# SMAI Final Presentation

## Similarity-Aware Deep Attentive Model for Clickbait Detection


- Parul Ansal (2020201036)
- Shweta Arya (2020201047)
- Anshul Kakraniya (2020201008)

# Understanding of the Problem Statement

- **Clickbait** refers to a certain kind of web content advertisement that is designed to entice it's users into clicking an accompanying link.
- Most recent work handles the clickbait detection problem with deep learning approaches to extract features from the metadata of content. However, little attention has been paid to the relationship between the misleading titles and the target content, which we found to be an important clue for enhancing clickbait detection.
- Existing method (linear methods) lack the expressiveness when compared to non-linear method.
- There has been no consideration given to  hidden global information in the entire content.
- Thus, we have to look for a method to provide a way of untangling the non-linear connections between content and titles for the further prediction.

# Understanding of the Problem Statement

**Proposed Methodology:**

- We propose a deep attentive similarity model for capturing the discriminative information from local and global similarities. This way, we provide a way of untangling the non-linear connections between content and titles for the further prediction.

- Thus, 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 latent representations of textual inputs.

- Concluding, we then evaluate our framework on two benchmark datasets of clickbait detection - Clickbait Challenge Dataset and FNC dataset.

# Theory

1. **Preprocessing**: of the Headline, Article Body and converting text to WordtoVec.
2. **Learning Representation:** of Headline and Article Body through Bidirectional GRU.
3. **Learning Similarity:** applying Deep Semantic Similarity Model on Headline and Article body. Then calculating the similarity between the Headline and Article body.
4. **Prediction** : now applying softmax function to predict Clickbait or not.

# Preprocessing

- We first preprocess the texts by removing the stop words and lemmatization. The processed Clickbait Challenge dataset has an average of 10 words in the titles and 50 words in the bodies, while the FNC dataset has an average of 8 and 200 words accordingly.
- For the clickbait dataset, among the 19538 samples, we keep 14k for training and reserve 5k for testing. For better accuracy, we have used glove embeddings in one of the models to get a better vocabulary for training our model.
- We further vectorize the data using word-embedding techniques , which can be conducted unsupervised. Given that some titles only contain one word and this word is unique among the corpus, we train the word vectors with the “Min Word Count” set to 1.

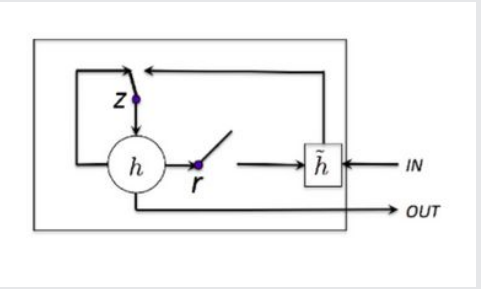
# Components

- wordToVec

  The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector. It capture their meanings, semantic relationships.

- Bi GRU

  A Bidirectional GRU, or BIGRU, is a sequence processing model that consists of two GRUs, one taking the input in a forward direction, and the other in a backward direction.
  It is a bidirectional recurrent neural network with the only input and forget gates.




- Cosine Similarity

  Cosine similarity measures the **similarity** between two vectors of an inner product space. It is measured by the **cosine** of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document **similarity** in text analysis.

# Model description

We have used two embedding layers as a stand-in for the heading and the body of our clickbait data representation.

We pass the embedding layers next to the input layer, where we use bidirectional GRU for our model of 300 words, with 250 units.

We use relu activation function in the dense layer for similarity calculation , which is passed on to cosine similarity for getting the global similarity.

Now to predict Its clickbait or not we pass it on to a softmax layer for getting the probability n range 0 -1.

Since our primary purpose is a yes/no answer for clickbait detection, we apply binary_crossentropy and use Adam/RMSprop optimiser to get the final model.

# Dataset Description

Two datasets will be used for our model :

- **Clickbait Challenge-**
  The dataset contains over 20,000 labelled pairs of posts for training and validation. The dataset is for the clickbait detection that was released in 2017.
  - There are five judges, each giving a clickbait score (from 0 to 1) to label the post. And a higher score stands for the higher probability of a post being clickbait.
  - All posts were annotated on a 4-point scale [not clickbaiting (0.0), slightly clickbaiting (0.33), considerably clickbaiting (0.66), heavily clickbaiting (1.0)] by five annotators from Amazon Mechanical Turk.
  - However, we'll regard the post with the mean score over 0.5 as clickbait. In the json data, we are mostly concerned with the targetTitle and targetParagraphs.

# Dataset Description

- **FNC dataset -**
  It is taken from the Fake News Challenge in 2017.
  It comprises of the labels-

  1. **Agrees**: The body text agrees with the headline.
  2. **Disagrees**: The body text disagrees with the headline.
  3. **Discusses**: The body text discuss the same topic as the headline, but does not take a position
  4. **Unrelated**: The body text discusses a different topic than the headline

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.

# Model (For FNC, using gensim)

```
embed1=Input(shape=(25,))
embed2=Input(shape=(1500,))

embedding_layer1 = Embedding( num_words1,300,embeddings_initializer=Constant(embed_matrix1),
                             input_length=max_length1,trainable=False)(embed1)
embedding_layer2 = Embedding( num_words2,300,embeddings_initializer=Constant(embed_matrix2),
                             input_length=max_length2,trainable=False)(embed2)
bigrulayer1=Bidirectional(GRU(units=250, dropout=0.3, recurrent_dropout=0.5))(embedding_layer1)
bigrulayer2=Bidirectional(GRU(units=250, dropout=0.3, recurrent_dropout=0.5))(embedding_layer2)

pin_part = Dense(64, activation='relu')(bigrulayer1)
sku_part = Dense(64, activation='relu')(bigrulayer2)

global_similarity=keras.layers.Dot(axes=1, normalize=True)([pin_part, sku_part])

probb= Dense(1, activation='softmax')(global_similarity)

model1 = Model(inputs=[embed1] + [embed2], outputs=probb)
model1.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])
model1.summary()
```

# Secondary Model (For Clickbait, using glove)

```
model = Sequential()

model.add(Embedding(input_dim = max_features, output_dim = EmbeddingSize, weights = [embedding_matrix], input_length = maxlen ,
                    trainable = False))
model.add(Dropout(dropout_embedding))
model.add(Bidirectional(GRU(512, dropout=0.2, recurrent_dropout=0.5)))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='mse', optimizer='rmsprop')

batch_size = 64

earlystop_cb = keras.callbacks.EarlyStopping(monitor='mse', patience=7, verbose=1, mode='auto')

model.fit(X_train, y_train, batch_size=batch_size, epochs=20, validation_split=0.1, callbacks=[earlystop_cb])
```

# Experimental Results

| DataSet | Experiments | Test Accuracy |
|---|---|---|
| FNC | Optimizer - Adam<br>Activation - relu | 83.47% |
| FNC | Optimizer - RMSprop<br>Activation - Sigmoid | 83.46% |
| FNC | Optimizer - Adam<br>Activation - Sigmoid<br>Epochs(trained on) - 5 | 83.57% |
| Clickbait Challenge | Without Glove, RMSprop | 85% |
| Clickbait Challenge | With Glove, RMSprop (Train) | 77% |

# Results

**Final train and test accuracy on FNC dataset**

```
model1.fit(inp_list, out_list.astype("float64"), batch_size = 64,epochs=2)
```

```
Epoch 1/2
235/235 [==============================] - 7324s 31s/step - loss: 0.6869 - accuracy: 0.7448
Epoch 2/2
235/235 [==============================] - 7342s 31s/step - loss: 0.6382 - accuracy: 0.7442
<tensorflow.python.keras.callbacks.History at 0x7f4b793f2d90>
```

```
score = model1.evaluate([tensort1,tensort2], out_list, verbose=0)
print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')
```

```
Test loss: 0.5790790319442749 / Test accuracy: 0.8346999883651733
```

# Results:

## (For clickbait challenge)

### Primary model:

Train:

```
Epoch 1/2
229/229 [==============================] - 2291s 10s/step - loss: 0.6813 - accuracy: 0.7329
Epoch 2/2
229/229 [==============================] - 2272s 10s/step - loss: 0.6303 - accuracy: 0.7588
<tensorflow.python.keras.callbacks.History at 0x7fc1551b6490>
```

### Secondary Model: (With glove)

```
Mean Squared Error = 0.031406514
accuracy = 0.8553777597731416
precision_score = 0.745158002038736
recall_score = 0.6117154811715482
f1_score = 0.671875
```

# Individual Contributions

| Task | Contribution: |
| --- | --- |
| Parul | Cleaning and preprocessing of data |
| Shweta | Learning the similarity and training the model |
| Anshul | Predicting, Testing and Comparing different models |

# Thank You!