BanglaBait: Using Transformers, Neural Networks & Statistical Classifiers to detect clickbaits in New Bangla Clickbait Dataset

by

Motahar Mahtab
18301023
Monirul Haque
18301055
Mehedi Hasan
18301052

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2022

Declaration

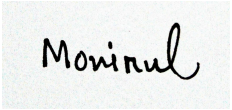
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

MD. Motahar Mahtab
18301023

Mehedi Hasan
18301052



Monirul Haque
18301055

Approval

The thesis titled "BanglaBait: Using Transformers, Neural Networks Statistical Classifiers to detect clickbaits in NewBangla Clickbait Dataset" submitted by

1. MD. Motahar Mahtab (18301023)
2. Mehedi Hasan (18301052)
3. Monirul Haque (18301055)

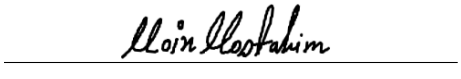
Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 16, 2021.

Examining Committee:

Supervisor:
(Member)

Mujtahid Al-Islam Akon
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-supervisor:
(Member)



Moin Mostakim
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

The art of luring us to click on certain content by exploiting our curiosity is recognized as clickbait. Clickbait might be aggravating at times because it is misleading. Several studies have worked on the detection of clickbait in online platforms as we transition from the Information Age to the Age of AI. Nonetheless, predicting clickbait in Bengali new articles is still a work in progress. Here, we use deep learning, the process of extracting pattern or feature from data using neural networks, to determine whether an online Bengali article is clickbait or not. We scrape data from online Bengali news articles, manually annotate them and employ deep neural network architectures like CNN, Bi-LSTM,Bi-GRU and pre-trained fine-tuning language representation approaches –i.e. BERT, BanglaBERT, M-BERT to provide inputs for various types of classifiers. Finally, we evaluate the classifiers' outputs and choose the best outcome to predict clickbait in Bengali news articles.

Keywords: Clickbait; Deep Learning; Bengali; Online News; Prediction; Binary Classification; BERT;

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Mr. Mujtahid Al-Islam Akon sir and co-supervisor Mr. Moin Mostakim sir for their kind support and advice in our work. They helped us whenever we needed help.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$BERT$ Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

DT Decision Tree

GNB Gaussian Naive Bayes

GRU Gated Recurrent Units

KNN K-nearest Neighbour

LR Logistic Regression

$LSTM$ Long short-term Memory Network

POS Part of Speech

RF Random Forest

SVM Support Machine Vector

Chapter 1

Introduction

Due to the widespread usage of the internet over the last two decades, the news industry has progressively evolved into an online news industry. The business paradigm of today's internet news sector differs from that of print media. While headline writing has traditionally been regarded as a skill, a new term has emerged to describe online journalism in the internet world: clickbait. The explosion of clickbait titles in recent years has not gone unnoticed. According to Murtha [1], for three posts a week, Slant (Q&A site for product recommendation) offers its writers $5 for every 500 clicks along with $100 per month. In this regard, Slant is not alone to follow this economic strategy, rather it is becoming more widespread. It would be a mistake to believe that Bengali news publishers are far behind in this competition. Although most reputable Bengali newspapers, TV media and online news portals offer a subscription system, the majority of people are likely to read news online without paying a subscription cost. As a consequence, advertisements continue to be one of the most important sources of revenue for news providers. Now, the more people that click and visit their website, the more traffic they receive, resulting in an increased ads revenue. Therefore, they create enticing and intriguing headlines that include exaggerated information rather than actual content to get readers to click titles. Visitors appear to be disappointed because the news does not provide the information promised in the headline. The phrase "clickbait" is frequently used as a derogatory phrase; however, the reality is more complicated. Moreover, the concept of clickbait can be a little hazy to grasp. Thus, the classification of clickbait is a highly subjective endeavour. In a research paper, Biyani *et al.* [2] utilized article informality to conduct a study on detecting clickbait in news streams. The study defines and categorizes eight different types of clickbait. Table 1.1 displays several examples of clickbait news titles and their categories.

To find clickbait headlines, we created automation scripts to collect raw data from online Bengali newspaper sites, TV news sites, online news portals and manually labelled the dataset into two categories - clickbait and non-clickbait. After that, we used pretrained word embeddings such as fastText, Glove, bnwiki, self-trained word embedding and n-gram word & character techniques to extract features from our dataset. To classify news we employed statistical classifier models such as Logistic Regression, Decision Tree, Random Forest, Gaussian Naive Bayes, SVM, deep neural network architectures such as CNN, Bi-LSTM, Bi-GRU and transformer models like

BERT[3], RoBERTa[4], Distilbert[5], ELECTRA[6], XLM-RoBERTa[7] etc to find the best performing model for detecting clickbaits.

Category	Reason	Title
Wrong	Factually wrong headline.	মাত্র ১ রাতে পা ফাটা ও পায়ের গোড়ালি ফাটা থেকে মুক্তি!
Teasing	Title creates a suspense by teasing.	২৩ বছরে ১১ শিশুর মা, নিতে চান ১০০ সন্তান!
Formatting	Overuse of punctuation mark.	বুবলীর সন্তানের বাবা কে? জেনে নিন এক্ষুনি!
Inflammatory	Title presents inappropriate words.	চাচির গোসলের ছবি তুলে ব্ল্যাকমেইল করে ভাতিজার নিয়মিত শারীরিক সম্পর্ক!
Inflammatory	Indication of vulgar title.	প্রেম পরবর্তী মেডিকেল ছাত্রীর কাছে অন্য কিছু চাওয়ায় পুলিশি অ্যাকশন!
Ambiguous	Title without context.	আজ বিস্কুট খেয়েছেন তো?
Exaggeration	Exaggerating title.	জামের সঙ্গে যে তিন খাবার খেলে হতে পারে মৃত্যুও
Ambiguous	Confusing or unclear title.	'আন্দোলন, আন্দোলন, আন্দোলন'

Table 1.1: Clickbait news titles and their categories.

- We release an annotated dataset of ≈ 14k Banlgla articles for clickbait detection which will propel future researches on it.
- We investigate different statistical models, deep neural networks and transformer models like BERT [3], RoBERTa [4], Distilbert [5], ELECTRA [6], XLM-RoBERTa [7] etc to find the best performing model for clickbait detection.

Following is a breakdown of the paper's structure - i.Following section lists various similar works/literature review for identifying clickbait headlines that are currently available. ii.The problem statement is clearly stated in Section 3. iii.Data collection, processing, and analysis are discussed in Section 4. iv.The proposed mechanism for detecting clickbait articles is presented in Section 5. v.Section 6 describes the preprocessing and experimental setup. vi. Section 7 analyzes the results vii. Section 8 draws the conclusion and contains some proposals about future research work

1.1 Aims and Objectives

There is an estimated 11.4 million internet users in Bangladesh[1] who receive their daily news mostly fromonline news portals. But there has been no researchconducted on how to tackle the increasing amount ofclickbaits that appear on these portals and other newswebsite. The goal of this paper is to design an efficient model that can determine whether or not a Bengali online news title is clickbait. Although there are several researches on clickbait, most of them are done on languages other

[1]https://www.cia.gov/the-world-factbook/countries/bangladesh/

than Bengali. Thus, our core objective is to train and tune multiple models to see what works best for detecting clickbaits. To accomplish the objective, we aim to appropriately preprocess and represent the raw data so that they can be validated using deep learning classification methods. Besides, comprehending the various attributes is also required in order to achieve greater precision in the results. After we complete all of our objectives, we can construct a good predictive model.

Chapter 2

Related Work

Automatic classification of clickbaits in Bengali language is essentially important to limit the astounding growth of clickbait contents in Bengali news sources. The origin of clickbait is rooted in tabloids which have been in journalism since 1980's [8]. During 2015-2017, the first substantial researches on clickbait identification were conducted which relied on handcrafted linguistic features to train different classifiers like SVM, Gradient Boosted Decision Trees, Naive Bayes etc [2], [9]–[12]. For example, an in depth analysis of semantic features such as unresolved pronouns, effective language & action words, climactic language, and excessive use of numerals, syntax features such as forward reference and reverse narrative, and image features such as image placement and emotional content in news articles were conducted by researchers in [9] but did not provide automatic classification. Both researchers Chen *et al.* [9] and Zheng *et al.* [10] incorporated user behaviour analysis as a feature but only Zheng *et al.* [10] provided automatic classification via Gradient Boosted Decision Trees (GBDT).

Clickbait detecting features can be obtained from 3 different origins: the clickbait teaser phrase or the post text, the attached article that post text wants the user to click, and metadata for both [13]. Apart from the post text which is used by most to identify clickbait, Potthast *et al.* [12] and Biyani *et al.* [2] also considered the linked article and the metadata. Potthast *et al.* [12] aimed to detect Twitter clickbaits using bag-of-words, image tags, sentiment polarity, readability, length, contractions and punctuation use of a tweet. They also included several metadata features like whether a tweet contains media, gender of the user etc. They compared different classifiers like Logistic Regression, Random Forest, Naive Bayes and the Random Forest classifier proved to be the most effective among all.

Biyani *et al.* [2] used textual similarity between TF-IDF characteristics of headline and the top five sentences of the article, informality features (amount of informal language used in article), forward reference features, URL features of the web page. To categorize clickbaits according to their characteristics, they developed a Gradient Boosted Decision Trees (GBDT).

However, handcrafted feature dependent models are not useful when it comes to non-English settings as the linguistic parameters would be changed. Furthermore, engineering these features require much time, domain-specific knowledge and corpus-

specific features like Twitter metadata will also become unusable when working on a different corpus. Nonetheless, deep learning algorithms can reduce the need for feature engineering by identifying relevant characteristics from unstructured text data automatically, Collobert *et al.* [14] suggests.

Potthast *et al.* [15] suggested that instead of a binary classification challenge, clickbait detection should be a regression problem as the former provides a way to actually measure how much clickbait is in the teaser message. They built the first large scale annotated clickbait corpus (Webis Clickbait Corpus 2017) and there were 338,517 tweets from 27 major news organizations in the United States. Following this, the Webis clickbait challenge 2017 was formed by them to boost research activity in clickbait detection giving rise to some highly effective and flexible deep learning techniques. In this challenge, the teams had to construct a regressor that calculates the degree of clickbait in a post on twitter from the Webis Clickbait Corpus 2017.

For clickbait challenge 2017, Zhou [16] first used self-attentive RNN to select the important words for clickbait detection and created a bi-GRU network to encode the contextual information. Glenski *et al.* [17] proposed another model for clickbait challenge 2017 where it was hypothesized that additional performance gain can be achieved by incorporating tweet text, image and linked article content. They used linguistically infused neural network architectures like Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) that include a max-pooling layer, two 1-dimensional convolution layers and an embedding layer. Similar to the idea of Glenski *et al.* [17], Thomas [18] also incorporated article content to an LSTM model for the clickbait challenge. For text preprocessing, they used a simple whitespace tokenizer and converted them word embeddings, which are inserted into the LSTM unit. They used batch normalization and dropout between the individual neural network layer for their model.

For the same challenge, Omidvar *et al.* [19] experimented on "postText", "targetDescription", and "targetTitle" of twitter corpus and found that evaluation on only postText gives the best result. They compared bidirectional simple recurrent units [20], bidirectional GRU, and bi-directional LSTM and found bi-GRU as the most effective design for detecting clickbait. For word embeddings, they used 50, 100, 200, 300 dimensional GloVe word embeddings [21]. To reduce overfitting, they employed Drop out technique for forward GRU, Embedding and backward GRU layers and used mini batched gradient descent of size 64 for training.

In a research paper Anand *et al.* [22] created a browser add-on that could inform users about the likelihood of a headline being a clickbait or not in various media sites. To build their model, they used LSTM and GRU models with word and character embeddings as feature inputs. Although, they only considered the headings, they classified clickbaits with an F1 score of 98 percent.

Rony *et al.* [23] used continuous skip-gram model [24] to learn the generate the word embeddings of clickbait title. The average of the word embeddings are used to create the concealed depiction of every sentence. A linear classifier is trained using

these sentence depictions specified in [25].

Indurthi *et al.* [11] first inquired the application of transformer regression models in clickbait detection and achieved first position in clickbait challenge. They used ELMo [26], Universal Sentence Encoder [27], transformer encoders like BERT [3], RoBERTa [4] and OpenAI's GPT2 [28] for word and sentence embedding representations. Then they insert these vector representations to regression models like Linear Regression, Ridge Regression, Gradient Boosted Regression, Random Forest Regression, Adaboost Regression and choose the best model using MAE (mean squared error) as scoring metric.

Wu *et al.* [29] used four types of weighted summation to classify clickbaits. They used Transformers to assign a score to clickbait headline and body according to their representations. Then they used a co-attention network to compute the contextual similarity between headline and article body and assigned a title-body matching score. They also computed a title stylistic score learned from a character-level transformer.

Hossain *et al.* [30] created the first Bengali newspaper dataset for Bengali fake news detection containing an annotated dataset of 50K Bangla news. But there hasn't been any research to our knowledge to tackle the ever-increasing clickbaits in our news outlets and media.

Chapter 3

Problem Statement

In this study, we define clickbait detection as a supervised binary classification problem where the set of categories, $\mathcal{C} = \{clickbait, non\_clickbait\}$. Given a set of article titles T = $\{t_1, t_2, t_3, ..., t_N\}$, and their bodies B = $\{b_1, b_2, b_3, ..., b_N\}$, our goal is to predict a label Y = $\{y_1, y_2, y_3, ..., y_N\}$ for these articles where $y_i = 1$, if title i is clickbait and $y_i = 0$, if it is not clickbait. In order to represent the correspondent (head/title, body/content) pairings, we utilize a collection of tuples P = $\{t_1, b_1), (t_2, b_2), (t_3, b_3), ..., (t_N, b_N)\}$ and in case of only title, P = $\{(t_1, t_2, t_3, ..., t_N\}$. Our model includes three main segments- i. Statistical models taking linguistic features,ii. Word embeddings passed to Deep Neural Network Models,iii. Transformer Networks

Chapter 4

New Dataset for Detecting Bengali Clickbait News

4.1 Data Collection

As there was no previous research on Clickbait Detection in Bengali News, there was no publicly available dataset. The existing news datasets like Patrika[1] which consists of 320,000 news articles contain data from popular news portals like Jugantor, Jaijaidin, Ittefaq, Kaler Kontho etc. which do not contain a lot of clickbaits. Their dataset was mainly created to facilitate automatic headline classification of articles. That is why we had to create a completely new dataset for detecting Bengali Clickbait News. As we wanted to create our dataset using latest data, we scrapped news articles from web. So, to perform web scraping operations we utilised the Python Selenium module in creating automation scripts separately for each website to invoke the chrome webdriver. We scraped notorious websites that frequently use clickbait headlines. In order to incorporate news that most people daily encounter, we also incorporated news from the most popular Bengali news portal websites. This will also provide future researchers with the means to investigate clickbait practices in these popular news medium. We collected news headlines along with the main news contents and dates. Table 4.1 represents the list of websites we have scraped from and the number of news we have scraped from each site to create the dataset for our research. After removing duplicate news we got 11,826 unique news headlines along with content in our dataset. All our news articles were from scrapped in date 1 to date 2 time period.

We also took advantage of the already available Banfake Dataset [30], consisting of 48,000 authentic news fetched from popular, reliable Bangla online news portals. As stated in that paper, the articles that were scraped for the authentic news were pulled from the top 22 online news portals of Bangladesh according to Alexa ranking. Besides, the 2,000 fake news that were added in the dataset includes misleading, clickbait and satire news. Although these articles are labelled, there were overlaps among satire, clickbaits and misleading lables. So, we manually labelled these articles into clickbait and non clickbait to maintain the consistency of labelling throughout articles. After adding all the clickbait news from that dataset, our dataset has grown with total 13,460 unique news. Lastly, we constructed our corpus from the

[1] https://data.mendeley.com/datasets/v362rp78dc/2

News Portal	Web Address	No. of News
Somoy TV Online	www.somoynews.tv	436
RTV News Online	www.rtvonline.com	1,090
Newzcitizen	www.newzcitizen.com	986
Topdhaka	www.topdhaka.com	1,930
CityNewsZet	www.citynewszet.com	986
twentyfourbd	www.twentyfourbd.com	2,801
AuthorityNewz	www.authoritynewz.com	971
nbtimes24	www.nbtimes24.com	990
NewzAuthority	www.newzauthority.com	613
TheBaseNewz	www.thebasenewz.com	2,153
Kaler Kantho	www.kalerkantho.com	321
NewsHolder21	www.kalerkantho.com	479
	Total	11,826

Table 4.1: List of websites with the number of news that have been scraped.

dataset we assembled. We intend to make our dataset publically available when we finish collecting 40k news articles for any future research on Bengali Clickbait News Detection.

4.2 Annotation Process

As we wanted to run supervised machine learning algorithms, the data corpus we constructed needed to be annotated. So, we added an extra column in our data corpus and manually labelled all the data with either "clickbait" or "non-clickbait" following the categories of clickbait shown by Biyani *et al.* [2]. We marked clickbaits as numeric value 1 and non-clickbaits as numeric value 0. The process of annotating is one of the most difficult tasks of this research as there is a lot of gray area when to comes to distinguish between clickbait and non clickbaits. Also it is one of the most important tasks of this project as annotating normal news titles as clickbaits may result in more false positives which we are trying to avoid. All the annotations were done by ourselves where we followed a strict criteria to annotate titles as clickbaits. Most common clickbaits titles that create a knowledge gap that entice readers to click and see their content. But this is a broad category and many normal titles may contain some amount of knowledge gap in them as all titles basically want to attract reader's attention. In order to differentiate between the two, we judge whether the held out information could be easily associated inside title to determine whether knowledge was held out deliberately. For example, **'আমি ধানমন্ডিতে, 'আব্বু-আম্মু পুরান ঢাকায়'** creates a curiosity inside reader's mind about who quote this title. As the name of the person who quote this title could be easily included within the title, knowledge was deliberately withheld and this is a clickbait. Second form of clickbaits exaggerate their titles and their content under delivers. For example,**সালমানের স্ত্রী হচ্ছেন সোনাক্ষীই!** title's content says that the addressed event is only in a movie, not in real life. These type of titles are time consuming to detect as the content has to be addressed to judge the title's validity. The main difference between these type of clickbaits and fake news is that if the title and content convey

the same information, even if it is incorrect or totally fake, it is non clickbait. But if the title and content differs in their messaging, then it is taken as clickbait. But there are many other types of clickbaits that may not fall under these two broad categories and thus consistency among annotations is important. As we set out to separately label our dataset, our inter annotator agreement should be satisfactory. At first, each of us labelled 200 news articles where we received a Fleiss Kappa Interrater Agreement score of 0.472 which is moderate. For exmples, Cilckbait Challenge Dataset [15] which is the benchmark dataset for clickbait detection in English articles had an inter annotator Fleiss Kappa score of 0.36. Clickbait titles are most frequent when in Entertainment type titles. If the dataset is skewed to a specific type of news, models will face problems to detect clickbaits from low prevalent classes. Based on the current headline distirbution of articles, we decide which type (Entertainment, National, Sports etc) of articles to collect. As headlines between sites differ, we employ an automatic headline classification model which is similar to the Bangla News Headlines Categorization model[2] except bi-LSTM was used instead of GRU network. Furthermore, we trained our model on a Patrika dataset[3] which contains 320,000 news articles from 8 categories each containing 40,000 articles. Our model achieved 95% accuracy on this dataset which we used to automatically create headline category for our news titles. After finishing the annotation process we have got 8813 non-clickbait and 4647 clickbait news. Table 4.2 illustrates number of news in each category in our dataset.

Category	News count
Entertainment	5112
National	2283
International	2246
Sports	1504
Education	767
Science & Technology	702
Economy	606
Politics	240

Table 4.2: Number of news in each category in dataset

4.3 Exploratory Data analysis

Clickbait titles are mischievous in nature and most of the time hide critical information, present misleading information and exaggerate news with their raunchy titles. We aim to analyze whether any stylistic distinction can be made between the clickbait and non clickbait in terms of title length,content length,frequency of punctuation and title-content similarity. If title length is a trait for distinguishing between clickbait and non clickbait titles, then their distribution will look different. The same can be said about content/body length. Furthermore, as there is a tendency of using more punctuation marks such as '!','?' and digits in clickbait titles, it can also be a contributing factor.

[2]https://github.com/eftekhar-hossain/Bangla-News-Headlines-Categorization
[3]https://data.mendeley.com/datasets/v362rp78dc/2

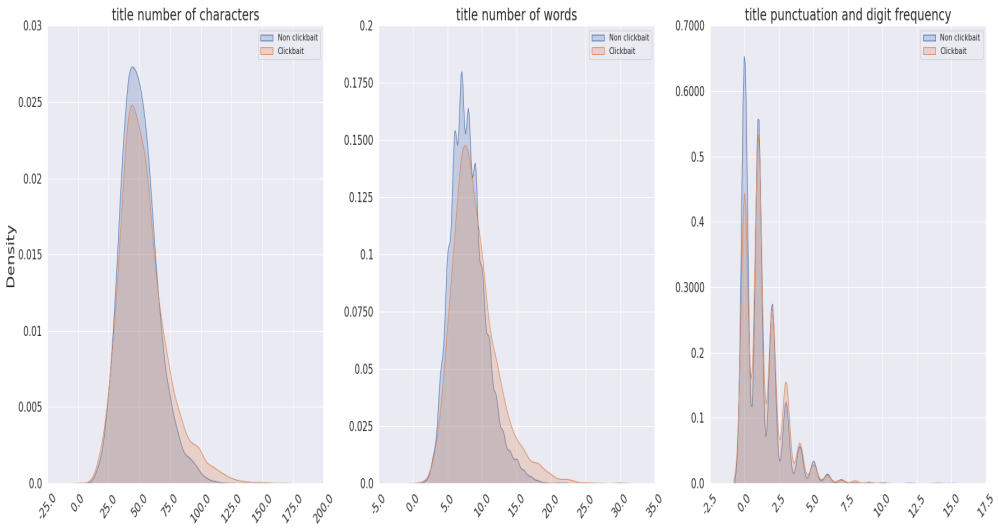


Figure 4.3.1: KDE of title length and punctuation, digit frequency

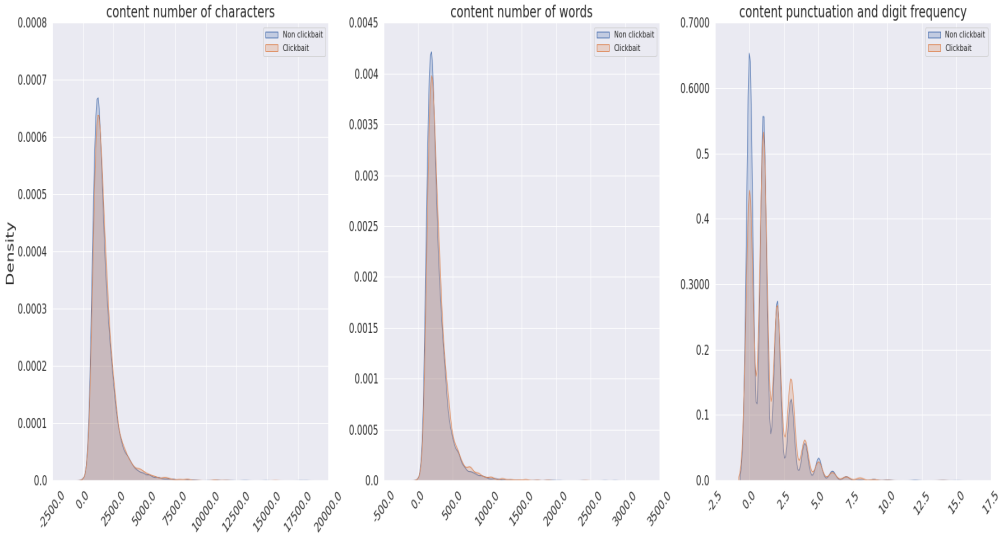


Figure 4.3.2: KDE of content length and punctuation, digit frequency

We create KDEs of title character, word usage and punctuation,digit usage to analyze whether the distributions look different. From Figure 4.3.2, we can see that clickbait titles tend to be slightly longer than non clickbaits as they have more characters and words than non clickbaits. In case of punctuation and digit, non clickbait titles have less punctuation than clickbaits. Figure 4.3.2 shows that content character, word and punctuation usage do not differ much from non clickbait titles. Thus, it is evident that title is statistically more significant than content for detecting clickbait news.
Clickbait titles often exaggerate the details in their title while their actual contents under deliver. So, similarity between the title and content might be an indicator of clickbaits. Word embeddings are dense representation of texts that contain semantic and linguistic properties of words.Different types of word embedding used by us is described in section 5.2. Each title and content instance has to be converted into their own vector representation to compute the cosine similarity between them. We use Doc2Vec [31] algorithm to create embeddings for title and content. We train a Doc2Vec model via the gensim [32] framework on the whole dataset. After training we have four document vector arrays: clickbait title, clickbait content, non clickbait title and non clickbait content in both models each containing a single embedding for each sequence. The cosine similarity of title-content vectors of both clickbaits and non clickbaits are plotted into a kernel density map to figure out the differences in their distribution.

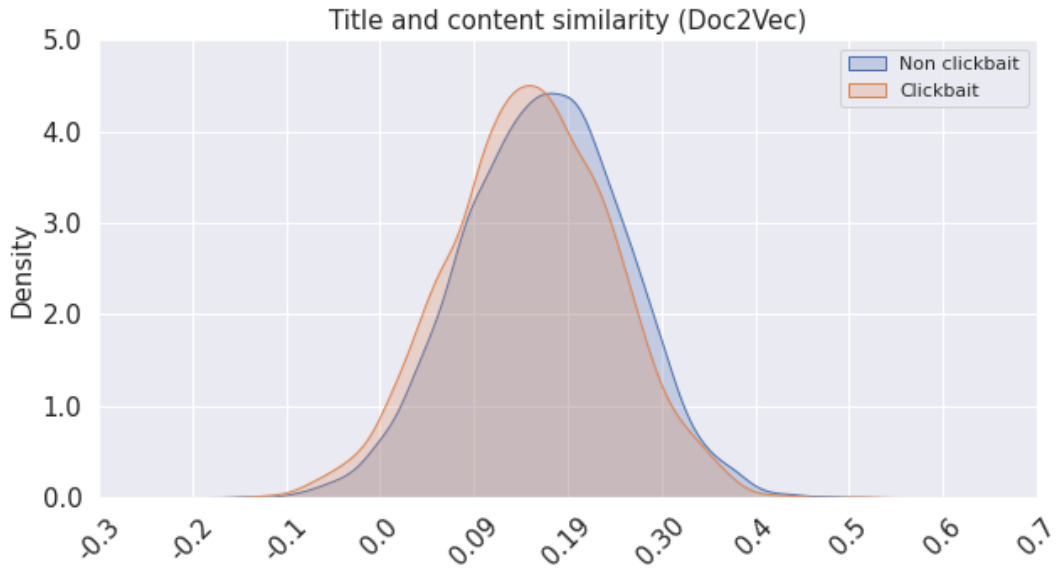


Figure 4.3.3: comparison of title and body similarity between clickbait and non clickbait

From that figure4.3.3, we can see that similarity scores are slightly higher in non clickbait titles. That is why we can conclude that clickbait titles appear to be more disjoint from their content than non clickbaits. That is why we can conclude that experiments incorporating content should be conducted to detect clickbaits.

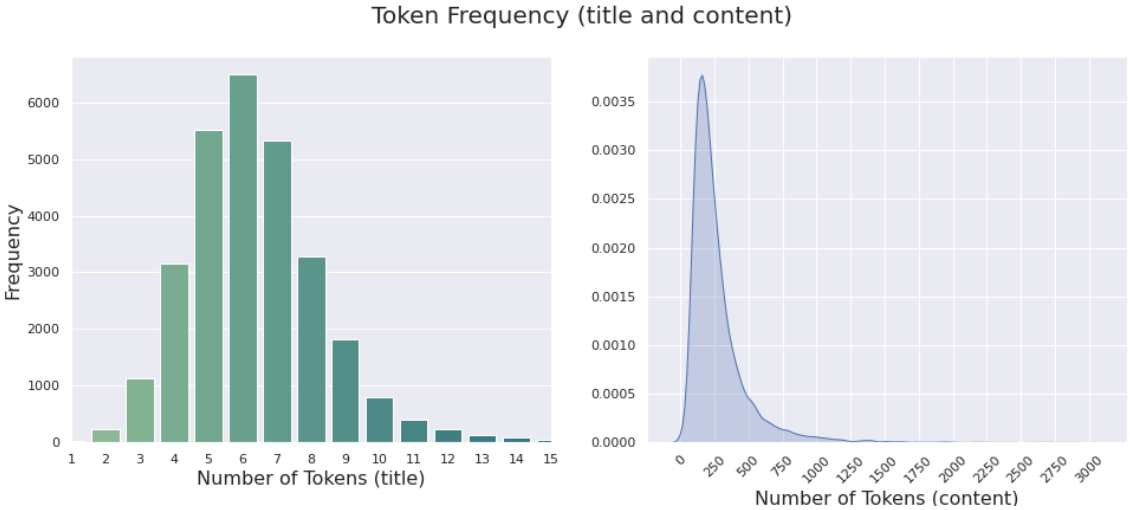


Figure 4.3.4: Token frequency of title and content (training data)

Chapter 5

Approach

5.1 Linguistic Features

As we saw that clickbait and non clickbait titles have semantic and syntactic differences, we collect different semantic and lexical features from the title and content. Firstly, we experiment with character n-grams of size 3-5 and word n-grams of size 1-3 with weighting scheme term frequency-inverse document frequency (TFIDF). Previously, we saw in our exploratory data analysis part in section 4.3 that clickbaits have a slightly higher punctuation and digit frequency. We include the frequency of punctuation and digits as a feature. We also noticed that clickbait websites are often unpopular, unheard websites that stand low in alexa ranking. So, we incorporate the normalized value of alexa ranking of the domain as a feature in our dataset. In order to capture useful syntactic information, we incorporate parts of speech tag (POS) as feature.We use the POS tagging tool created by Sarker [33]- a pretrained conditional random fields (CRF) model trained on nltr [34] dataset of 2997 sentences. We used the normalized frequency of different parts of speech tags like as a feature for each document. We also extract the named entities from our dataset using the same bnlp toolkit and use the normalized frequency as feature.

5.2 Word embedding

Word embedding is a word representation technique that allows similar meaning words to have similar representations. They transform words to dense word vectors containing tens or hundreds of dimensions. Inside these vectors, the semantic and linguistic properties of words are contained. Various ways of generating word embeddings are: Word2Vec [24], Neural Probabilistic Language Model [35],GloVe [21], LexVec [36], Fasttext [37], dependency-based embeddings (DepVec) [38], ELMo [26] etc. We use distributed word and sub-word embeddings to capture the semantic meaning within our text.
Word2Vec generates word embeddings via pre-training tasks like skip gram and continuous bag-of-words approach (CBOW). Stanford NLP researchers, Pennington *et al.* [21] found that by calculating the singular value decomposition (SVD) of word-word co-occurance matrix, they can generate the same weight matrices of Word2Vec. As they considered global co-occurances of words, they named it Global vectors for word representation or GloVe.In GloVe vector space, distance between two word vectors equal to the ratio of the logarithm of their co-occurrence prob-

ability. Furthermore, Word2Vec uses back propagation which is time consuming whereas GloVe is guaranteed to produce the best optimized word vectors in less time with smaller corpus. Researchers in Facebook [37] found that sub-words can mitigate the problem of not finding suitable embedding for rare words. In this new approach, each word is broken into character trigrams and the skip-gram model predicts the context subwords instead of whole words.
We use pre-trained Bengali Word2Vec, fastText, GloVe word embeddings created by Sarker [33]. Their Word2Vec and fastText embedding is trained on Bangla Wikipedia dump with embedding size of 100 and 300 respectively. The GloVe embedding is trained on wikipedia and crawled newspaper articles and have an embedding size of 300. We had 53.8 percent coverage on Word2Vec, 55.8 percent coverage on fastText and 54.1 percent coverage on GloVe.

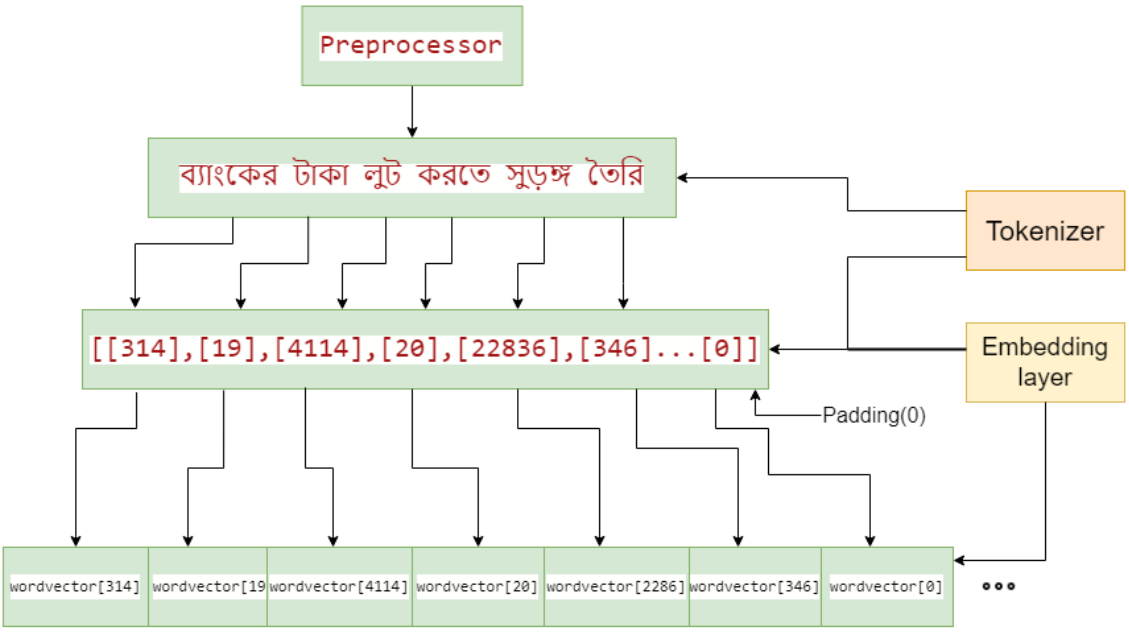
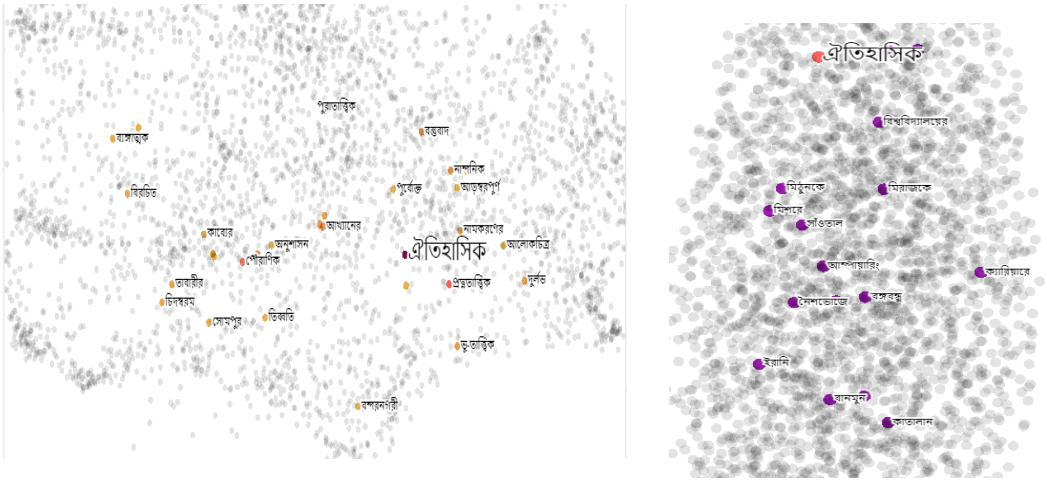


Figure 5.2.1: How word embeddings are passed into model

Figure 5.2.1 shows the process of how input sequence is converted to word embeddings and passed to neural network models. We let the vocabulary size of our corpus=$|V|$ and embedding vector size=emb and create an embedding matrix $W^{|V|emb}$. Every word w^i has a corresponding word vector wv^i and every entry of word matrix is mapped to the word vector following this equation: $wv^i = W^{v^i emb}$. This mapping is done via Embedding layer, the first layer of all our neural network models. The Embedding layer takes an embedding matrix as weights which has the word embeddings for all the indexes. All the input sequences have fixed token size which is achieved via padding and truncating depending on input size. We experiment with either keeping these pre-trained word embeddings fixed or train with with the network via the Embedding layer.
Apart from these pre trained word embeddings, we also train our own embeddings from scratch via the gensim framework [32]. To train our own embeddings, we preprocess the text according to the preprocessing section 6.1 and tokenize the preprocessed texts. We can compare the word embeddings from the figure 5.2.2. The figure shows that word2vec embeddings carry more generalized meanings than self



(a) Word embedding of **ঐতিহাসিক** (word2vec)(b) Word embedding of **ঐতিহাসিক** (self trained)

Figure 5.2.2: Comparison of word embeddings

trained embeddings as word **ঐতিহাসিক** has nearest neighbours like **পুরাতাত্ত্বিক**. But self trained embeddings are more datast biased as **ঐতিহাসিক** has nearest neighbours: **মিশরে** and **ইরানি** where the relation seems less obvious than word2vec embeddings ans seems location based. Gathering more data may lead to better generalization of our word embeddings. Table 5.1 lists out all the features that we extracted from our dataset after preprocessing.

5.3 Statistical Classifier Models

The features in listed in table 5.1 are passed onto various statistical classifier models like Logistic Regression binary classifier, Random Forest classifier, Decision Tree classifier, K-nearest Neighbors, Support vector machines (SVM) with linear and Radial Basis Function (RBF) kernel and Gaussian Naive Bayes. These classifier models require several hyper-parameters according to different kind of problems and datasets. So, Grid Search algorithm is used to identify best parameters for each type of features and are validated using Stratified K-fold cross validation as our dataset do not have equal amount of clickbait and non-clickbait news. After that full train set is passed to the statistical classifier models to predict test set and measure metrics and scores ultimately.

5.4 Neural Network Models

Given enough data, neural network models can outperform generic regression models without the need of handcrafted linguistic features. These models perform excellently in variety of text classification, generation and translation tasks. We primarily experiment with CNN, Bi-LSTM [39], Bi-GRU models where the word embedding

Feature names (extracted from title and content)	Feature Description
Unigram(u)	TF-IDF of Word unigrams
Bigram(b)	TF-IDF of Word bigrams
Character trigram ($char_3$)	TF-IDF of character trigrams
Character 4-gram ($char_4$)	TF-IDF of character 4-gram
Character 5-gram ($char_5$)	TF-IDF of character 5-gram
Word embedding bnwiki(bw)	Embedding trained on bnwiki dump
Word embedding fasttext(ft)	Embedding trained on Bengali fastText_wiki
Word embedding GloVe(gv)	Embedding trained on Bengali GloVe Wordvectors
Self Trained embedding	Embedding trained on the dataset

Table 5.1: All features extracted from dataset

features from table 5.1 are passed to the models as described in Figure 5.2.1.

5.4.1 CNN

Convolutional Neural Networks have the ability to encode reliable features from short and long texts and thus used heavily in text classification. We create a multi-channel CNN model that is similar to the model demonstrated by Kim [40]. Multi-channel CNN models use different size kernels that convolve other the text vectors. So, this model can process documents at different levels of n-gram groups like unigram, bigram and tigram group. Then the model learn how to integrate these various n-gram combinations to generate the best features. Although Kim [40] experimented with two channels one having dynamically updating embeddings and one static or unchanged embeddings, we keep our word embeddings static throughout the experiment. We use kernel size of 1-4 with 256 hidden units for each kernel size and activation function ReLU [41].
After each convolution operation, we perform a maxpooling 1D operation [42] over the token sequence axis as shown in Figure 5.4.1. This maxpooling operation chooses the most important feature from each kernel output. These outputs from 4 different kernel sizes are concatenated and passed onto a dense layer of 512 units to extract rich features. Before the dense layer, we insert a Dropout layer which randomly drops out a percentage of the hidden units during forward propagation coming from the previous dense layer. This way our model does not become dependent on some combination of hidden units and is able to generalize better. Finally the probability of being clickbait is calculated with an one unit sigmoid activation layer. We will experiment with pre-trained CNN architectures to see if transfer learning helps us achieve better results than this model.

5.4.2 LSTM

One big limitation of CNN is that due to filter or kernel size, CNN can only learn dependencies between words within the kernel size. So, in long texts where outcome at the end of a sentence may depend on the starting of the sentence, CNN fails to

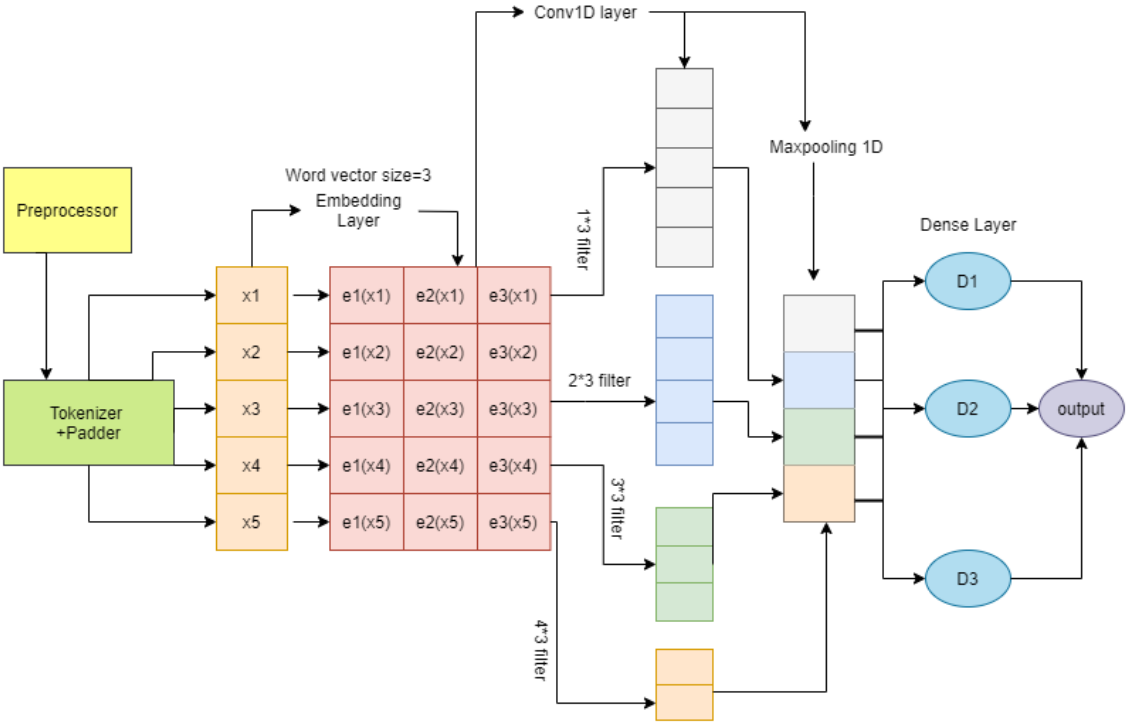


Figure 5.4.1: CNN architecture

capture this relationship. This is crucial in clickbait detection as many clickbait titles bait users by describing an incident and then keep users guessing about its outcome. In these cases, CNN might fail to learn this connection. LSTM has memory cells that help it preserve past information and learn encodings of sequential information efficiently. It has input gate, output gate and forget gate with each gate having specific functionality. Forget gate decides what to remember from past sequences or what to forget, input gate decides which portion of the current token's value should be added to current time state and output gate decides how much of current state should be passed to the hidden output. Bi-LSTM extends on LSTM and can encode both left to right and right to left dependencies. LSTM can encode how much past information can effect future state but cannot do the reverse whereas Bi-LSTM can encode both.

One caveat of LSTM is that even it can also be forgetful in very long sentences. LSTMs can be thought of as an encoder that creates an encoded representation of the whole input sequence into a fixed sized vector. But for very long sentences, it is harder to encode all important information of that long sentence into a fixed sized vector which may make LSTM forget about the past information in the sentence. According to Bahdanau *et al.* [43], attention mechanism allows a model to choose which part of the encoder hidden state is more important for correct classification. This prevents forgetfulness as the encoder does not have to produce an encoded vector that contains all the information because attention mechanism will help in choosing which part of the sequence is important. Figure 5.4.2 shows the architecture of our bi-LSTM model with attention head which is similar to the model demonstrated by Zhou *et al.* [44]. The only difference between our LSTM and GRU model is that the LSTM units are replaced with GRU. We stack two LSTM layers of hidden units size =256 and add a Dropout layer between them. The encoded

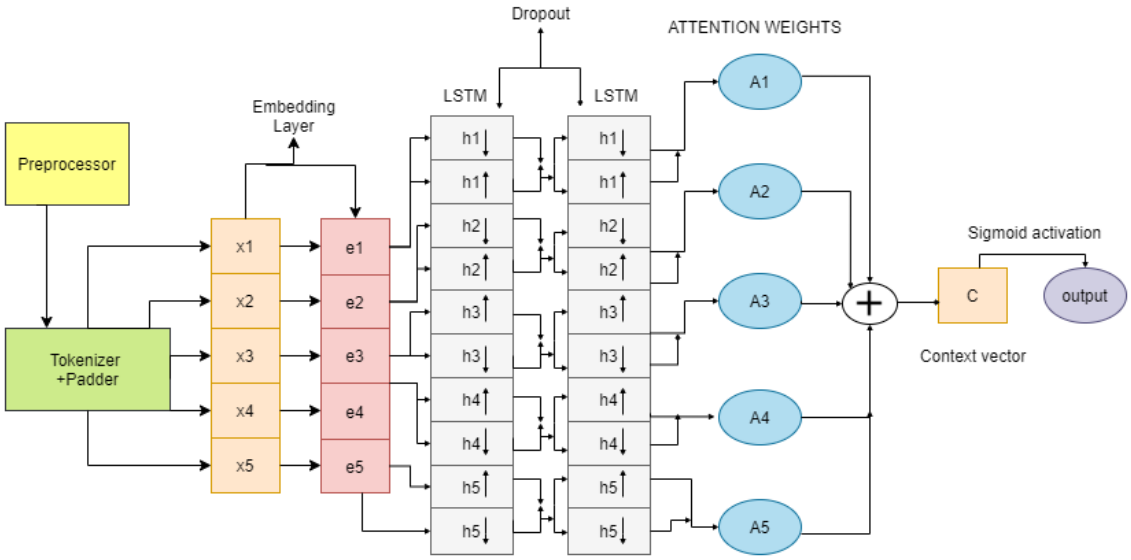


Figure 5.4.2: BiLSTM architecture

output is sent to the attention head which generates the context vector C. We let $H \in R^{bT_x u}$ to be the encoder output where b=batch size, T_x is the length of the input sequence and u=hidden units size. H is the hidden output of the stacked LSTM for each input token. $E \in R^{bT_x u}$ is the associated energy vector of attention weight vector $A \in R^{bT_x u}$. These three vectors are computed in the following way:

$$e^{ij} = w^T H$$

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

a_{ij} is the attention weight that is given to the encoder output h_j of H by the attention head. This is derived via softmax function so that all the attention weights equal to one. The softmax function is applied over the associated energy vector E which is a feed forward network with weights w_T being updated via backpropagation. This feed forward network determines via backpropagation how much weight should be assigned to each input h_j. The final output or context vector C is just the dot product between the Attention A and input H. In order to visulaize how the attention layer is prioritizing some input texts over others, we plot the attention weights output after training the bi-LSTM model. In Figure 5.4.3, we plot a clickbait title tokens in the x-axis and attention weights on the y-axis of input **আপনার বয় ফ্রেন্ড আছে কিনা জবাবে যা বললেন জান্নাতুল ঐশী** . From this we can see that the model is giving more priority to **জবাবে যা বললেন জান্নাতুল** .This words are clickbaity as they are increasing users' curiosity about something someone said and our model is giving more attention weights to these clickbait defining words.
We also experiment with another attention head called additive attention or Bahdanau Attention by Bahdanau *et al.* [43]. This is called additive attention because the attention weights not only depends on the encoded output H but also the cell

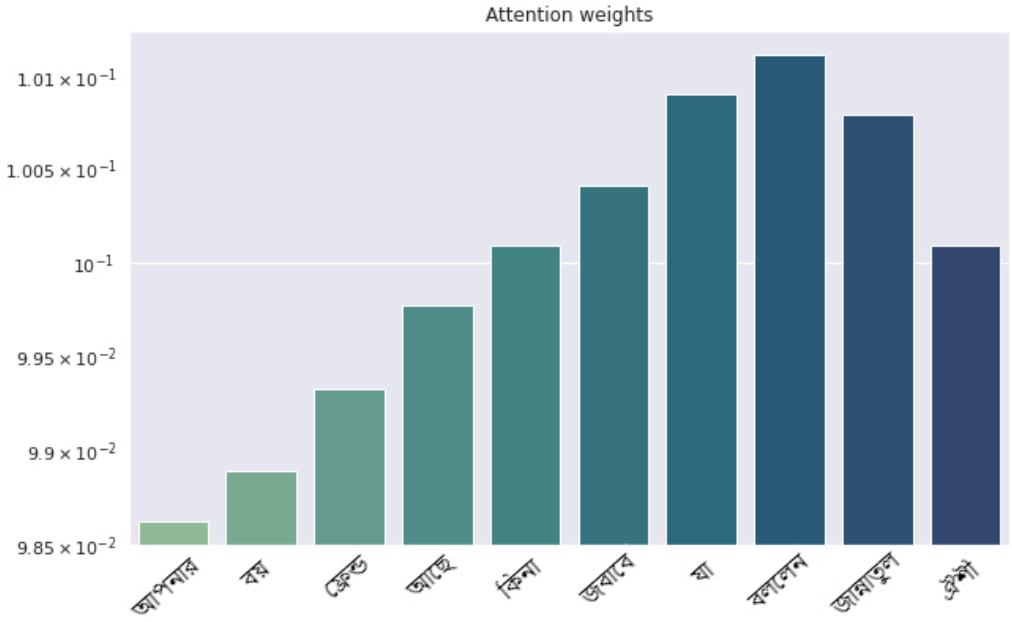


Figure 5.4.3: Attention weights of an input clickbait title

state C of the last input token. Here as we are implementing bi-LSTM, we concatenate the state of the first and last input token to build C. This is passed as Query vector $Q \in R^{b1u}$ where u=hidden units size and b=batch size. The calculation of the context vector C is provided below-

$$E = w^T(tanh(w_1^T Q + w_2^T H))$$
$$A = softmax(E)$$
$$C = dot(A, H)$$

where $w \in R^{bu1}$, $w_1 \in R^{buu}$, $w_2 \in R^{buu}$,$E \in R^{bT_x1}$, $A \in R^{bT_x1}$ and $C \in R^{bu}$. The output of the attention layer is passed onto a single sigmoid activation layer to generate the predictions.

5.4.3 BiGRU

Gated Recurrent Unit (GRU) is very similar to LSTM. It can also be said the a lighter version of LSTM and in some cases, GRU seem to perform similar to LSTM as well. The reset gate vector and the update gate vector are same in GRU and are calculated the same way as LSTM.

5.5 Transformer Models

Transformer models [45] took self attention mechanism described in previous section to a new height. Instead of complex recurrent and convolution techniques used previously in encoder-decoder models, the researchers were able to prove that self attention mechanism sufficed to learn language context given enough training data. Since their inception, transformers and its variants have performed superior

to other CNN and LSTM based models in GLUE [46], SuperGLUE [47] and SQuAD The Stanford Question Answering Dataset (SQuAD) provided by Rajpurkar *et al.* [48] benchmarks that constituted different tasks like sentiment analysis, semantic textual similarity, natural language inference, question answering etc. Similar to Image based models where transfer learning is standard, Transformer models ushered the era of transfer learning in natural language processing. Bidirectional Encoder Representations from Transformers (BERT) [3] constructed their training process in such a way that the same pre-trained model can be fine tuned using a small task specific model head without changing the main model architecture. When trained on enough data, BERT can act as a zero shot classifier as it can perform on par with other models without fine tuning. We use pre-trained models that were either multi-lingual or pre-trained specifically on Bengali data. We use the following BERT models for our task:

1. mBERT: is a multilingual version of BERT that supports 104 languages including Bangla trained on Wikipedia corpus data and available on HuggingFace's transformer library.[1]

2. Bangla BERT Base: Same architecture as Bert model and available on HuggingFace's transformer library.[2] and pretrained on Bangla Wikipedia Dump dataset and Bangla commoncrawl corpus dataset from OSCAR.

3. Indic-BN-BERT: Available on HuggingFace's transformer library.[3] and pretrained on 3GB of multilingual corpus from OSCAR dataset.[4]

Roberta [4] pre-trains BERT with larger batch size of 8k and on a larger dataset of 160GB of text and only on a new dynamically changing masked language modelling training objective. The Roberta model we used is Indic-BN-RoBERTa avialable on Huggingface's Transformers library.[5]. Distilbert [5] uses knowledge distillation [49] method to reduce orignal Transformer size by 40% while maintaining similar performance with 60% faster inference time. We used the following Distlbert models for our task:

1. Indic-BN-Distilbert: Avialable on Huggingface's Transformers library[6] and pretrained on OSCAR dataset[7].

2. Distilbert-multilingual: Available on Huggingface's Transformers library [8] and pretrained on Wikipedia data of 104 different languages.

ELECTRA [6] pre-trains the transformer encoder in a discriminative style where the encoder tries to predict whether the masked token came from a generator model or was actually part of the original text rather than generating the original text. It produces much better encoded representations than BERT with less pre-training time. The ELECTRA models we used are as follows:

[1]https://huggingface.co/bert-base-multilingual-cased
[2]https://huggingface.co/sagorsarker/bangla-bert-base
[3]https://huggingface.co/neuralspace-reverie/indic-transformers-bn-bert
[4]https://oscar-corpus.com/
[5]https://huggingface.co/neuralspace-reverie/indic-transformers-bn-roberta
[6]https://huggingface.co/neuralspace-reverie/indic-transformers-bn-distilbert
[7]https://oscar-corpus.com/
[8]https://huggingface.co/distilbert-base-multilingual-cased

1. Bangla-Electra: Avialable on Huggingface's Transformers library[9] and pretrained on OSCAR[10] and Bangla Wikipedia dump[11]

2. CSENLP-BanglaBert: Available on Huggingface's Transformers library [12] and pretrained on Bangla Natural Language Inference (NLI) dataset.[13]

XLM-RoBERTa [7] is a multilingual Roberta model pretrained on 2.5TB of filtered CommonCrawl data for 100 different languages which performs better than other multi-lingual models. The XLM-RoBERTa used for our task are:

1. Indic-BN-XLM-RoBERTa: Available on Huggingface's Transformers library[14] and pre trained on 3GB of multilingual corpus from OSCAR dataset.[15]

2. XLM-RoBERTa Large: Available on Huggingface's Transformers library[16] and pretrained on commoncrawl corpus dataset.

For our clickbait detection task, we experiment with two different approaches 1. article title 2. article title and content combined
This is because clickbait content often underdelivers what is promised in the title. The Transformer models can encode relationship between sequence pairs in their special [CLS] token which might be useful to detect clickbaits. So, when using title and content both, we pass title and content pair as input to the model. The Transformer models output encoded vector for each token of the sequence. The first token is the special [CLS] token. Although, it does store whole sequence's information and discriminative information between sequence pairs, it is not entirely effective. That is why we take the average of all the encoded vector instead and pass it to a classifier head. The classifier head is only a dropout layer followed by a binary classifier. The Transformer encodings contain rich semantic and contextual information of sequence and works good as features. We also use Transformers as features extractors where the hidden states from previously fine tuned Transformer models are passed to an bi-LSTM model with the same architecture as the previously described bi-LSTM model in section 5.4.2 as embeddings and trained separately. Throughout this paper, we address the Transformer models that are finetuned as Transformer finetuning and Transformer models whose features are passed to an LSTM model as Transformer feature extractors. The entire model architecture is described in figure 5.5.1

[9]https://huggingface.co/monsoon-nlp/bangla-electra
[10]https://oscar-corpus.com/
[11]https://dumps.wikimedia.org/bnwiki/
[12]https://huggingface.co/csebuetnlp/banglabert
[13]https://huggingface.co/datasets/csebuetnlp/xnli_bn
[14]https://huggingface.co/neuralspace-reverie/indic-transformers-bn-xlmroberta
[15]https://oscar-corpus.com/
[16]https://huggingface.co/docs/transformers/model_doc/xlmroberta

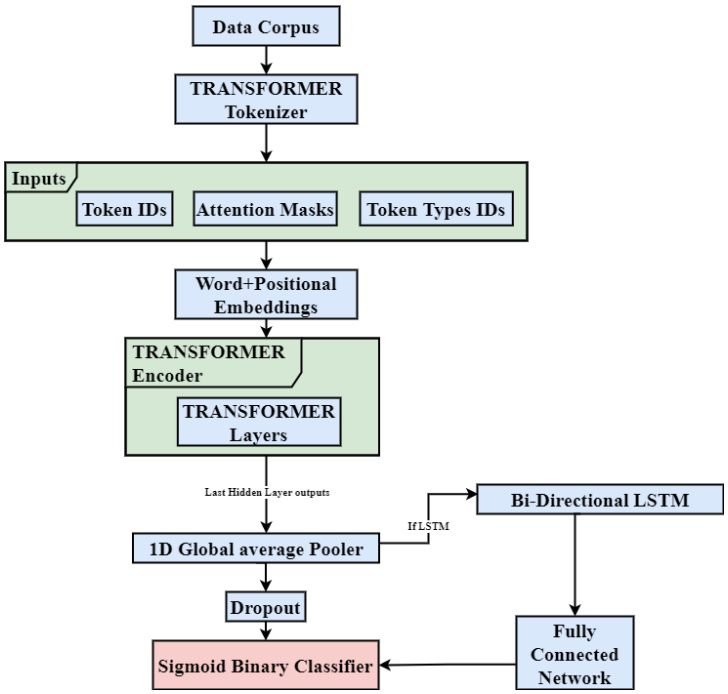


Figure 5.5.1: Transformer based model's architecture

Chapter 6

Experimental Setup

6.1 Dataset Preprocessing

There were many instances where the words in title and content had punctuation - mostly apostrophe (') in between letters which might cause many embedding-based approaches to perform poorly. Such as example is - **নামাজ পড়তে সম'স্যা হওয়ায় অভিনয় ছা'ড়লেন নায়িকা মুক্তি..!** here **সম'স্য ছা'ড়লে,** have apostrophes inside words. We get rid of these unwanted punctuation. Punctuation removal is important as many news portals with mostly clickbait news, insert punctuation marks in the middle of words to avoid getting detected by the ad services. There were also unnecessary text like **২৪ ঘন্টা আগে,আরো পরুন** which does not contribute to clickbait detection and were removed. The Normalizer [50] module was used for unicode normalization, removing HTML tags, URL links,the newline escape sequences (\n), emojis which are necessary after scraping texts from websites. The punctuation was preserved as punctuation can be considered as features in the model. Duplicate title and contents were removed from the dataset and the publication time was converted into datetime variable type for better data exploration using dateparser[1] module. After removing duplicate news articles, our dataset had 13,460 news articles of which 8813 were non clickbait and 4647 were clickbaits.

6.2 Experimental Setup

We split our dataset in 90:10 ratio of training-to-test. We experimented on three combinations of the data: title+content, only title and only content.
For our statistical classifier models, we use Grid Search Algorithm to with Stratified 5-fold cross validation to tune hyper-parameters in statistical models for our train set. As, our text classification task is not a linear task, we need to rely on a Logistic Regression solver method which provides the least error for our text classification problem. There are several solver methods such as Newton's Method, Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm(L-BFGS), Library for Large Linear Classification (LIBLINEAR), Stochastic Average Gradient (SAG), SAGA etc. We use LIBLINEAR for our classification problem as our dataset is fairly large. Also, to avoid overfitting we use regularization method in our Logistic Regression such as l1, l2 and elasticnet. We are using l2 regularization method

[1]https://dateparser.readthedocs.io/en/v0.3.1/

for this problem as LIBLINEAR is compatible with l2 regularization and l2 penalizes shrinking a factor not eliminating it so that we do not completely lose any coefficients while minimizing error. According to Grid Search result, the value for hyper-parameter Inverse of Regularization Strength for Unigram, TF-IDF character level-3,4 5 should be 10, for Bigram 1000 and for word embeddings 1. Decision Tree model can run on two different criterion function that measures thequality of splits in tree, which are gini and entropy. the TF-IDF character vectorization feature extractions gave better results on gini criterion and other feature extractions gave better results on entropy criterion. Decision Tree model can run on two different criterion function that measures the quality of splits in tree, which are gini and entropy. the TF-IDF character vectorization feature extractions gave better results on gini criterion and other feature extractions gave better results on entropy criterion. Random Forest can on two different criterion function similar to decision tree model. But we did not use Grid Search Algorithm to determine criterion as it takes very long time to train on entropy criterion function. We tuned hyper-parameters maximum depth, maximum number of features to consider, number of trees in the forest and Bagging of Random Forest Classifier model using Grid Search model for every language features extractions. Gaussian Naive Bayes model is a simple classification model that requires less hyper-parameters tuning than the other models used here. For our classification problem we did not set any prior probabilities but we tuned variance of the distribution of our Gaussian Naive Bayes model according to the best results from Grid Search Algorithm for each of the feature extraction method we used. Our variance hyper-parameter varies from 1e-6 to 1. K-nearest Neighbour creates cluster for our texts to classify them and it can run on different algorithms to create those clusters such as Ball Tree, KD Tree and Brute Force. In our clickbait classification problem we set auto for the algorithm choosing part and so, our model finds the best algorithm itself after checking the train set. Also, It requires various hyper-parameters such as number of neighbours, weight function, leaf size of clusters, distance metric and power parameter for that distance metric. We used minkowski metric and used Grid Search algorithm to find out the best power parameter of minkowski metric for our train set, which we found out to be either 1 or 2 as per the features. Then we again used Grid Search algorithm to find out the best weight function, leaf size and number of neighbors, which were different for every feature. Support Vector Machine(SVM) also creates cluster to analyze provided dataset but unlike other models, to acquire the best categorization results for any particular classificaion problem, numerous hyper-parameters must be specified appropriately for SVM. Although SVM runs on various kernals such as linear, poly, rbf, sigmoid and precomputed, text classification problems run better on either rbf kernal of linear kernal and so we we ran Grid Search algorithm for these two choices. After setting the best kernal we set the kernal co-efficient to auto as linear kernal do not require any kernal co-efficient but rbf kernal does. Then we used Grid Search algorithm again to find out the best regularization parameter for our features which ranges from 1 to 20.

Figure 4.3.4 shows the token length of title and content of our training data. We choose token length =10 for title and token length = 750 for our content. Title and content less than this length was padded with zero and greater than this length was truncated. This fixed size tokens were passed to the Embedding layer to generated

corresponding embeddings that were passed as features to CNN and LSTM models. For CNN and LSTM model, we perform Hyperband search on different parameters shown by Li *et al.* [51]. Firstly for CNN, the search is performed on the kernel size 1-10 for all four different kernel channels. We also search for the optimum number of hidden units of each layer and dropout ratio before the penultimate layer. For CNN, kernel sizes of 1-4, dropout ratio of 0.5 and 256 hidden units for all kernel work best. For LSTM, we performed the search on hidden units of LSTM layer and dropout ratio between two LSTM layers. Hidden units of 256 and dropout ratio of 0.5 works the best in this case for both the LSTM and GRU model. We used Adam optimizer [52] where we specify early stopping on F1-score of fake class with patience=3 and decrease learning rate by 90% if the validation loss does not decrease for 2 epochs. We use binary cross entropy as the loss function to minimize. In order to prevent gradient vanishing or explosion, we clip the gradients at value 2 but this will require more fine tuning. We perform 5 fold stratified cross validation with epoch=20 for each round. For the attention layer, we will check whether normal attention or additive attention performs better. We will experiment with word2vec, GloVe, fastText and self trained embeddings to evaluate which performs best in our case and whether these embeddings should be trained along with the network or kept fixed. As LSTM learns long range dependencies and CNN short term dependencies, we will check whether concatenating these two layers' output can lead to better results.

The Transformer models can be quickly fine tuned as they are already pre trained. When title and content were passed in pairs, their total length was truncated to maximum 512 tokens for all models except XLM-RoBERTa models. For XLM-RoBERTa models, the tokens were truncated to 256 tokens to save memory. Mini batch size for title was 16 and for title,content pair 8 except for XLM-RoBERTa models where batch size was 4 for because of GPU memory constraints. The attention dropout probability, hidden layer dropout probabiltiy of Transformer models was 0.3 . For Transformer finetuning models, dropout between last layer of Transformer and classification head was 0.3 . For all training, we split the dataset into 80:10:10 fashion for training,validation,test splits respectively. The held out test set was used for the final result of all the models. Weight initialization and learning rate seems to have a great effect on training metrics. We experiment with different random seeds,learning rate and use EarlyStopping(patience=3) on validation F1-Score to terminate bad trials. Based on this,$learning_rate = e - 5$ and best random initialization was used to train the model. For Transformer finetuning models, we used Hugginface's AdamW [53] optimizer with AdamW's betas parameters (b1, b2) = (0.9, 0.999), $epsilon = 1e-6$ and weight decay = 0.01. We checked the validation F1-Score after every thousand training steps and saved the best model based on that. The Transformer feature extractor LSTM model has all the same parameters as the Bi-LSTM model. But to prevent overfitting, we decreased the LSTM hidden layer size to 64 and set the dropout to 0.5 between two LSTM layers. Categorical cross entropy loss was used in both Transformer finetuning and feature extractor approaches. The four experiments - Transformer Finetuning (title, title+content) and Transformer feature extractors (title, title+content) were repeated five times and the average accuracy (overall), precision, recall, f1-score of clickbait class on test data were reported.

6.3 Results and Analysis

6.3.1 Statistical Classifier Models

After extracting features (unigram, bigram, TF-IDF character level-3, TF-IDF character level-4, TF-IDF character level-5, word embedding using fastText GloVe word vector models, vectorized bnwiki dump and self trained word embedding) from train set we put use Grid Search algorithm with Stratified 5-fold cross validation to tune hyper-parameters for our statistical classifier models. Then we pass full train sets to the statistical classifier models with the best parameters found in Grid Search cross validation.
In Logistic Regression, we got the best result after using Unigram features where we got 76.97% accuracy, 68.32% precision, 62.15% recall and 65.09% f1 score. Accuracy remains almost same on every preprocessings for Logistic Regression. Results of Logistic Regression are shown in table 6.1.

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	76.97	68.32	62.15	65.09
1	bigram	73.63	64.40	52.90	58.09
2	char_3	74.81	65.00	58.71	61.69
3	char_4	75.78	67.42	57.85	62.27
4	char_5	75.50	62.93	57.03	59.84
5	word_bnwiki	70.95	61.14	43.66	50.94
6	word_fasttext	70.51	60.49	42.15	49.68
7	word_glove	71.99	62.57	47.10	53.74
8	word_self_trained	71.55	67.67	33.76	45.05

Table 6.1: Clickbait classification report of Logistic Regression

In Decision Tree, we got the best accuracy after using Bigram features where we got 70.28% accuracy, 58.76% precision, 46.88% recall and 52.15% f1 score. However, we got the best f1 score after using TF-IDF character level-3 features where we chieved 68.35% accuracy, 53.92% precision, 57.63% recall and 55.72% f1 score. Results of Decision Tree are shown in table 6.2.

In Random Forest, we got the best result after using Unigram features where we got 77.6% accuracy, 75.31% precision, 52.47% recall and 65.09% f1 score. Classification report of Random Forest is illustrated in the table 6.3.

In Gaussian Naive Bayes, we got the best accuracy after using TF-IDF character level-5 features where we got 72.25% accuracy, 58.59% precision, 45.31% recall and 51.10% f1 score. However, we got the best f1 score after using bnwiki word embedding where we found 65.08% accuracy, 49.62% precision, 70.97% recall and 58.41% f1 score. Classification report for Gaussian Naive Bayes classifier is illustrated in the table 6.4.

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	68.42	54.13	56.34	55.22
1	bigram	70.28	58.76	46.88	52.15
2	char_3	68.35	53.92	57.63	55.72
3	char_4	67.16	52.25	57.42	54.71
4	char_5	67.00	48.61	54.69	51.47
5	word_bnwiki	65.23	49.68	49.68	49.68
6	word_fasttext	63.52	47.35	49.89	48.59
7	word_glove	64.93	49.25	49.46	49.36
8	word_self_trained	61.66	44.65	45.81	45.22

Table 6.2: Clickbait classification report of Decision Tree

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	77.64	75.31	52.47	61.85
1	bigram	70.43	84.54	17.63	29.18
2	char_3	77.04	75.32	49.89	60.03
3	char_4	76.00	74.65	46.24	57.10
4	char_5	76.75	77.78	38.28	51.31
5	word_bnwiki	74.37	72.39	41.72	52.93
6	word_fasttext	74.07	71.01	42.15	52.90
7	word_glove	72.81	70.54	36.56	48.16
8	word_self_trained	70.21	64.81	30.11	41.12

Table 6.3: Clickbait classification report of Random Forest

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	59.81	44.85	71.18	55.03
1	bigram	62.78	47.51	73.98	57.86
2	char_3	57.88	43.48	73.12	54.53
3	char_4	68.35	53.89	58.06	55.90
4	char_5	72.25	58.59	45.31	51.10
5	word_bnwiki	65.08	49.62	70.97	58.41
6	word_fasttext	63.37	47.94	70.11	56.94
7	word_glove	63.30	47.91	71.40	57.34
8	word_self_trained	53.57	38.92	60.43	47.35

Table 6.4: Clickbait classification report of Gaussian Naive Bayes

In K-nearest Neighbour classifier, we got the best accuracy after using Unigram features again where we got 73.11% accuracy, 67.22% precision, 43.23% recall and 52.62% f1 score. However, we got the best f1 score after using TF-IDF character level-4 features where we achieved 72.50% accuracy, 58.57% precision, 63.23% recall and 60.81% f1 score. Classification report for K-nearest Neighbour classifier model

is illustrated in the table 6.5.

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	73.11	67.22	43.23	52.62
1	bigram	66.94	52.36	47.74	49.94
2	char_3	71.99	63.17	45.38	52.82
3	char_4	71.84	58.57	63.23	60.81
4	char_5	72.50	60.23	41.41	49.07
5	word_bnwiki	71.32	61.13	46.67	52.93
6	word_fasttext	70.65	56.89	62.15	59.40
7	word_glove	68.05	53.37	59.57	56.30
8	word_self_trained	64.71	48.11	27.31	34.84

Table 6.5: Clickbait classification report of K-nearest Neighbour

In SVM classifier, we got the best accuracy after using Unigram features again where we got 78.01% accuracy, 71.28% precision, 60.86% recall and 65.66% f1 score. Classification report for SVM classifier model is illustrated in the table 6.6.

	Preprocessing	Accuracy	Precision	Recall	F1 Score
0	unigram	78.01	71.28	60.86	65.66
1	bigram	72.81	61.81	55.70	58.60
2	char_3	76.45	70.44	54.84	61.67
3	char_4	76.52	71.35	53.55	61.18
4	char_5	74.00	59.84	57.03	58.40
5	word_bnwiki	70.43	56.94	59.14	58.02
6	word_fasttext	73.18	64.77	49.03	55.81
7	word_glove	73.63	66.87	46.88	55.12
8	word_self_trained	68.72	78.95	12.90	22.18

Table 6.6: Clickbait classification report of SVM

From the classification reports of the experiments conducted via statistical classifier models we have found out that different model had better score on different features but most of the models gave good scores for Unigram features. Also, the best scores were attained from SVM classifier after using unigram features where we got 78.01% accuracy and 65.66% f1 score. Figure 6.3.1 demonstrates the best results achieved from statistical classifier models. Also, figure 6.3.2 illustrates the confusion matrix of the best model among the statistical classifiers, which is SVM after using unigram features.

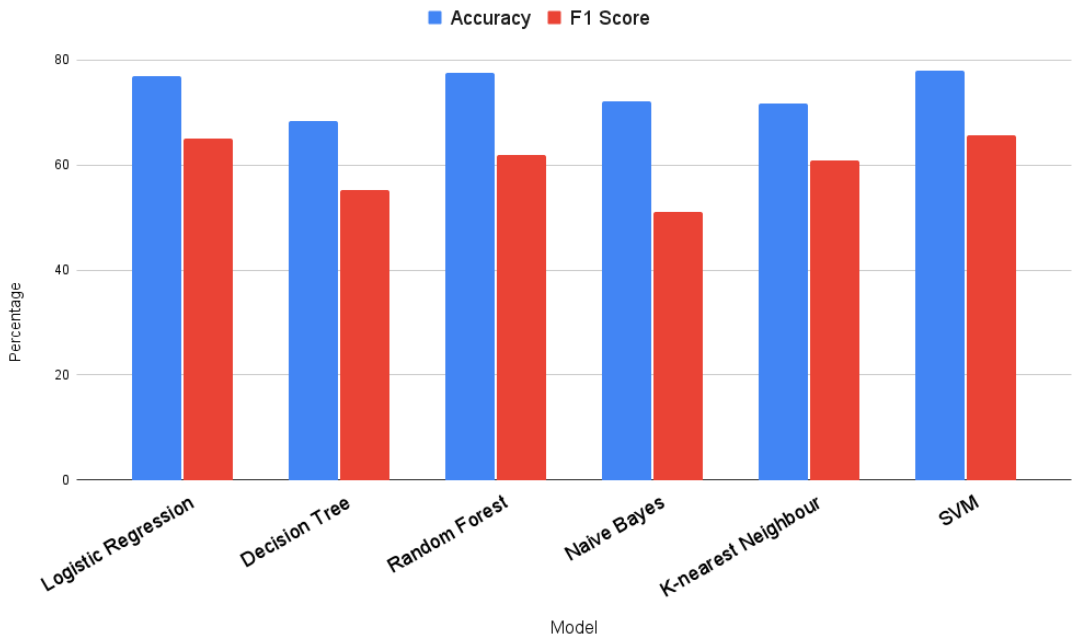


Figure 6.3.1: Best results from statistical classifier models

Figure 6.3.2: Confusion Matrix of the best model (SVM with Unigram)

6.3.2 Neural Network Models

For neural network models, we worked on with CNN, LSTM and BiGRU. Here, we experimented with three types of word embeddings named word2vec, fast-text, and GloVe and tried to evaluated their performance for different neural network models. The best outcome has been show in Table 6.8 for all three models detecting clickbait on news title.

Method	Preprocessing	Accuracy	Precision	Recall	F1 Score
CNN	word2vec (wv)	74.81	65.22	58.08	61.43
	fasttext (ft)	75.11	65.78	52.28	61.8
	GloVe (gv)	77.19	70.15	59.14	64.18
LSTM	word2vec (wv)	72.14	59.53	60.43	59.98
	fasttext (ft)	73.48	60.98	64.52	62.7
	GloVe (gv)	73.25	61.10	62.15	61.62
LSTM (Attention)	word2vec (wv)	74.37	63.7	60.00	61.79
	fasttext (ft)	72.51	60.30	59.78	60.04
	GloVe (gv)	75.63	66.91	58.28	62.30
BiGRU	word2vec (wv)	74.44	63.47	61.29	62.36
	fasttext (ft)	74.15	62.91	61.29	62.09
	GloVe (gv)	74.00	63.10	59.57	61.28
BiGRU (Attention)	word2vec (wv)	75.41	65.88	59.78	62.68
	fasttext (ft)	73.55	61.72	61.72	61.72
	GloVe (gv)	75.93	66.67	60.65	63.51

Table 6.7: News-title clickbait classification report of Neural Network models

The Table 6.7 illustrates the accuracy, precision, recall and f1 score of CNN model for detecting clickbait news-title for both of the word embeddings word2vec, fast-text and GloVe. Here for CNN, both word2vec and fast-text performs almost equally with very less difference in each of the sections and generates approximately 75% accuracy, 66% precision, 58% recall and 62% f1 score. However, GloVe embedding performs better with approximately 77% accuracy, 70% precision, 59% recall and 64% f1 score.

Above Table 6.7 further describes the accuracy, precision, recall and f1 score of LSTM model for detecting clickbait news-title for the word embeddings word2vec, fast-text and GloVe. Here for LSTM, both word2vec and fast-text performs similar to CNN. However, GloVe gives approximately 75%, 67%, 68%, and 62% as accuracy, precision, recall and f1 score respectively.

Furthermore, the Table 6.7 describes the accuracy, precision, recall and f1 score of BiGRU model for detecting clickbait news-title for the word embeddings word2vec, fast-text and GloVe. Here for BiGRU, both word2vec and fast-text performs similar to CNN and LSTM, however, GloVe gives approximately 75%, 65%, 60%, and 62% as accuracy, precision, recall and f1 score respectively.

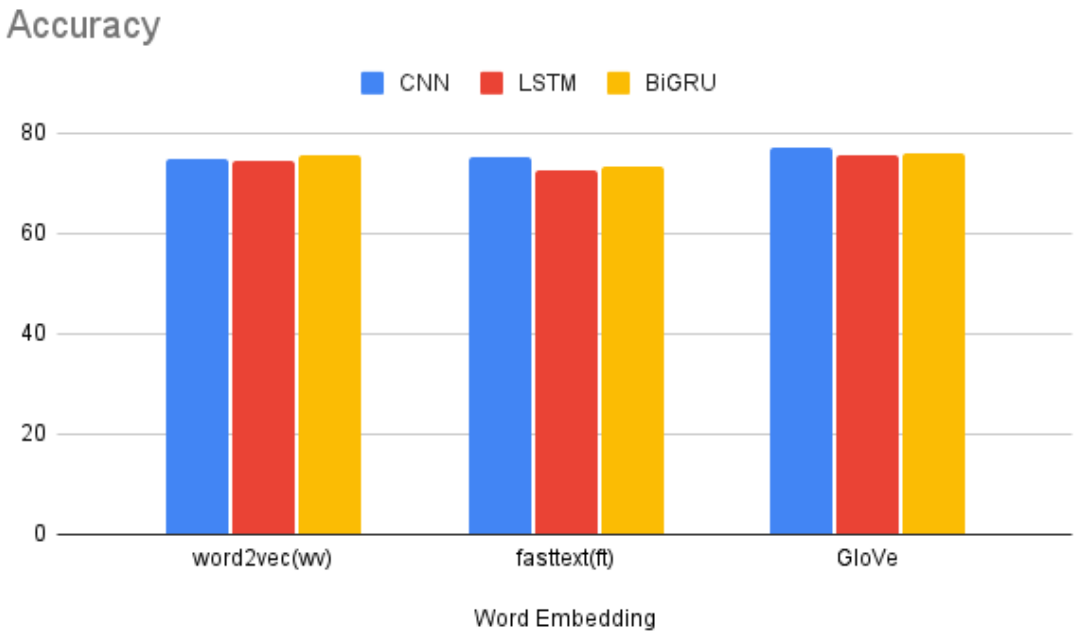


Figure 6.3.3: Accuracy comparison among Neural Network models

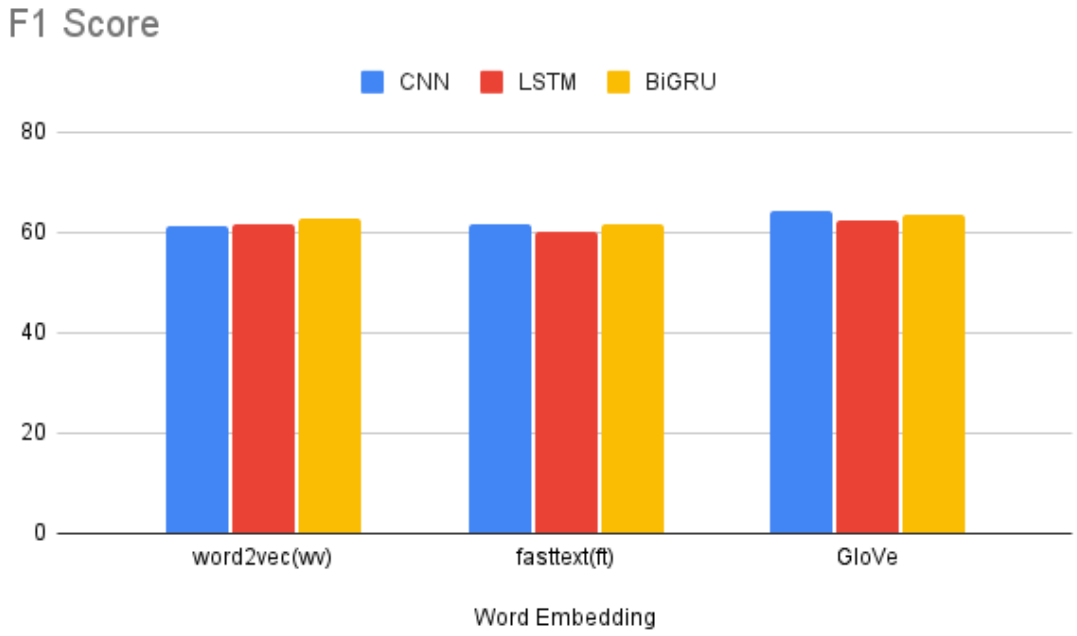


Figure 6.3.4: F1 Score comparison among Neural Network models

Method	Accuracy	Precision	Recall	F1 Score
CNN	77.19	70.15	59.14	64.18
LSTM	74.48	61.10	64.52	62.70
LSTM (Attention)	75.63	66.91	60.65	62.30
BiGRU	74.44	63.47	61.29	62.36
BiGRU (Attention)	75.93	66.67	61.72	63.51

Table 6.8: Best result from Neural Network models

From the Table 6.8 we can see that, even though CNN, LSTM, and BiGRU seem to perform equally for the best cases, CNN is better by a very slight margin.

6.3.3 Transformer Models

Table 6.10 shows all the results performed on article title and table 6.9 shows all the results when title and content are passed as sentence pairs. Indic-BN-Bert performs the best on title and reports 72.37% F1-Score and 80.08% accuracy. Indic-BN-Distilbert performs the best on title-content combined and reports 72.13% F1-Score and 79.79% accuracy. All Feature extractor Transformer models degrade the performance of Transformer finetuning models. The best performing model there was Indic-Roberta which reported 64.37% F1-Score. According to these results, Transformer models outperform the best models from statistical classifier and Deep Neural network models (CNN,Bi-LSTM,Bi-GRU). This is because Transformer networks can learn the contextual and syntactical information of text better than these models.But unlike other Transformer based models, XLM-Roberta multilingual model fails to detect any clickbaits. This might be the case of pretraining with commoncrawl data that are crawled from different web sources and may have not seen many news articles or that we need more data to see any performance from XLM-Roberta model. Huge models like XLM-Roberta require quite a lot of data to show their actual performance. This is not the calse always as Indic XLM-Roberta model performs well on titles. So, the issue might lie in their pre training process. All the transformer models trained on only titles except Indic Distilbert outperform their title-content combined counterpart. This proves the fact that title plays a very significant role in identifying clickbait. That is why there should be more emphasis on the stylistic cue of titles to improve these model's metrics. The Transformer Feature Extractor models are trained on only clickbait titles and their results are shown in 6.11. All of the models' performance worsen when the finetuned transformer embeddings are passed into LSTM network. Thus, the models are not trained on title-content combined. This proves that the attention mechanism is Transformer network is sufficient to encode the inter dependency between sequence tokens. Figure 6.3.5 shows the full results of three best performing models from each section. Indic-Distilbert is the best performer as its gives the best performance when title and content are combined and also performs similarly with Indic-Bert model on title. So, we can conclude that Indic-Distilbert model can be deployed into real world to detect clickbaits in Bengali news articles.

Model	Accuracy	F1-Score	Precision	Recall	Loss
mBERT	0.736998514	0.675603	0.594548552	0.750537634	0.5492
Bangla Bert Base	0.756315	0.67002	0.62949	0.716129	0.643
Indic Bert	0.768945022	0.661588683	0.669603524	0.653763441	0.6427
Indic Roberta	0.746656761	0.682790698	0.601639344	0.789247312	0.5567
Distilbert multilingual	0.757800892	0.66322314	0.638170974	0.690322581	0.73813
Indic Distilbert	0.797919762	0.721311475	0.688845401	0.756989247	0.74684
Bangla Electra	0.741456166	0.669201521	0.599659284	0.756989247	0.54799
CSEBuetNLP Bangla Bert	0.783803863	0.706357215	0.66539924	0.752688172	0.64815
XLM-ROBERTA Large	0.654531947	0	0	0	0.65485
Indic XLM-ROBERTA	0.777117385	0.678111588	0.676659529	0.679569892	1.1499

Table 6.9: Results on Finetuning Transformer models on title and content combined

Model	Accuracy	F1-Score	Precision	Recall	Loss
mBERT	0.757800892	0.666	0.636007828	0.6989	0.556323051
Bangla Bert Base	0.791233284	0.6881	0.711009174	0.667	0.633811712
Indic Bert	0.80089153	0.7237	0.695049505	0.7548	0.4733271
Indic Roberta	0.775631501	0.7022	0.64845173	0.7656	0.558732808
Distilbert multilingual	0.754829123	0.6771	0.621184919	0.7441	0.536099553
Indic Distilbert	0.799405646	0.7199	0.695390782	0.7462	0.531267524
Bangla Electra	0.762258544	0.6708	0.642998028	0.7011	0.509521484
CSEBuetNLP Bangla Bert	0.790490342	0.7087	0.681908549	0.7376	0.52011174
XLM-ROBERTA Large	0.654531947	0	0	0	0.648890674
Indic XLM-ROBERTA	0.772659733	0.7018	0.64171123	0.7742	0.503365397

Table 6.10: Results on Finetuning Transformer models on title

Model	Accuracy	F1-Score	Precision	Recall	Loss
mBERT	0.754437864	0.479733527	0.599704146	0.452324599	0.5109
Bangla Bert Base	0.773668647	0.592722595	0.619723856	0.641251028	0.528
Indic Bert	0.776627243	0.628307939	0.637813449	0.686376452	0.48167
Indic Roberta	0.773668647	0.643769383	0.647745788	0.709847867	0.48103
Distilbert multilingual	0.770710051	0.582766235	0.627542973	0.601704717	0.4893
Indic Distilbert	0.78550297	0.641640604	0.644012392	0.706635654	0.6100
Bangla Electra	0.780325472	0.555520058	0.661932945	0.545139492	0.4944
CSEBuetNLP Bangla Bert	0.790680468	0.633271515	0.634601295	0.69742173	0.45718
XLM-ROBERTA Large	0.653846145	0	0	0	0.66373
Indic XLM-ROBERTA	0.763313591	0.630877197	0.61349678	0.718624949	0.52128

Table 6.11: Results on Features Extracted from Transformer models on title

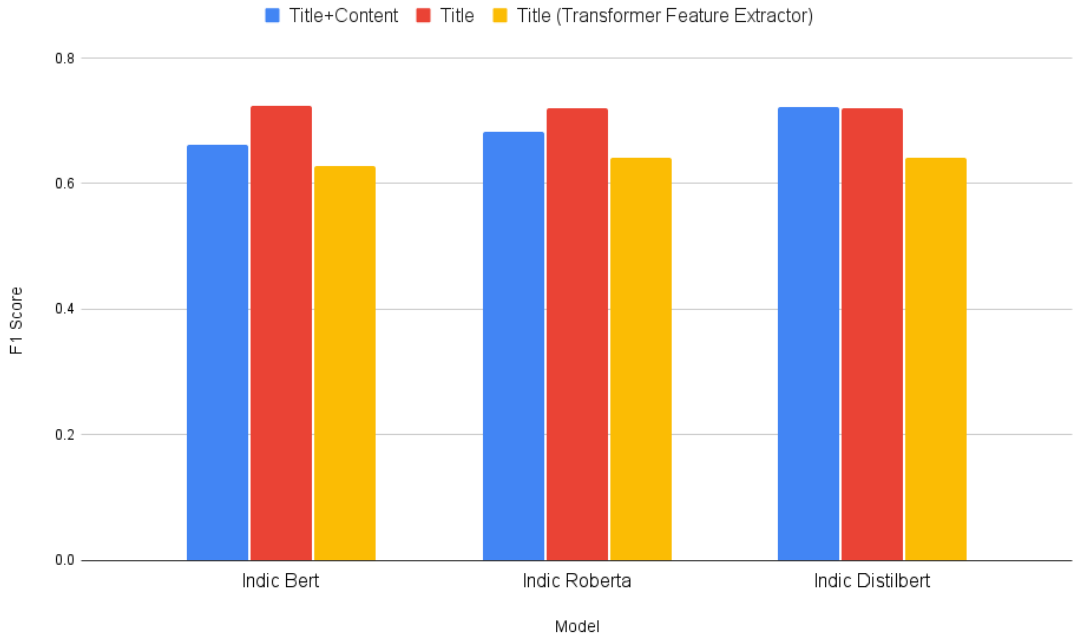


Figure 6.3.5: BERT models Score summary

Chapter 7

Conclusion

We present the first labeled clickbait detection with 13,460 news articles with variety of information like article publish time, publisher, title and content to enable researchers to use this dataset to build state of the art clickbait detection models. As this is the first research in clickbait detection in Bengali news artilces, we provide a baseline for future researchers by conducting comprehensive study on linear models, deep neural networks and Transformer models. Our key finding is that Transformer models outperform linear and deep neural network models by capturing title and content context. To further increase the performance, we think that Pre training should be done focused on Bengali news articles to facilitate these Transformer models with the difficult to detect stylistic pattern of clickbait titles. Furthermore, more emphasis should be given to find novel methods on how to incorporate content with the titles to detect those sneaky titles that would be hard to detect without looking at their contents. We wish to extend our dataset size to 40k and publicly release it. We hope our dataset will provide researchers to innovate new methods for detecting clikcbaits in Bengali news articles.

Bibliography

[1] J. Murtha, "What it's like to get paid for clicks," *Columbia Journalism Review*, Jul. 13, 2015. [Online]. Available: https://www.cjr.org/analysis/the_mission_sounds_simple_pay.php (visited on 06/03/2021).

[2] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer, "" 8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter*, 2020. arXiv: 1910.01108 [cs.CL].

[6] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, *Electra: Pre-training text encoders as discriminators rather than generators*, 2020. arXiv: 2003.10555 [cs.CL].

[7] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, *Unsupervised cross-lingual representation learning at scale*, 2020. arXiv: 1911.02116 [cs.CL].

[8] S. E. Bird, "Tabloidization," *The International Encyclopedia of Communication*, 2008.

[9] Y. Chen, N. J. Conroy, and V. L. Rubin, "Misleading online content: Recognizing clickbait as" false news"," in *Proceedings of the 2015 ACM on workshop on multimodal deception detection*, 2015, pp. 15–19.

[10] H.-T. Zheng, J.-Y. Chen, X. Yao, A. K. Sangaiah, Y. Jiang, and C.-Z. Zhao, "Clickbait convolutional neural network," *Symmetry*, vol. 10, no. 5, 2018, ISSN: 2073-8994. DOI: 10.3390/sym10050138. [Online]. Available: https://www.mdpi.com/2073-8994/10/5/138.

[11] V. Indurthi, B. Syed, M. Gupta, and V. Varma, "Predicting clickbait strength in online social media," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 4835–4846.

[12] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, "Clickbait detection," in *European Conference on Information Retrieval*, Springer, 2016, pp. 810–817.

[13] M. Potthast, T. Gollub, M. Hagen, and B. Stein, "The clickbait challenge 2017: Towards a regression model for clickbait strength," *arXiv preprint arXiv:1812.10847*, 2018.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. 76, pp. 2493–2537, 2011. [Online]. Available: http://jmlr.org/papers/v12/collobert11a.html.

[15] M. Potthast, T. Gollub, K. Komlossy, *et al.*, "Crowdsourcing a large corpus of clickbait on twitter," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 1498–1507.

[16] Y. Zhou, *Clickbait detection in tweets using self-attentive network*, 2017. arXiv: 1710.05364 [cs.CL].

[17] M. Glenski, E. Ayton, D. Arendt, and S. Volkova, *Fishing for clickbaits in social images and texts with linguistically-infused neural network models*, 2017. arXiv: 1710.06390 [cs.LG].

[18] P. Thomas, *Clickbait identification using neural networks*, 2017. arXiv: 1710.08721 [cs.CL].

[19] A. Omidvar, H. Jiang, and A. An, "Using neural network for identifying clickbaits in online news media," in *Annual International Symposium on Information Management and Big Data*, Springer, 2018, pp. 220–232.

[20] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[21] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[22] A. Anand, T. Chakraborty, and N. Park, "We used neural networks to detect clickbaits: You won't believe what happened next!" In *European Conference on Information Retrieval*, Springer, 2017, pp. 541–547.

[23] M. M. U. Rony, N. Hassan, and M. Yousuf, "Diving deep into clickbaits: Who use them to what extents in which topics with what effects?" In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*, 2017, pp. 232–239.

[24] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, *Bag of tricks for efficient text classification*, 2016. arXiv: 1607.01759 [cs.CL].

[26] M. E. Peters, M. Neumann, M. Iyyer, *et al.*, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[27] D. Cer, Y. Yang, S.-y. Kong, *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[29] C. Wu, F. Wu, T. Qi, and Y. Huang, "Clickbait detection with style-aware title modeling and co-attention," in *China National Conference on Chinese Computational Linguistics*, Springer, 2020, pp. 430–443.

[30] M. Z. Hossain, M. A. Rahman, M. S. Islam, and S. Kar, "Banfakenews: A dataset for detecting fake news in bangla," *arXiv preprint arXiv:2004.08789*, 2020.

[31] Q. V. Le and T. Mikolov, *Distributed representations of sentences and documents*, 2014. arXiv: 1405.4053 [cs.CL].

[32] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *IN PROCEEDINGS OF THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP FRAMEWORKS*, 2010, pp. 45–50.

[33] S. Sarker, *Bnlp: Natural language processing toolkit for bengali language*, 2021. arXiv: 2102.00405 [cs.CL].

[34] abhishekgupta92, *Abhishekgupta92/bangla$_p$os$_t$agger: Pos tagger for bangla language based on conditional random fields*, Jul. 2012. [Online]. Available: https://github.com/abhishekgupta92/bangla_pos_tagger.

[35] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The journal of machine learning research*, vol. 3, pp. 1137–1155, 2003.

[36] A. Salle, M. Idiart, and A. Villavicencio, "Matrix factorization using window sampling and negative sampling for improved word representations," *arXiv preprint arXiv:1606.00819*, 2016.

[37] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," *arXiv preprint arXiv:1802.06893*, 2018.

[38] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 302–308.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[40] Y. Kim, *Convolutional neural networks for sentence classification*, 2014. arXiv: 1408.5882 [cs.CL].

[41] A. F. Agarap, *Deep learning using rectified linear units (relu)*, 2019. arXiv: 1803.08375 [cs.NE].

[42] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, *Natural language processing (almost) from scratch*, 2011. arXiv: 1103.0398 [cs.LG].

[43] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL].

[44] P. Zhou, W. Shi, J. Tian, *et al.*, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 207–212. DOI: 10.18653/v1/P16-2034. [Online]. Available: https://aclanthology.org/P16-2034.

[45] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[46] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[47] A. Wang, Y. Pruksachatkun, N. Nangia, *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.

[48] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[49] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML].

[50] T. Hasan, A. Bhattacharjee, K. Samin, *et al.*, "Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 2612–2623. DOI: 10.18653/v1/2020.emnlp-main.207. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.207.

[51] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-558.html.

[52] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[53] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG].