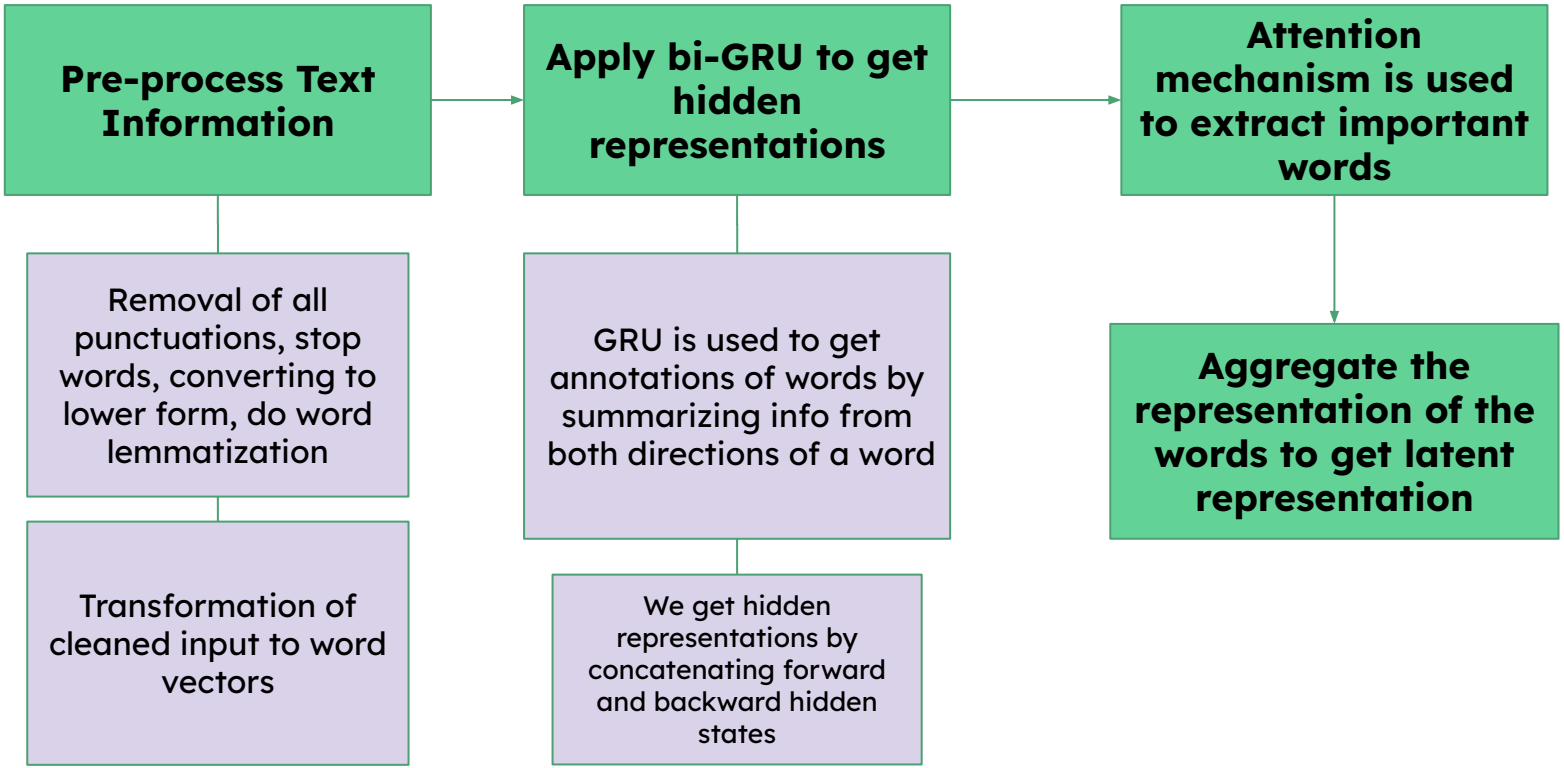
Understanding the Proposed Method

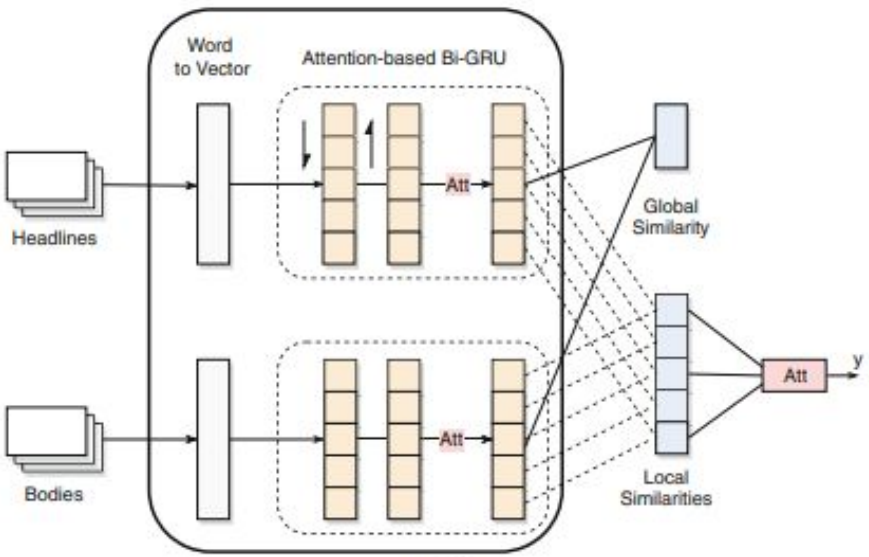
- We propose a model that will exploit both the local and global similarities between the header text and the body text. Moreover, the similarities are modelled as vectors so that they can be augmented with other features for future prediction easily.
- Given a set of titles, $H = \{h_1, h_2, ..., h_N\}$ and their bodies, $B = \{b_1, b_2, ..., b_N\}$ and the labels, $Y = \{y_1, y_2, ..., y_N\}$; $y_i = 1$ if the headline is a clickbait.

The framework in the paper includes: -

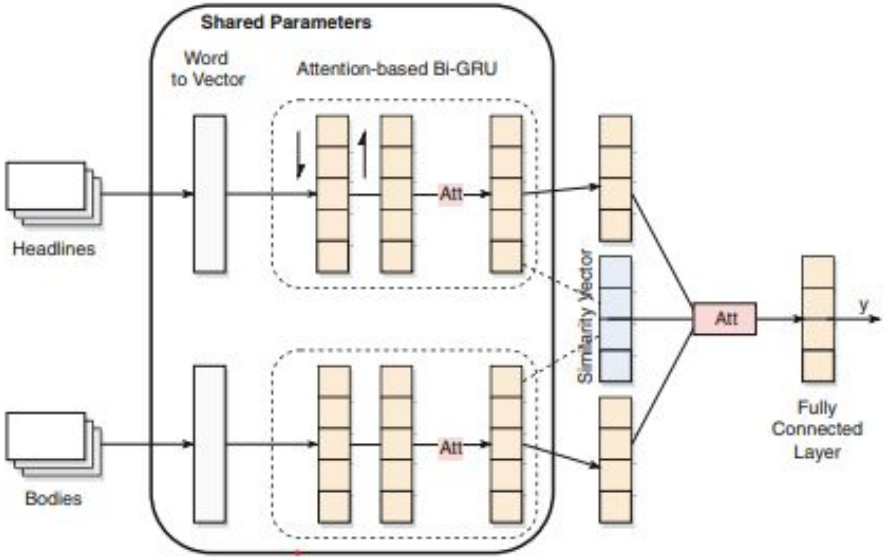
- Learning latent representations
- Learning the similarities
- Using the similarities for further predictions

Learning the Latent Representation



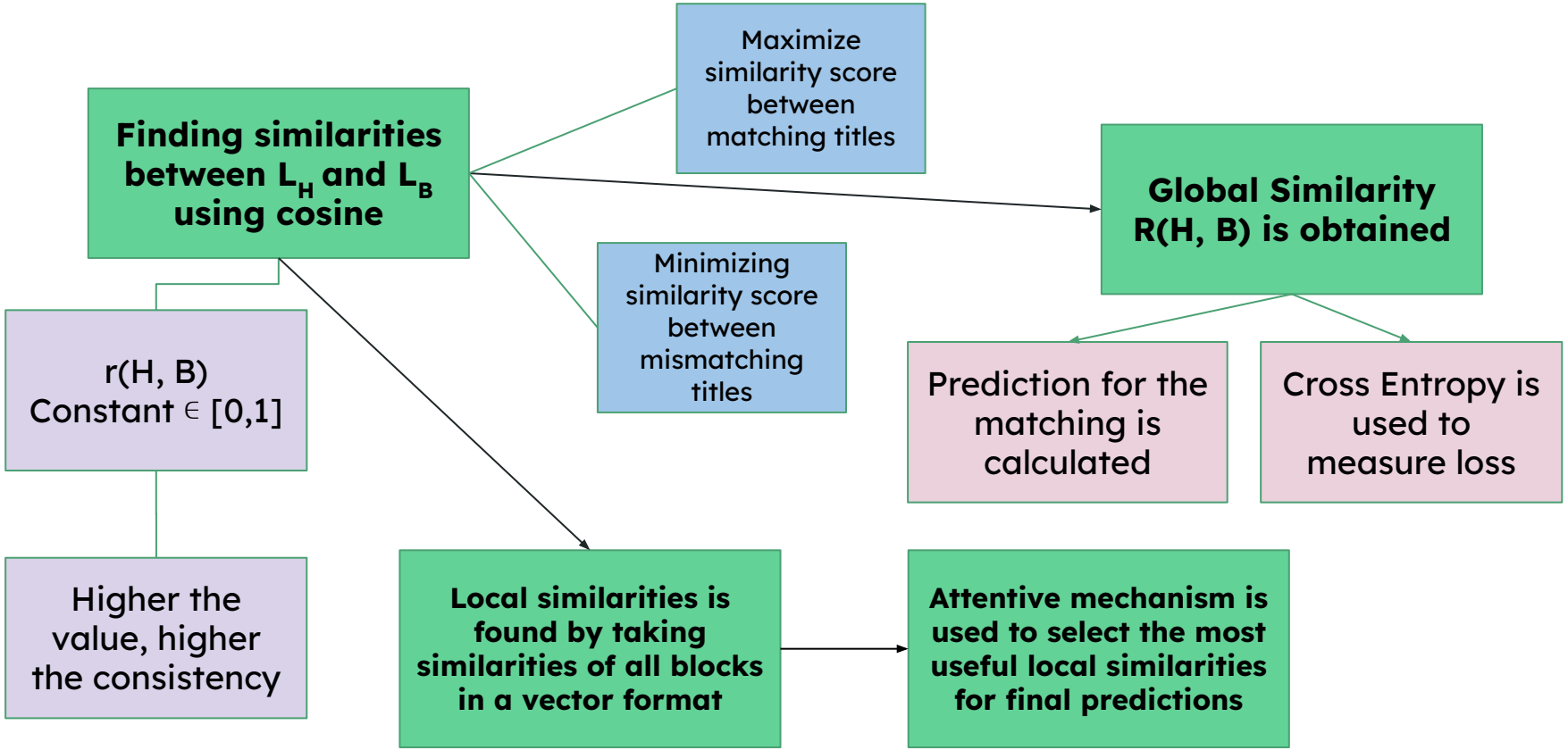

Word
to Vector
Attention-based Bi-GRU
Headlines
Bodies
Att
Global
Similarity
Local
Similarities
Att
y

(a)


Shared Parameters
Word
to Vector
Attention-based Bi-GRU
Headlines
Bodies
Att
Similarity Vector
Att
y
Fully
Connected
Layer

(b)

Learning the Similarities



Using the Similarities for further Predictions

Once we have the similarities, we can combine them with standard clickbait indicators like writing style and text quality. In order to combine these, we make use of the latent representations.

$$L'_H = f(W_H L_H + b_H)$$
$$L'_B = f(W_B L_B + b_B)$$

We then calculate self attention values which are representative of the writing style and text quality and combine it with the latent representations to get a final combination layer on which multilayer perceptron will be applied to get the prediction.

Layer Information

```
LSDA(
  (LSD_model): LSD(
    (head_Model): SimilarityAware(
      (bi_gru): GRU(100, 120, num_layers=2, batch_first=True, bidirectional=True)
      (att): Linear(in_features=240, out_features=1, bias=True)
      (tanh): Tanh()
      (softMax): Softmax(dim=1)
    )
    (body_Model): SimilarityAware(
      (bi_gru): GRU(100, 120, num_layers=2, batch_first=True, bidirectional=True)
      (att): Linear(in_features=240, out_features=1, bias=True)
      (tanh): Tanh()
      (softMax): Softmax(dim=1)
    )
  )
  (mlpLayer): Linear(in_features=240, out_features=60, bias=True)
  (att1): myLinear(in_features=180, out_features=1, bias=False)
  (att2): myLinear(in_features=180, out_features=180, bias=False)
  (mlpLayer2): Linear(in_features=180, out_features=2, bias=True)
  (tanh): Tanh()
  (sigmoid): Sigmoid()
  (softMax1): Softmax(dim=1)
)
```

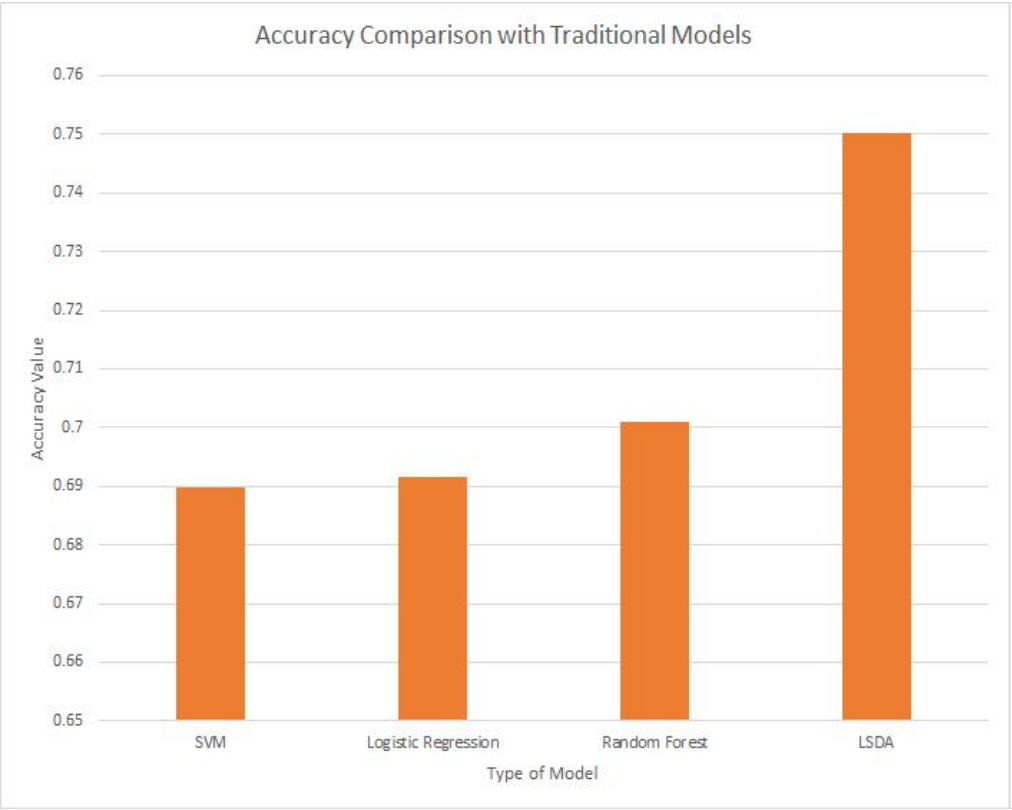
Layer Information

Layer (type:depth-idx)	Input Shape	Output Shape	Param #	Mult-Adds
LSDA	[20, 96, 100]	[20, 2]	5,025	--
├─LSD: 1-1	[20, 96, 100]	[20, 200]	--	--
│ └─SimilarityAware: 2-1	[20, 96, 100]	[20, 200]	--	--
│ │ └─GRU: 3-1	[20, 96, 100]	[20, 96, 200]	121,200	232,704,000
│ │ └─Linear: 3-2	[20, 96, 200]	[20, 96, 1]	201	4,020
│ │ └─Tanh: 3-3	[20, 96, 1]	[20, 96, 1]	--	--
│ │ └─Softmax: 3-4	[20, 96, 1]	[20, 96, 1]	--	--
│ └─SimilarityAware: 2-2	[20, 96, 100]	[20, 200]	--	--
│ │ └─GRU: 3-5	[20, 96, 100]	[20, 96, 200]	121,200	232,704,000
│ │ └─Linear: 3-6	[20, 96, 200]	[20, 96, 1]	201	4,020
│ │ └─Tanh: 3-7	[20, 96, 1]	[20, 96, 1]	--	--
│ │ └─Softmax: 3-8	[20, 96, 1]	[20, 96, 1]	--	--
├─myLinear: 1-2	[20, 1, 25]	[20, 25, 25]	25	500
├─Tanh: 1-3	[20, 25, 25]	[20, 25, 25]	--	--
├─myLinear: 1-4	[20, 25, 25]	[20, 25, 25]	625	12,500
├─Softmax: 1-5	[20, 625]	[20, 625]	--	--
├─Linear: 1-6	[20, 25]	[20, 2]	52	1,040
├─Softmax: 1-7	[20, 2]	[20, 2]	--	--

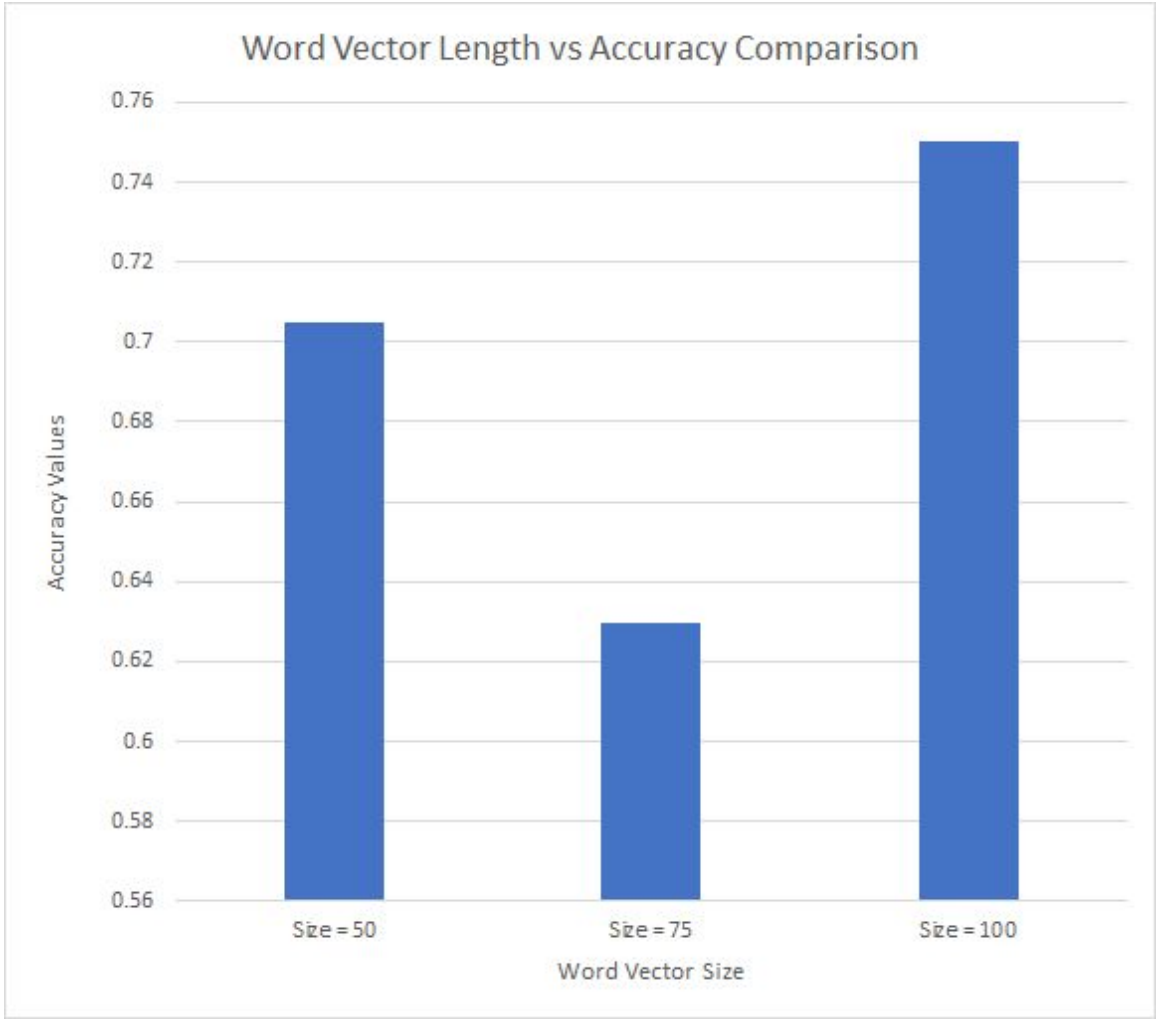
Total params: 248,529
Trainable params: 248,529
Non-trainable params: 0
Total mult-adds (M): 465.43

Input size (MB): 1.54
Forward/backward pass size (MB): 6.38
Params size (MB): 0.97
Estimated Total Size (MB): 8.89

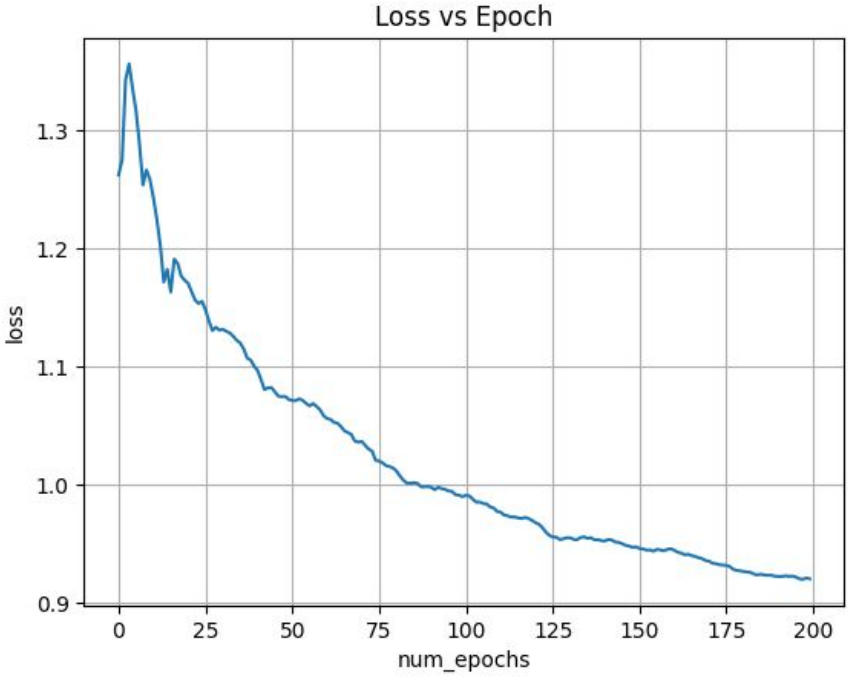
Results

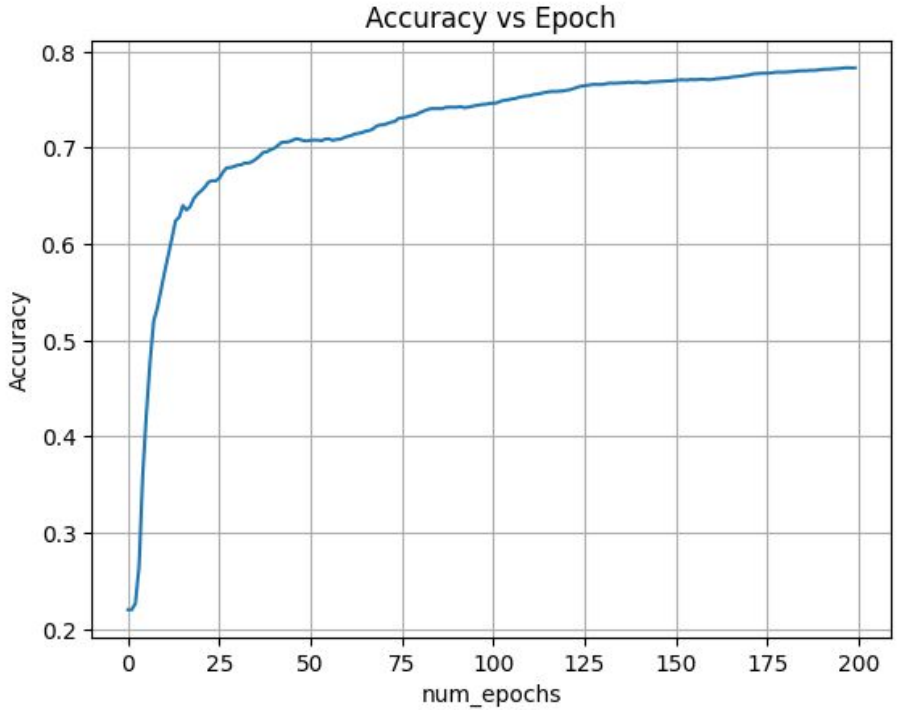

Accuracy Comparison with Traditional Models
0.76
0.75
0.74
0.73
0.72
0.71
0.7
0.69
0.68
0.67
0.66
0.65
Accuracy Value
SVM
Logistic Regression
Random Forest
LSDA
Type of Model

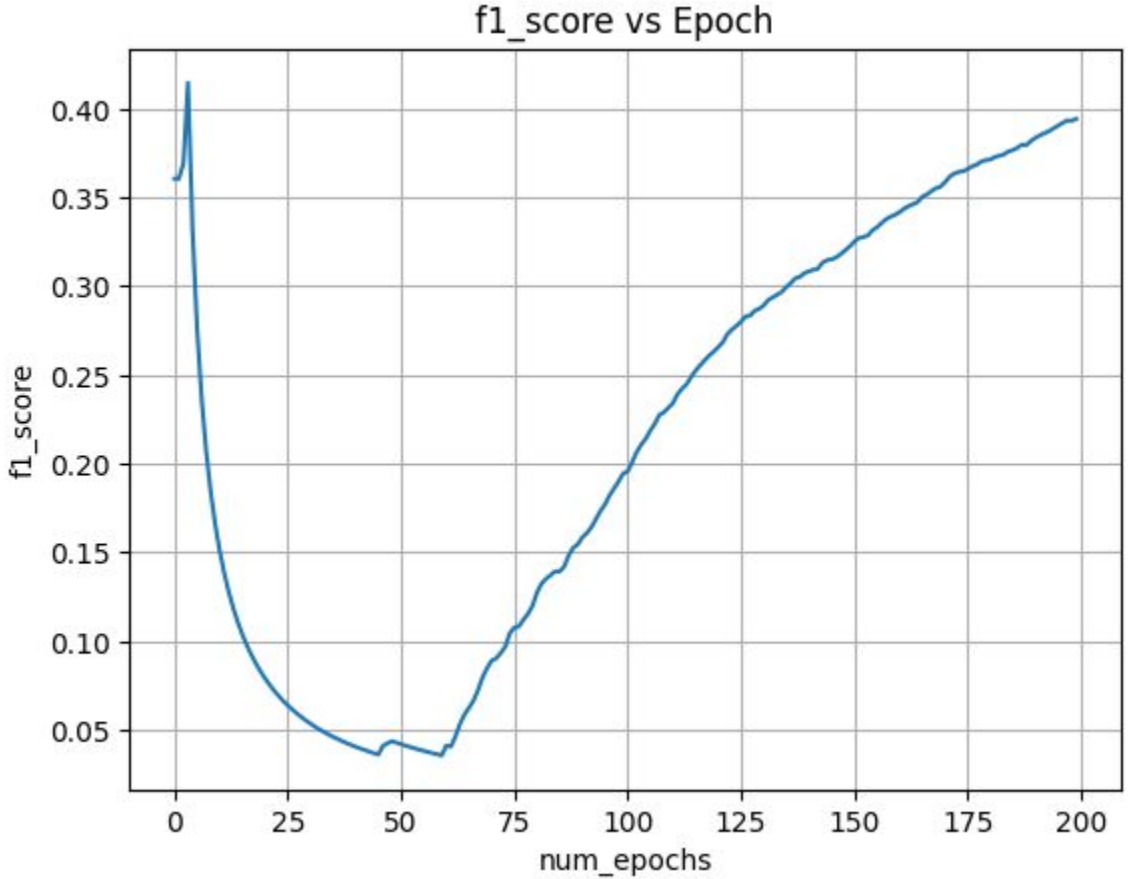

Local Similarity Block Size vs Accuracy Comparison
0.76
0.74
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
Accuracy Value
K = 20
K = 25
K = 50
K = 75
Block Size


Word Vector Length vs Accuracy Comparison
0.76
0.74
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
Accuracy Values
Size = 50
Size = 75
Size = 100
Word Vector Size

The accuracy of the training set came out to be 81.99%.



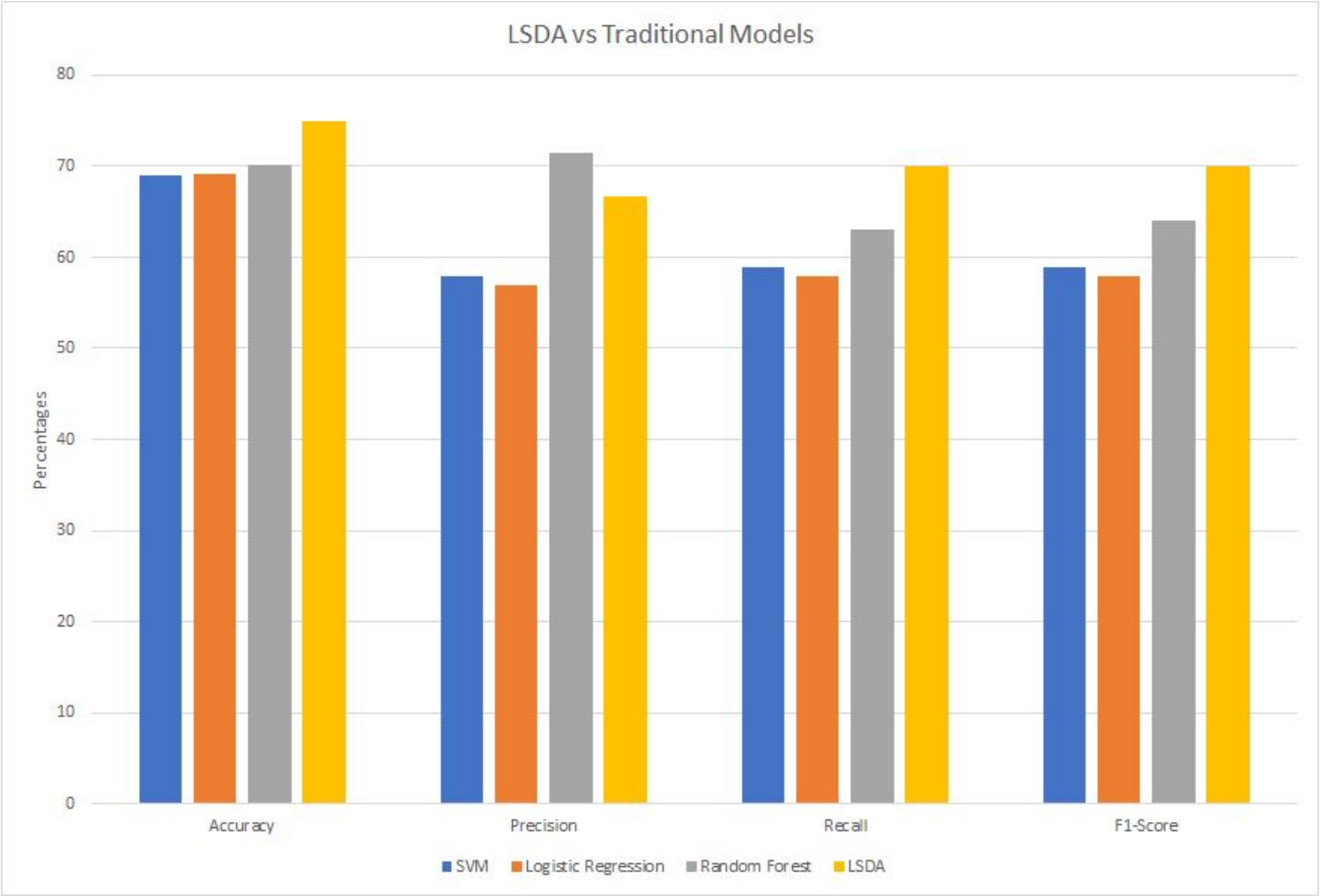



f1_score vs Epoch
f1_score
num_epochs
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
25
50
75
100
125
150
175
200

Comparisons with other Models

Trained and Tested on Clickbait Dataset - 2017. These are all weighted-averages.

Methods	Accuracy	Precision	Recall	F1-Score
SVM	61.78%	58%	59%	59%
Logistic Regression	64.22%	57%	58%	58%
Random Forests	72.35%	71.5%	63%	64%
LSDA	75.02%	66.66%	69.99%	69.


LSDA vs Traditional Models
80
70
60
50
40
30
20
10
0
Percentages
Accuracy
Precision
Recall
F1-Score
SVM
Logistic Regression
Random Forest
LSDA

Limitations of the method

The model considered had the following shortcomings: -

- The model doesn’t consider patterns or styles of the titles. For example, the sentence, “You WON'T believe what happened next?”, would be converted to lowercase and an important cue for clickbait would be lost.
- The model also ignores pictorial information while detecting clickbait text.
- If the word to vector size is too small, the output of the deep learning model built overflows. This is called the Gradient Explosion Problem caused due to large updates to the neural network model while training.

Conclusion

In our project, we successfully implemented the similarity method to solve the problem of clickbait detection by exploiting global as well as local similarities, as opposed to the traditional feature engineering which lack the properties in representing the matching information between titles and targeted bodies and have gotten a rough accuracy value of 75.0255%. We have presented a local similarity-aware deep attentive model that learns both local similarities and raw input features to make predictions in an attentive manner.

Datasets

- Clickbait Challenge - link
- FNC Dataset - link

Thank You :)