Clickbait Titles Detection with Machine and Deep Learning Methods

Krivičić, Armin

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:568022

University of Rijeka – Faculty of Informatics and Digital Technologies

Information and Communication Systems

Armin Krivičić

Clickbait Titles Detection with Machine and Deep Learning Methods

Master thesis

Mentor: prof. dr. sc. Sanda Martinčić-Ipšić

Rijeka, September 2023

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Informacijski i komunikacijski sustavi

Armin Krivičić

Otkrivanje "clickbaitova" pomoću metoda strojnog i dubokog učenja

Diplomski rad

Mentor: prof. dr. sc. Sanda Martinčić-Ipšić

Rijeka, rujan 2023.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 25.4.2023.

Zadatak za diplomski rad

Pristupnik: Armin Krivičić

Naziv diplomskog rada: Otkrivanje „clickbaitova“ pomoću metoda strojnog i dubokog učenja

Naziv diplomskog rada na eng. jeziku: Clickbait Titles Detection with Machine and Deep Learning Methods

Sadržaj zadatka:
Zadatak diplomskog rada je isprobati tehnike strojnog i dubokog učenja u svrhu otkrivanja „clickbaitova“ (hrv. mamilica). U sklopu pripreme podataka bit će izvršena analiza vokabulara i lingvističkih značajki. Metode strojnog učenja koje će biti učene uključuju: naivni Bayesov klasifikator, strojeve s potpornim vektorima (SVM), šumu slučajnih stabala (RF), perceptron i ostale metode gdje će se pomoću mjera točnosti, preciznosti i odziva odrediti koji klasifikacijski model najbolje odgovara zadanom problemu. Koristit će se više različitih reprezentacija teksta od osnovne TF-IDF do word2vec i FastText reprezentacije te će se odabrati najprimjerenija. U diplomskom radu će biti primijenjene metode učenja neuronskih mreža poput ćelije s dugoročnom memorijom (Long Short-Term Memory – LSTM) i upravljačke rekurentne jedinice (Gated Recurrent Unit – GRU). U radu će se koristiti već pripremljeni standardni skupovi podataka na engleskom jeziku poput onih koje su pripremili autori A. Chakraborty i A. Agrawal, a koji sadržavaju naslove koji su podijeljeni na „clickbait“ i „non-clickbait“. Također će se koristiti i skup podataka Webis-Clickbait-17 koji, uz naslove članaka, sadrži i cjelovite tekstove. S odabranom reprezentacijom, koja se na klasifikacijskom zadatku pokaže primjerenom, riješit će se i zadatak uspoređivanja usklađenosti naslova s tekstom koristeći vektorsku reprezentaciju i odgovarajuću mjeru sličnosti. Osim osnovnog mjerenja sličnosti (npr. kosinusna sličnost), naučit će se i neuronska mreža u odgovarajućoj arhitekturi, npr. model duboke semantičke sličnosti (Deep Semantic Similarity Model – DSSM) koja uči funkciju sličnosti.

Mentor:
Prof. dr. sc. Sanda Martinčić-Ipšić

Voditeljica za diplomske radove:
Prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 27.4.2023.

(potpis pristupnika)

Fakultet informatike i digitalnih tehnologija
Radmile Matejčić 2, 51000 Rijeka, Hrvatska
http://inf.uniri.hr ured@inf.uniri.hr
Tel: +385 (0)51584700 | Fax: +385 (0)51 584 749

YUFE

Abstract

This master's thesis addresses the challenge of clickbait detection in digital media through the application of machine learning and deep learning techniques. Clickbait is a type of web content advertisement that aims to attract users' attention and encourage them to click on related links. Media scholars have expressed concerns regarding clickbait's contribution to misinformation, yet the clickbait industry is still growing quickly. The objective of this thesis is to classify clickbait titles using different models and text representation techniques. Multiple datasets, including the dataset collected by Chakraborty et al. and the Webis-Clickbait-17 dataset, are utilized in the research. Furthermore, these two datasets were merged, and classification was also done on that merged dataset. Classification models such as the naive Bayes classifier, support vector machines, random forest, linear perceptron, LSTM, and GRU are employed. Three text representation techniques, TF-IDF, word2vec, and FastText, are used and compared. The findings demonstrate that FastText representation yields the highest overall performance. Regarding the classification performance, support vector machines achieve the best results, with an F1 score of 0.8797 on the merged dataset. The study also explores title-text similarity using the neural Deep Semantic Similarity Model, confirming that non-clickbait titles better represent the content. This research contributes insights into clickbait classification models, aiming to enhance user experience and combat misinformation.

Keywords: clickbait, machine learning, deep learning, classification models, text representation, title-text similarity, misinformation.

Sažetak

Otkrivanje "clickbaitova" pomoću metoda strojnog i dubokog učenja

Tema diplomskog rada je otkrivanje "clickbaitova" (mamilica) u digitalnim medijima koristeći tehnike strojnog i dubokog učenja. "Clickbait" je vrsta oglašavanja web sadržaja kojoj je cilj privući pozornost čitatelja i potaknuti ih da kliknu na popratne poveznice. Industrija "clickbaitova" i dalje ubrzano raste bez obzira na zabrinutost vezanu uz širenje dezinformacija koju su iskazali znanstvenici. Cilj rada je klasificirati "clickbait" naslove koristeći različite modele i reprezentacije teksta. Više skupova podataka je korišteno u istraživanju, uključujući skup podataka koji su prikupili Chakraborty et al. i skup podataka Webis-Clickbait-17. Nadalje, ova dva skupa podataka spojena su i klasifikacija je također urađena na takvom skupu podataka. Koriste se modeli kao što su naivni Bayesov klasifikator, strojevi s potpornim vektorima, slučajna šuma, linearni perceptron, ćelija s dugoročnom memorijom (LSTM) i upravljačka rekurentna jedinica (GRU). Korištene su i uspoređene tri tehnike reprezentacije teksta, TF-IDF, word2vec i FastText. Rezultati pokazuju da FastText reprezentacija ima najbolje performanse. Što se tiče klasifikacijskih performansi, strojevi s potpornim vektorima postižu najbolje rezultate klasifikacije, s F1-rezultatom od 0,8797 na spojenom skupu podataka. Studija također istražuje sličnost naslova i tekstova članaka i pokazalo se da naslovi koji nisu "clickbaitovi" bolje predstavljaju sadržaj. Ovo istraživanje pruža uvid u modele klasifikacije "clickbaitova", a s ciljem poboljšanja korisničkog iskustva i borbe protiv dezinformacija.

Ključne riječi: "clickbait", strojno učenje, duboko učenje, klasifikacijski modeli, reprezentacija teksta, sličnost naslova i teskta, dezinformacije.

Contents

1 Introduction

Today, digital media has displaced print media, and there are more news sites than ever that offer a wide range of information [1]. Various techniques are being utilized in order to increase readability – some positive, some negative. One of them is *clickbaiting*. The term *clickbait* describes a particular type of web content advertisement created to persuade users to click a related link [2]. According to the Oxford Learner's Dictionaries[1], clickbait is described as:

> "*Material put on the internet in order to attract attention and encourage visitors to click on a link to a particular web page.*"[2]

Advertisers that write clickbait titles use sensationalist or misleading tactics. A typical example of a clickbait title would be: "You Won't Believe What This Celebrity Did on Live TV!". We do not know who that celebrity is and what he/she has done on live TV, and thus, it creates some kind of curiosity and intrigue inside us. As G. Loewenstein explained in his article about the information gap theory of curiosity [3], there is a gap between what people know and what they want to know. Namely, curiosity generally takes two main steps: first, a situation reveals a painful knowledge gap (the headline), and then we feel the urge to fill this gap and ease that pain (the click). Despite the fact that media analysts and commentators continuously depict clickbait content negatively, the market for this type of content has been rapidly expanding and is now reaching an increasing number of individuals worldwide [4]. The primary purpose of a clickbait title is profit, as many pages earn money from clicks – how many times a particular article or post is clicked [5]. Additionally, news websites display adverts and make money from them every time a user clicks and opens an item [6].

Research on clickbait detection has been ongoing for the past ten years. Potthast et al. [2] constructed one of the first clickbait datasets (a corpus of Twitter posts). They conducted research and created a clickbait classification model based on numerous characteristics of clickbait headlines. In his work [6], A. Agrawal employed deep learning techniques and distinguished clickbait headlines with a significant degree of accuracy. Anand et al. [7] used neural networks to detect clickbait. Their F1 score was 0.98 using bidirectional recurrent neural networks. Kumar et al. [8] developed a multi-strategy approach using neural networks to classify clickbait titles. Their research was conducted on the Webis-Clickbait-17 dataset and achieved F1 scores ranging from 0.39 to 0.65.

[1] `https://www.oxfordlearnersdictionaries.com/definition/english/clickbait`

[2] The definition is prone to slight changes over time.

Fighting clickbait is essential because news sites are making sly and cunning financial gains from it. There are existing initiatives to stop clickbait and alert people when it could be there. In particular, Chakraborty et al. [5] created the 'Stop Clickbait' browser plugin, which alerts users to the presence of clickbait on various webpages.

This master thesis aims to employ different machine learning and deep learning methods for classifying clickbait titles. Machine learning methods include the naive Bayes classifier, support vector machines, random forest and linear perceptron. Deep learning methods used in this work consist of two neural network architectures: LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit). The classification task will be performed on three datasets, and their performances will be compared using three different text representation techniques – TF-IDF, word2vec and FastText. Additionally, title-text similarity will be examined using DSSM (Deep Semantic Similarity Model) to determine whether clickbait titles differ more from their content than non-clickbait titles. All the work is done on the Google Colab platform[3] in Python 3 programming language. Google also provides a GPU for hardware acceleration, and NVIDIA T4 Tensor Core GPU was used for the training of neural networks, which is provided free of charge. The datasets used in this work are saved in CSV (comma-separated values) files and were loaded into the scripts using the *pandas*[4] library.

In Section 2, the datasets used in this work are described. Next, Section 3 elaborates on machine learning and deep learning is covered in Section 4. Title-text similarity is examined in Section 5. In the last Section, some concluding remarks are provided.

[3] `https://colab.research.google.com/`

[4] `https://pandas.pydata.org/`

2 Datasets

The first dataset used in this work is the dataset collected and prepared by Chakraborty et al. [5]. This dataset initially consisted of 7,500 clickbait and 7,500 non-clickbait titles in English language. The non-clickbait titles were extracted from Wikinews, where authors must follow strict rules for writing an article. Due to that, it is considered to be a good source for non-clickbait titles. The clickbait titles were collected from the following portals: BuzzFeed, Upworthy, ViralNova, Scoopwhoop and ViralStories. Later, the dataset was expanded to include 16,001 non-clickbait titles and 15,999 clickbait titles. The dataset is balanced because 50% of the instances belong to one class (clickbait) and 50% belong to another (non-clickbait). In order to label the dataset, six volunteers were enlisted, with at least three of them labeling each item. With a Fleiss' κ[5] of 0.79, a fair level of inter-annotator agreement was achieved. They performed an analysis and a comparison between clickbait and non-clickbait titles and showed that the average length of non-clickbait titles is seven words and the average length of clickbait titles is ten words. This is due to the fact that clickbait titles tend to have more connecting function words and stop words. Also, the average word length is shorter in the clickbait titles because of the aforementioned larger amount of stop words.

The second dataset is the Webis Clickbait Corpus 2017 (Webis-Clickbait-17) [10], created for a clickbait challenge in 2017 [11]. Potthast et al. crawled 459,541 English tweets from 27 major US news publishers[6] posted between December 2016 and April 2017. The objective was to draw at least 30,000 but no more than 40,000 tweets due to financial constraints. Consequently, they ended up with a total of 38,517 tweets. Alongside the titles, the dataset includes other metadata, the most interesting of which are the entire texts of the articles. The dataset was annotated by five annotators using Amazon Mechanical Turk[7]. This dataset is a trickier one since it has many more cases where *clickbaitiness* is debatable. Also, it is not balanced (75.92% of the instances are non-clickbait). That will result in poorer performance of our classifying tasks.

[5] As defined in [9], "Fleiss' κ is a measure of inter-rater agreement used to determine the level of agreement between two or more raters (also known as 'judges' or 'observers') when the method of assessment, known as the response variable, is measured on a categorical scale."

[6] ABC News, BBC World, Billboard, Bleacher Report, Breitbart News, Business, Business Insider, BuzzFeed, CBS News, CNN, Complex, ESPN, Forbes, Fox News, Guardian, HuffPost, Independent, Indiatimes, MailOnline, Mashable, NBC News, NY Times, Telegraph, USA Today, Washington Post, WSJ and Yahoo.

[7] `https://www.mturk.com/`

Lastly, for the purpose of this work, these two datasets were merged in a way that the maximum number of titles are included while preserving the balance, so 18,552 instances from the second dataset were added to the first one. That resulted in a dataset with 50,550 instances that has a balanced distribution of 25,275 non-clickbait titles and 25,275 clickbait titles. A sample of the dataset can be seen in Figure 1.

	title	clickbait
0	Should I Get Bings	1
1	Which TV Female Friend Group Do You Belong In	1
2	The New "Star Wars: The Force Awakens" Trailer...	1
3	This Vine Of New York On "Celebrity Big Brothe...	1
4	A Couple Did A Stunning Photo Shoot With Their...	1
...	...	...
50545	5 Dead, 7 Injured After Tornadoes in Alabama a...	0
50546	Seattle Seahawks Richard Sherman Says 'Karma' ...	0
50547	Older Viewers and Conservatives Are Watching L...	0
50548	A Superior Chicken Soup	1
50549	Panama Papers: Europol links 3,500 names to su...	0

50550 rows × 2 columns

Figure 1: Sample of the merged dataset: title ID, title content, class (1 – clickbait, 0 – non clickbait).

3 Machine Learning

Machine learning is a subfield of artificial intelligence and a branch of computer science that tries to enable computers to "learn" without being explicitly programmed. [12]. Machine learning is employed to learn models, using algorithms, capable to analyse and draw inferences from patterns in data. Data is the foundation of any machine learning task, including images, text, numbers, time-series, audio, sales records, and more. The general rule about data amount is – the more data, the merrier. Supervised learning and unsupervised learning are two categories of machine learning algorithms. In supervised learning, output or target values are known, and the task is to predict these target values in the test set where teaching is done on the training set. On the other hand, unsupervised learning tries to discover structure and regularities in the data [13]. Regarding that, it is crucial to choose the appropriate machine learning algorithm for a particular task. Since the data in this work has been labeled, supervised learning is employed. There are two labels: 0 for "non-clickbait" and 1 for "clickbait".

In this work, three text representation methods are used for the training of machine learning models: TF-IDF [14], word2vec [15], and FastText [16] representation. With each of these representations, the following machine learning methods were trained to classify clickbait titles: the naive Bayes classifier (NB)[8], SVM (Support Vector Machine)[9], random forest (RF)[10] and linear perceptron[11]. The NB, SVM and RF methods are described in [17] and linear perceptron is described in [18]. All machine learning methods implementations used in this thesis are from the *scikit-learn*[12] library for Python. For the naive Bayes classifier, Gaussian NB classifier was used (except for the TF-IDF representation where multinomial NB classifier was used because TF-IDF violates the underlying assumption of the Gaussian distribution).

All machine learning methods: NB, SVM, RF and Perceptron are trained on three datasets:

- The first dataset (collected by Chakraborty et al. with 32,000 instances),
- The second dataset (Webis-Clickbait-17 with 38,517 instances),
- The third dataset (merged dataset with 50,550 instances).

[8] `https://scikit-learn.org/stable/modules/naive_bayes.html`
[9] `https://scikit-learn.org/stable/modules/svm.html`
[10] `https://scikit-learn.org/stable/modules/ensemble.html#forest`
[11] `https://scikit-learn.org/stable/modules/linear_model.html#perceptron`
[12] `https://scikit-learn.org/stable/index.html`

For each method, a split of 70:30 ratio is used. Accuracy, precision, recall and F1 score are used as metrics for evaluating the models [14]. A brief explanation of evaluation metrics is given below, but before that, the abbreviations used in the equations are established:

- **TP** is true positive (a test result that the model correctly classified as clickbait),
- **TN** is true negative (a test result that the model correctly classified as non-clickbait),
- **FP** is false positive (a test result that the model wrongly classified as clickbait),
- **FN** is false negative (a test result that the model wrongly classified as non-clickbait).

Accuracy: total correctly classified examples divided by the total number of classified examples. Formula:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{1}$$

Precision: the actual correct prediction divided by the total prediction made by the model. Formula:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2}$$

Recall: the number of true positives divided by the total number of true positives and false negatives. Formula:

$$\text{recall} = \frac{TP}{TP + FN}. \tag{3}$$

F1 score: a weighted average of precision (2) and recall (3). Formula:

$$\text{f}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{4}$$

A usual way of representing TP, TN, FP and FN is a confusion matrix [19]. In a binary classification task, the matrix has two rows and two columns (Figure 2).

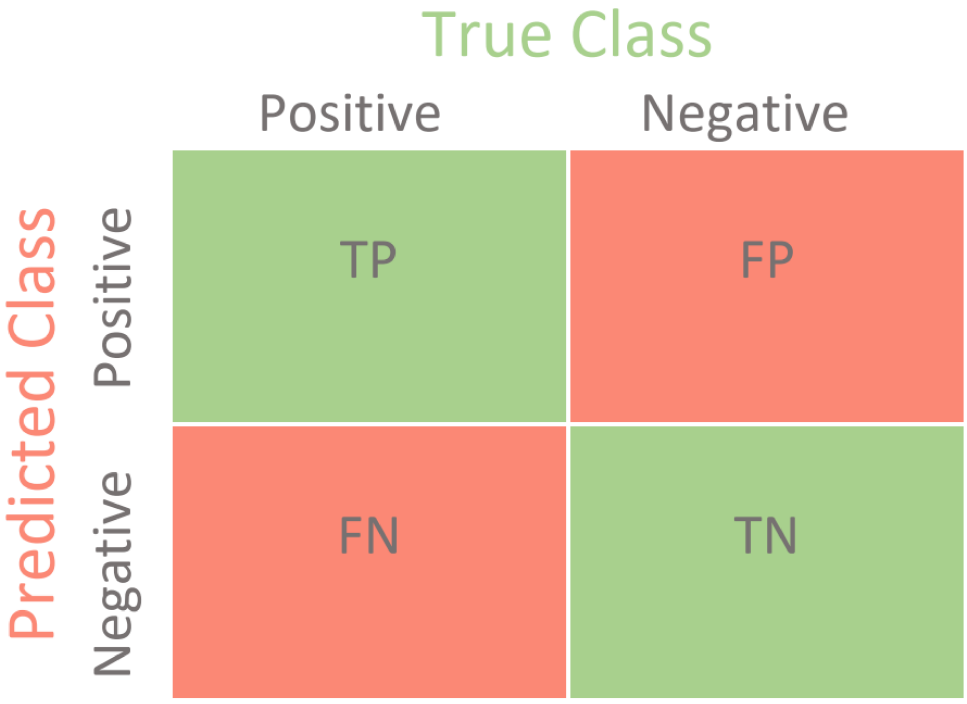


Figure 2: Confusion matrix on a binary classification task [20]. Rows represent the predicted class and columns represent the true class.

In the following sections, three text representations will be explained and for each one of them, three performance tables (for every dataset) will be displayed.

3.1 TF-IDF Representation

An important aspect of NLP (Natural Language Processing) is transforming the words into a numerical representation (a vector). Vectors in low-dimensional continuous space are called word embeddings [21]. The vectors try to capture various characteristics of that word with regard to the overall text. As described in [14] and [22], TF-IDF (Term Frequency - Inverse Document Frequency) is a text representation that reflects how important a word is to a document in a collection or corpus. TF-IDF is the product of **tf** (term frequency) and **idf** (inverse document frequency).

Term frequency, $\mathrm{tf}(t, d)$, is the relative frequency of term t within document d:

$$\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{5}$$

where $f_{t,d}$ is the raw count of a term in a document. Raw term frequency has one drawback – all terms are considered equally important, but in practice, this is not the case.

Document frequency, $\text{df}(t, D)$, is the number of documents in the collection that contain the term t. **Inverse document frequency**, $\text{idf}(t, D)$, tries to reduce the tf weight of a term by a factor that grows with its collection frequency. Inverse document frequency is basically a measure of how much information the word provides:

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}, \tag{6}$$

where N is the total number of documents in the corpus with the size $|D|$. $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears. Lastly, TF-IDF, $\text{tfidf}(t, d, D)$, is the product of $\text{tf}(t, d)$ (5) and $\text{idf}(t, D)$ (6):

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D). \tag{7}$$

For representing the texts with TF-IDF, TF-IDF vectorizer from the *scikit-learn*[13] library is used. Four algorithms were tested on three datasets using TF-IDF representation. Tables 1, 2 and 3 show performances on three used datasets. Also, time needed to perform the algorithm is also shown in the tables[14]. Bold values represent the highest value of the metric. Table 1 displays the performance on the first (Chakraborty et al.) dataset. The naive Bayes classifier showed the best results as it tops the performance in three out of four observed metrics.

Table 1: TF-IDF performance on the Chakraborty et al. dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	**0.9569**	0.9549	0.9015	0.9235
Precision	0.9498	**0.9681**	0.8598	0.9212
Recall	**0.9649**	0.9407	0.9593	0.9263
F1 score	**0.9573**	0.9542	0.9068	0.9237
Time (s)	0.05	82.41	20.20	**0.03**

The performance was degraded on the second dataset (Webis-Clickbait-17). Precisely, recall exhibit the highest decline. This happened because the dataset is unbalanced (way less clickbait than non-clickbait titles) and has more "hard titles"[15].

[13] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[14] The algorithms were executed five times with the same parameters. To measure the time they take, *timeit* (`https://docs.python.org/3/library/timeit.html`) library is utilized and the average time is displayed in the tables.

[15] The term "hard titles" refers to titles that possess a questionable level of *clickbaitiness*.

Overall, accuracy and precision are somewhat satisfactory as they are around 70%. It is hard to say which model is the best. Performance can be seen in Table 2.

Table 2: TF-IDF performance on the Webis-Clickbait-17 dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.7801	**0.7896**	0.7759	0.7038
Precision	0.7366	**0.7506**	0.5793	0.4084
Recall	0.1731	0.2242	0.3438	**0.4397**
F1 score	0.2804	0.3453	**0.4315**	0.4234
Time (s)	**0.04**	349.48	74.28	0.10

TF-IDF with the third (merged) dataset showed results that lie in between the first two and they are pretty good overall. SVM achieved the best results while taking by far the most time to complete. Table 3 illustrates this.

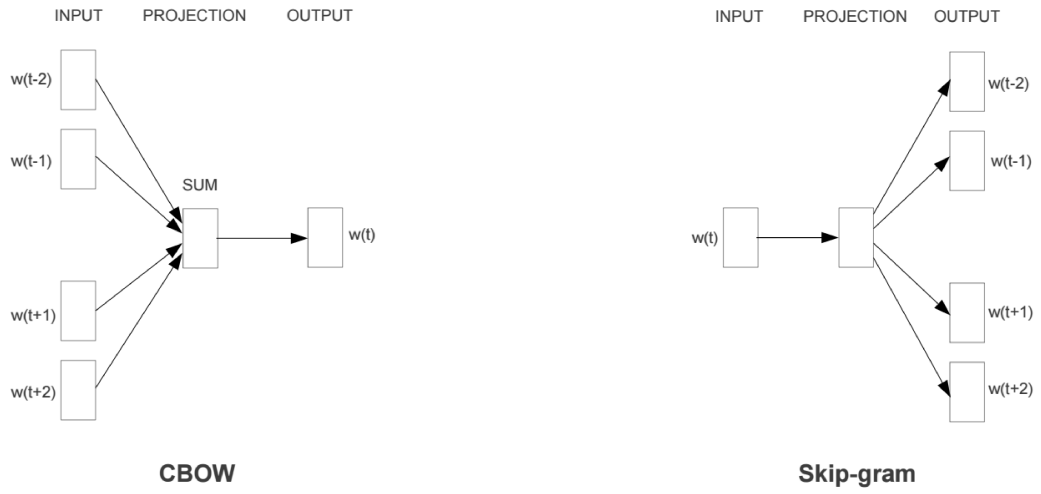
Table 3: TF-IDF performance on the merged dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.8404	**0.8432**	0.8106	0.7824
Precision	0.8481	**0.8609**	0.7945	0.7725
Recall	0.8312	0.8206	**0.8404**	0.8036
F1 score	0.8396	**0.8403**	0.8168	0.7877
Time (s)	0.23	319.86	92.09	**0.10**

3.2 Word2vec Representation

Word2vec is an effective way of representing words because it can make strong estimates of words' meanings based on their occurrences in the text [15]. It follows the NNLM (Neural Net Language Model) architecture proposed by Bengio et al. [23]. Mikolov et al. [15] introduced two new model architectures for learning distributed representations of words with the goal of minimizing computational complexity as most of the complexity is caused by the non-linear hidden layer in the NNLM. The first architecture is a bag-of-words (BOW) model where the hidden layer is removed and the projection layer is shared for all the words. In essence, all words get projected into the same position. Furthermore, they used a continuous distributed representation of the context so the architecture they proposed is called continuous bag-of-words (CBOW) [15] (Figure 3a).

The second architecture, called skip-gram, is similar to CBOW, but it tries to maximize the classification of a word based on another word in the same sentence. Each current word is used as an input to a log-linear classifier with a continuous projection layer. The goal is to predict words within a certain range before and after the current word (Figure 3b). In other words, CBOW is an architecture that predicts a target word from a list of context words, whereas the skip-gram, the opposite of the CBOW algorithm, tries to predict the context word for a given target word [15].



(a) CBOW architecture

(b) Skip-gram architecture

Figure 3: Architectures behind word2vec [15].

Figure 4 displays an example of vectors obtained using the word2vec model and their semantic and syntactic relationship. It can be seen that word pairs *queen–king* and *woman–man* are semantically close, while word pairs *big–biggest* and *small–smallest* are syntactically close.

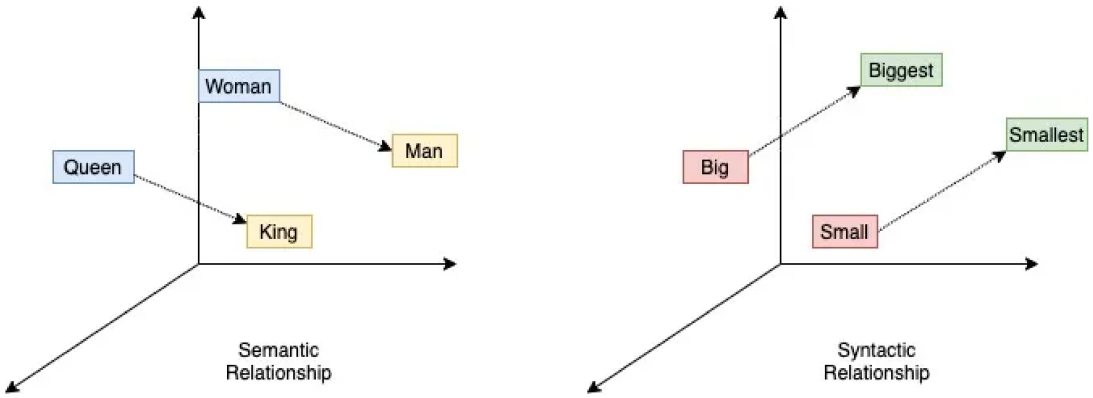


Figure 4: Trained word2vec vectors with semantic and syntactic relationship [24].

Implementation of word2vec representation is done using the *gensim*[16] library. Using word2vec representation on the Chakraborty et al. dataset showed good performance which can be seen in Table 4. As the classifier, random forest achieved the best performance since three of the four best metrics were achieved with it. However, random forest takes the longest time to complete.

Table 4: Word2vec performance on the Chakraborty et al. dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.8856	0.9208	**0.9332**	0.9277
Precision	0.9229	0.9351	0.9362	**0.9487**
Recall	0.8415	0.9042	**0.9297**	0.9042
F1 score	0.8803	0.9194	**0.9329**	0.9259
Time (s)	**0.06**	6.92	27.16	0.22

Word2vec performed quite poorly on the Webis-Clickbait-17 dataset. The best method is the naive Bayes since it has the highest F1 score among all tested methods and is the fastest one. In this example, linear perceptron failed quite a bit, as can be seen in the reported F1 score of only 0.0550. The performance of four classifiers with word2vec representation on the Webis-Clickbait-17 dataset is reported in Table 5.

Table 5: Word2vec performance on the Webis-Clickbait-17 dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.7212	0.7820	**0.7893**	0.7561
Precision	0.4497	**0.7401**	0.6423	0.6667
Recall	**0.5659**	0.1833	0.3347	0.0287
F1 score	**0.5012**	0.2938	0.4401	0.0550
Time (s)	**0.06**	86.41	46.58	0.12

While using the merged dataset, performance fell midway between the first two datasets. The random forest classifier was the best again and achieved the highest values in three of the four metrics. The performance can be seen in Table 6.

[16] `https://radimrehurek.com/gensim/models/word2vec.html`

Table 6: Word2vec performance on the merged dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.8062	0.8616	**0.8690**	0.8059
Precision	0.8955	0.9009	0.8933	**0.9427**
Recall	0.6954	0.8141	**0.8396**	0.6535
F1 score	0.7829	0.8553	**0.8656**	0.7719
Time (s)	**0.12**	129.92	58.02	0.53

3.3 FastText Representation

The third representation used in the thesis is FastText. FastText, as proposed by Bojanowski et al. [16], learns representations for character n-grams and represents words as the sum of the n-gram vectors. It is an extension to the continuous skip-gram model proposed by Mikolov et al. in [25]. FastText frames the problem of predicting context words as a set of independent binary classification tasks. For a chosen context position c, using the binary logistic loss, they propose the following negative log-likelihood [16]:

$$\log\left(1 + e^{-s(w_t, w_c)}\right) + \sum_{n \in \mathcal{N}_{t,c}} \log\left(1 + e^{s(w_t, n)}\right), \tag{8}$$

where $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary. Essentially, it is the skip-gram model with negative sampling. In FastText, each word w is represented as a bag of character n-gram. The boundary symbols ($<$ and $>$) are used to distinguish prefixes and suffixes. Furthermore, a different scoring function s that takes the internal structure of words into account is proposed. This scoring function is the sum of the vector representations of its n-grams [16]:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c, \tag{9}$$

where G is the size of a dictionary of n-grams and $\mathcal{G}_w \subset \{1, ..., G\}$ is the set of n-grams appearing in w. $\mathbf{z}_g$ is a vector representation of each n-gram g. By enabling the sharing of representations across words, this technique enables the learning of reliable representations for rare words also. FastText trains quickly and does not need any supervision or preprocessing. Additionally, it performs better than the baseline models that ignore subword information [16].

In the implementation, a module called *fasttext*[17] is used for representing the texts using FastText representation. Using FastText on the Chakraborty et al. dataset showed again pretty good results with the best-performing model being SVM. Overall, this is the best result achieved, with the F1 score of 0.9695 and all the other metrics hovering around 0.97, suggesting that FastText is the preferred representation over TF-IDF and word2vec on this dataset. Table 7 displays the performance of NB, SVM, RF and linear perceptron on Chakraborty et al. dataset using FastText representation.

Table 7: FastText performance on the Chakraborty et al. dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.8941	**0.9693**	0.9461	0.9600
Precision	0.9022	**0.9727**	0.9584	0.9510
Recall	0.8868	0.9665	0.9341	**0.9710**
F1 score	0.8944	**0.9695**	0.9461	0.9609
Time (s)	**0.21**	17.61	69.07	0.24

With the second Webis-Clickbait-17 dataset, performance again was pretty poor. The best-performing classification model is hard to select as not a single model achieved better performance on the Webis-Clickbait-17 dataset. Performance of NB, SVM, RF and linear perceptron on Webis-Clickbait-17 dataset using FastText representation is displayed in Table 8.

Table 8: FastText performance on the Webis-Clickbait-17 dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.6954	**0.7977**	0.7948	0.3660
Precision	0.4127	**0.7607**	0.7205	0.2693
Recall	**0.6079**	0.2407	0.2504	0.9436
F1 score	**0.4916**	0.3657	0.3716	0.4190
Time (s)	0.89	400.58	93.46	**0.34**

Finally, while using the merged dataset, pretty good performance was achieved, especially with the SVM classifier. A downside is that SVM takes, on average, around six minutes to complete, while linear perceptron takes under a second at the little expense of other metrics. Performance of NB, SVM, RF and linear perceptron on the merged dataset using FastText representation is reported in Table 9.

[17] `https://fasttext.cc/docs/en/python-module.html`

Table 9: FastText performance on the merged dataset.

Metric	NB	SVM	RF	Perceptron
Accuracy	0.8148	**0.8816**	0.8589	0.8506
Precision	0.8593	**0.8982**	0.8918	0.8274
Recall	0.7550	**0.8919**	0.8185	0.8881
F1 score	0.8038	**0.8797**	0.8536	0.8566
Time (s)	0.63	350.83	118.41	**0.60**

3.4 Discussion of the Results

Among the three text representations used in this work, TF-IDF performed as the fastest method for representing words as vectors. It takes 0.33, 3.42 and 9.40 seconds to complete on the Chakraborty et al., Webis-Clickbait-17 and merged dataset, respectively. FastText takes 6.04, 13.06 and 19.11 seconds whereas word2vec is the slowest one as it takes 12.89, 31.40 and 39.30 seconds. Results can be seen in Table 10. The naive Bayes classifier is the fastest algorithm. Linear perceptron requires a little more time, followed by random forest, while SVM takes the longest. Regarding the performance, FastText showed the highest average performance metrics excelling over TD-IDF and word2vec representations.

Table 10: Time to perform each text representation.

Dataset	TF-IDF	Word2vec	FastText
Chakraborty et al.	**0.33 s**	12.89 s	6.04 s
Webis-Clickbait-17	**3.42 s**	31.40 s	13.06 s
Merged	**9.40 s**	39.30 s	19.11 s

The merged dataset is the most representative one since it is a merge of two modalities of smaller datasets – one balanced and the other unbalanced with "borderline" cases. Therefore, models trained on the merged dataset are the most generalized ones. Again, SVM using FastText representation is the superior model in this setup, but it has a considerable downside – a long time to execute. When considering the execution time, linear perceptron using the FastText representation is the preferred model.

4 Deep Learning

Deep learning is a subset of machine learning. The main differentiation is that deep neural networks have more neurons, layers and connections [26]. Deep learning algorithms emerged in an attempt to make traditional machine learning techniques more efficient. Thus, allowing us to express more complex models, optimize more parameters and automate feature extraction [27]. Over time, six main deep learning architectures emerged:

- Feedforward neural network (FNN) (also known as Multi-Layer Perceptron (MLP)) [23],
- Convolutional Neural Network (CNN) [23],
- Recurrent Neural Network (RNN) [28],
- Generative Adversarial Network (GAN) [29],
- Autoencoders [23],
- Transformers [30][18].

For the NLP tasks, it has been shown that variants of RNN – Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are appropriate for long sequenced data (i.e., text), so they are elaborated in the continuation. The explanations of FNN, CNN, GAN, autoencoders and transformers are omitted since they are not used in the experiments performed in this thesis.

In recurrent neural networks, connections between nodes can form a cycle that allows some nodes' output to influence other nodes' input. RNNs are capable of processing input sequences with varying lengths by using their internal state (memory) [31]. This makes them suitable for various NLP tasks. Therefore, in this work, two RNN architectures are used: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) which will be explained in the following sections.

4.1 Long Short-Term Memory

Long short-term memory architecture was introduced in 1997 by S. Hochreiter and J. Schmidhuber in [32]. The memory cell and the gates (including the forget and the input gate) are the two essential parts of an LSTM. The input gates and forget gates

[18]Although there are many more types of neural networks and new types emerge almost every week, some of them may or may not fall into one of these six categories. These three are the most popular ones and they can be further divided into subcategories.

modify the information stored in the memory cell. The memory cell's contents will not change from one time step to the next, supposing that both of these gates are closed. Gradients can flow across multiple time steps because the gating structure enables information to be preserved across a large number of computations. Because of this, the LSTM model is able to avoid the vanishing gradient issue that affects the majority of recurrent neural network models. They offer better update equations and hence better perform backpropagation learning steps [32]. Some of the main tasks on long sequenced data they successfully complete include text generation, time series prediction, speech recognition, handwriting recognition and polyphonic music modeling, to name a few.

The structure of the LSTM network used in this work is displayed in Figure 5. Besides the input layer, the network consists of the following layers:

- Embedding layer with 128 neurons,
- LSTM layer with 128 neurons with a hyperbolic tangent (tanh) activation function,
- Output dense layer with one neuron with the sigmoid activation function.

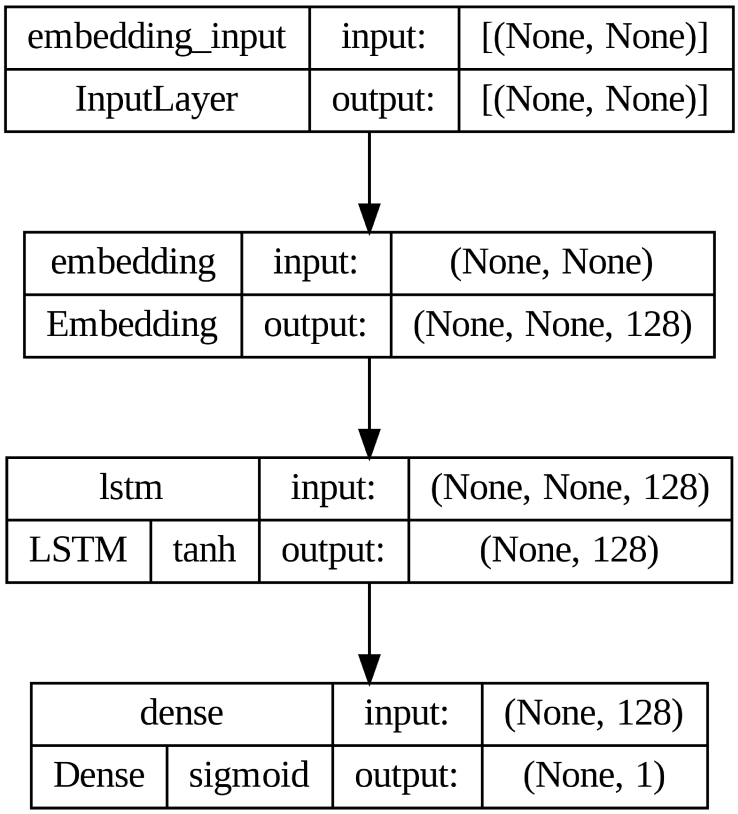


Figure 5: Structure of the LSTM network used in this work.

The sigmoid activation function [27] (labeled by $\Phi(x)$) is a non-linear activation function that looks like the letter S. It is calculated using the following formula:

$$\Phi(x) = \frac{1}{1 + e^{-x}}. \tag{10}$$

Its main advantage is that it exists between 0 and 1. Therefore, it is the ideal choice in models where the probability has to be emitted on the output.

The hyperbolic tangent activation function (tanh) [27] is also a non-linear activation function but has a range from -1 to 1. It is an s-shaped function with the main advantage that the negative inputs will be mapped strongly negative, and the zero inputs will be mapped near zero in the tanh graph. It can be calculated using the following formula:

$$\Phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{11}$$

Both the sigmoid and hyperbolic tangent activation functions are differentiable and monotonic while their derivatives are not monotonic. Figure 6 shows the differences between the two mentioned activation functions.

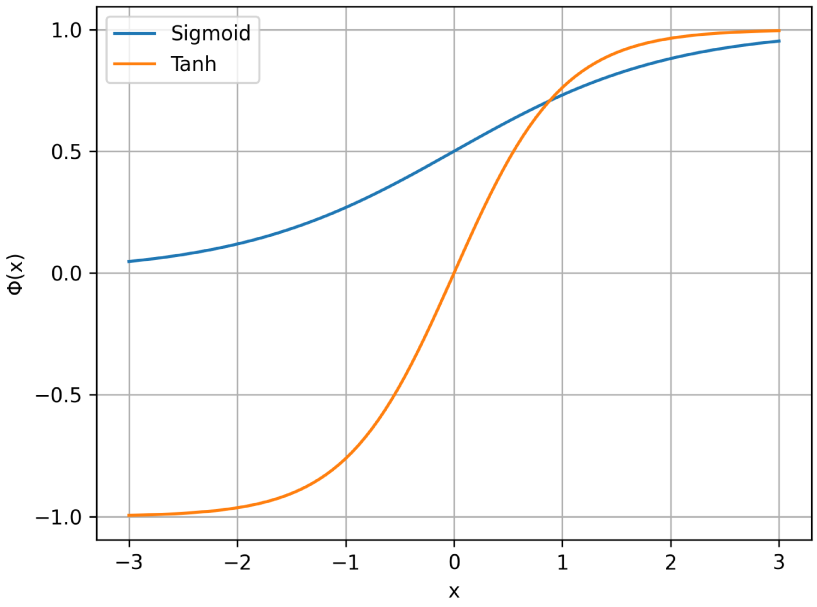


Figure 6: Differences between the sigmoid and hyperbolic tangent activation functions.

The LSTM network was trained across 20 epochs on all three datasets. Binary cross-entropy [26] is employed for the loss function together with the Adam [33] optimizer and a batch size of 64.

Binary cross-entropy (also known as log loss or logistic loss) is a loss function commonly used in binary classification tasks [26]. It measures the dissimilarity between the predicted probabilities and the true labels. Binary cross-entropy (labeled as $L(y, \hat{y})$) is calculated using the following formula:

$$L(y, \hat{y}) = -\frac{1}{N} \cdot \sum_{i=1}^{N} y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}), \tag{12}$$

where y is the true label and $\hat{y}$ is the predicted probability.

Figures 7, 8 and 9 show accuracies and losses during the training process on each of the datasets.

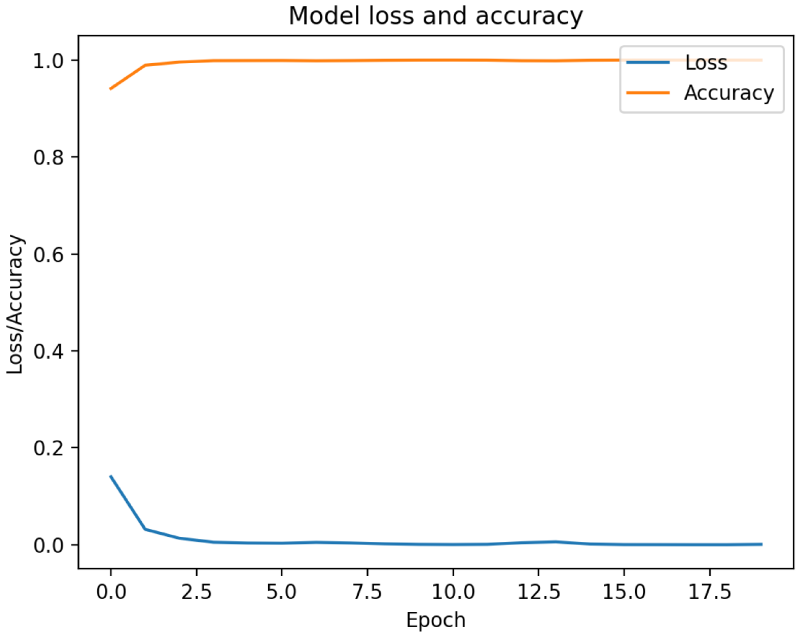


Figure 7: Accuracy and loss during the training of the LSTM network using the Chakraborty et al. dataset.

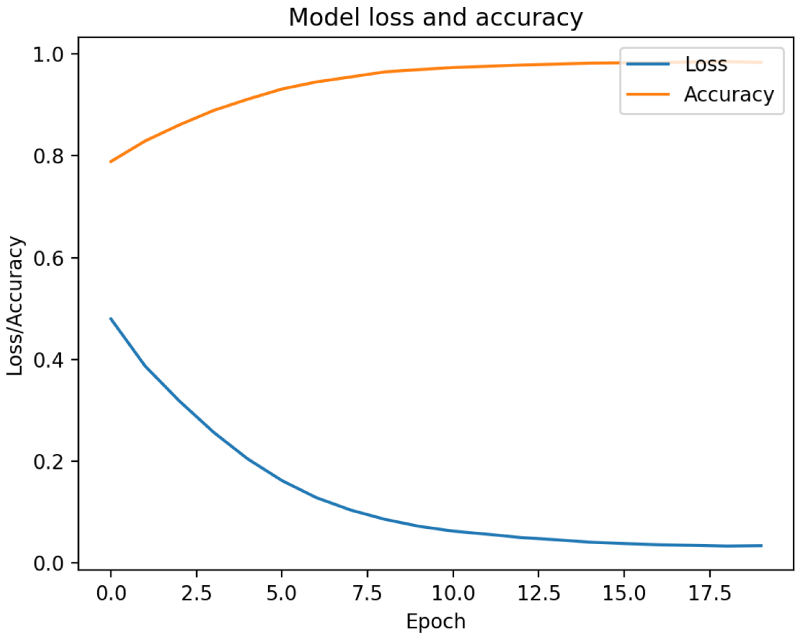


Figure 8: Accuracy and loss during the training of the LSTM network using the Webis-Clickbait-17 dataset.

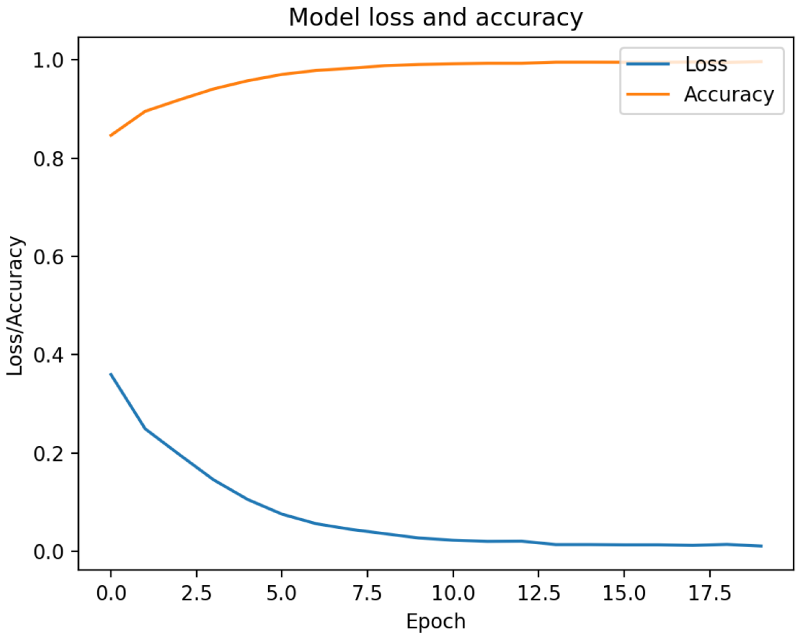


Figure 9: Accuracy and loss during the training of the LSTM network using the merged dataset.

4.2 Gated Recurrent Unit

Gated recurrent units were introduced in 2014 by Cho et al. in [34]. The design of a gated recurrent unit is comparable to an LSTM. Similar to the forget/input gates in the LSTM unit, the GRU contains a reset gate and an update gate. The primary distinction is that the GRU uses leaky integration to completely reveal its memory content. Although it is thought to be easier to compute and implement, it was inspired by the LSTM [34]. Figure 10 shows the structure of the GRU network employed in this work. The GRU network consists of an input layer alongside:

- Embedding layer consisting of 100 neurons,
- GRU layer consisting of 32 neurons with a hyperbolic tangent activation function,
- Output dense layer with one neuron with the sigmoid activation function.

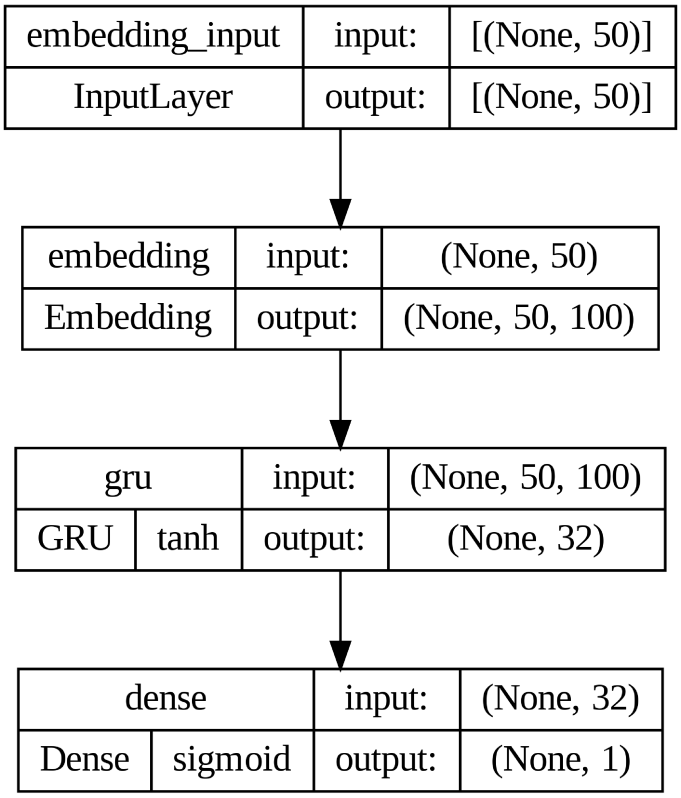


Figure 10: Structure of the GRU network used in this work.

Similarly to the LSTM network, the GRU network was trained across 20 epochs on all three datasets. The loss function is calculated using binary cross-entropy. Adam optimizer and a batch size of 64 were used. Figures 11, 12 and 13 show accuracies and losses during the training process on each of the datasets.

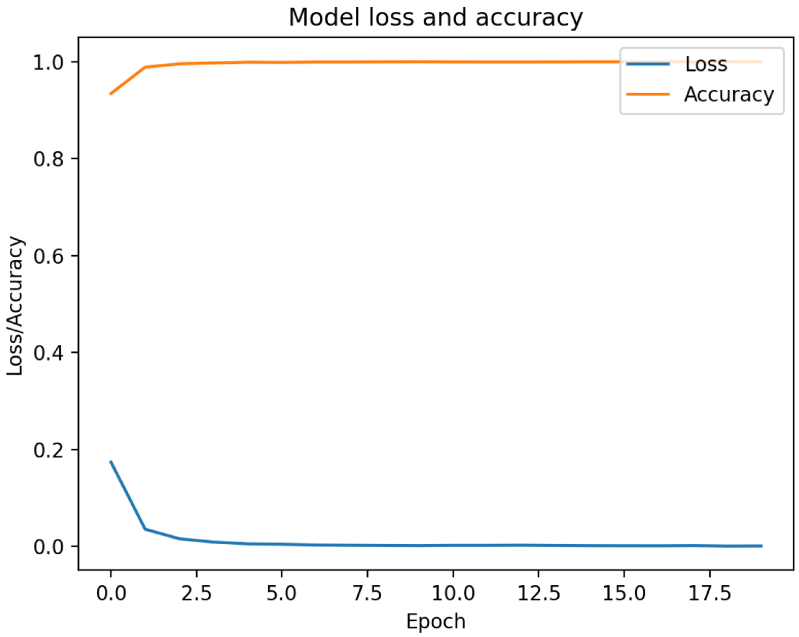


Figure 11: Accuracy and loss during the training of the GRU network using the Chakraborty et al. dataset.

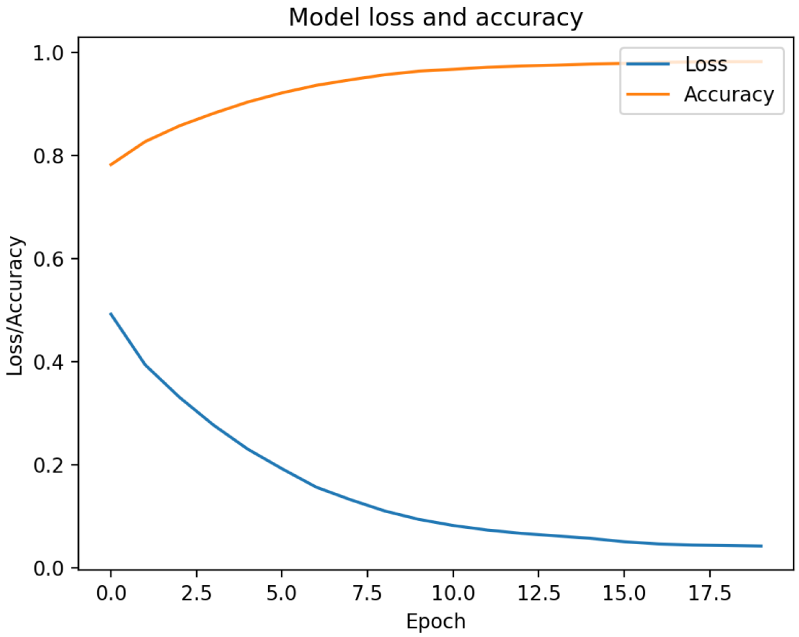


Figure 12: Accuracy and loss during the training of the GRU network using the Webis-Clickbait-17 dataset.

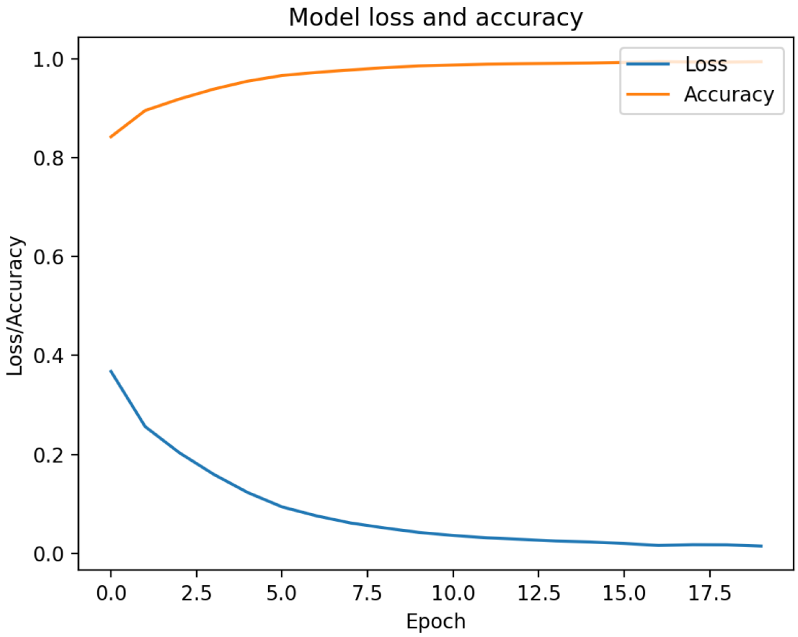


Figure 13: Accuracy and loss during the training of the GRU network using the merged dataset.

4.3 LSTM and GRU Results

This section compares the performances of LSTM and GRU using the usual evaluation metrics (i.e., accuracy, precision, recall, F1 score and time). Table 11 shows the performance of LSTM and GRU on the Chakraborty et al. dataset. Very similar performance between LSTM and GRU can be noticed, with GRU slightly outperforming LSTM.

Table 11: LSTM and GRU performance on the Chakraborty et al. dataset.

Metric	LSTM	GRU
Accuracy	0.9647	**0.9714**
Precision	**0.9883**	0.9655
Recall	0.9413	**0.9784**
F1 score	0.9643	**0.9719**
Time per epoch (s)	40.35	**37.85**

On the Webis-Clickbait-17 dataset, the performance results are similar but, as expected, lower due to the nature of this dataset. In this case, GRU is again a slightly better-performing model. Table 12 shows LSTM and GRU performance on the Webis-Clickbait-17 dataset.

Table 12: LSTM and GRU performance on the Webis-Clickbait-17 dataset.

Metric	LSTM	GRU
Accuracy	**0.7523**	0.7328
Precision	**0.4864**	0.4470
Recall	0.3950	**0.4339**
F1 score	0.4359	**0.4404**
Time per epoch (s)	109.20	**99.60**

Finally, using the merged dataset, performances are in between the first two. Here, the LSTM is a better-performing model. The LSTM and GRU performance on the merged dataset is in Table 13.

Table 13: LSTM and GRU performance on the merged dataset.

Metric	LSTM	GRU
Accuracy	**0.8364**	0.8318
Precision	**0.8417**	0.8351
Recall	**0.8306**	0.8290
F1 score	**0.8361**	0.8321
Time per epoch (s)	130.25	**122.40**

5 Title-Text Similarity

5.1 Cosine Similarity

Cosine similarity measures the similarity between two vectors in space [35]. The following formula is used to calculate the cosine similarity between two documents d_1 and d_2 and their vector representations $\vec{V}(d_1)$ and $\vec{V}(d_2)$:

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}, \tag{13}$$

where the numerator is the dot product of the vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$ and the denominator is the product of their Euclidean lengths [14]. The values of the cosine similarity range between -1 and 1.

Only the Webis-Clickbait-17 dataset contains both the titles and the texts of the articles. Therefore, the cosine similarity between the corresponding titles and texts can be calculated. The cosine similarity was calculated using the *spacy*[19] library and the *en_core_web_md*[20] trained pipeline for the English language. The average cosine similarity between non-clickbait titles and corresponding texts is 0.6024, and the average cosine similarity for clickbait titles and corresponding texts is 0.5411 (Table 14). This indicates that the non-clickbait titles better represent the corresponding texts hence they are more similar. This result was expected.

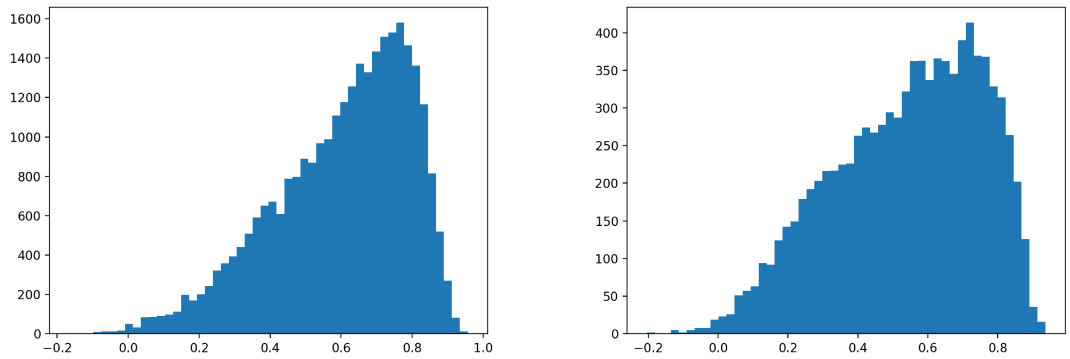
Table 14: Average cosine similarity between titles and texts on the Webis-Clickbait-17 dataset.

	Non-clickbait	Clickbait
Cosine similarity	0.6024	0.5411

In Figure 14, the histograms of cosine similarities can be seen for non-clickbait titles and texts (Figure 14a) and clickbait titles and texts (Figure 14b).

[19] `https://spacy.io/`

[20] `https://spacy.io/models/en/`



(a) Non-clickbait titles and texts

(b) Clickbait titles and texts

Figure 14: Histogram of cosine similarity between titles and texts.

5.2 Deep Semantic Similarity Model

Inspired by the Deep Semantic Similarity Model (DSSM)[21] developed by Huang et al. [36] in 2013, a similar DSSM network was constructed in this work.

Using the obtained cosine similarity scores and word2vec for text representation, a deep neural network that learns the similarity function was trained. Cosine similarity scores were appended as a new column in our dataset and used as a target variable. This is not a classification problem, but rather a regression model since we are predicting continuous values. Therefore, standard metrics for evaluating of the model, such as accuracy, cannot be used. Instead, mean squared error (MSE) [37] and root mean squared error (RMSE) [37] were used.

MSE measures how close a regression line is to a set of data points, i.e., it measures their difference. The differences are taken, squared and averaged across our data. MSE is calculated using the following formula:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{14}$$

where N is the number of samples we are testing against. RMSE, on the other hand, is the square root of MSE:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{15}$$

[21] Also known as Deep Structured Semantic Model [36].

Besides the input layer, the DSSM neural network employed in this work consists of three layers:

- Dense layer with 128 neurons and the ReLU activation function,
- Dense layer with 64 neurons with the ReLU activation function,
- Output dense layer with one neuron with the sigmoid activation function.

Figure 15 shows the structure of the DSSM neural network.

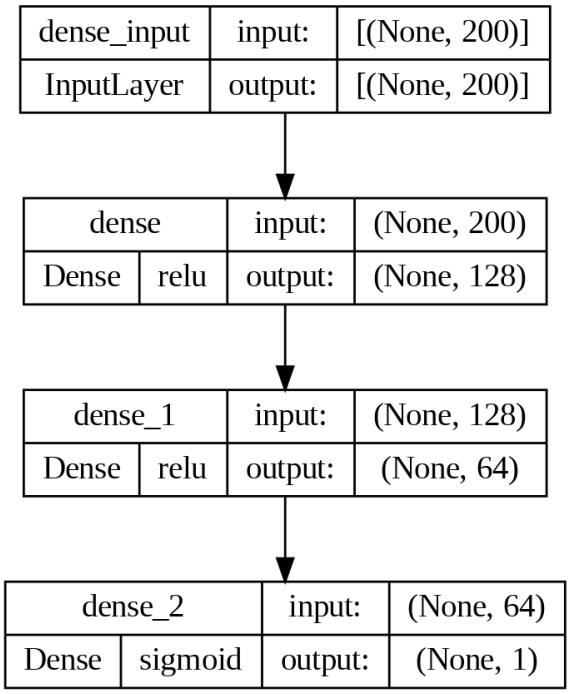


Figure 15: Structure of the DSSM network used in this work.

Alongside the sigmoid activation function described earlier, here the ReLU activation function [27] is used. ReLU or Rectified Linear Unit has a range from 0 to infinity, making it ideal for convolutional neural networks and deep learning in general. The function and its derivative are both monotonic. The following formula is used for the ReLU:

$$\Phi(x) = \max(0, x). \tag{16}$$

In the ReLU activation function, all negative values become zero. The ReLU graph can be seen in Figure 16.

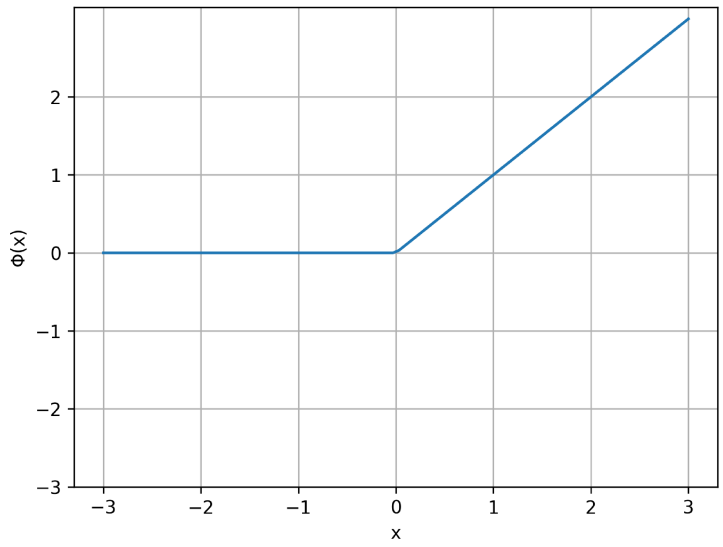


Figure 16: ReLU activation function.

Using the Adam optimizer, the network was trained over the course of 100 epochs with a batch size of 32 using MSE and RMSE as loss functions. Figure 17 shows the loss function over the training epochs. The blue line denotes train loss and the orange one denotes test loss.

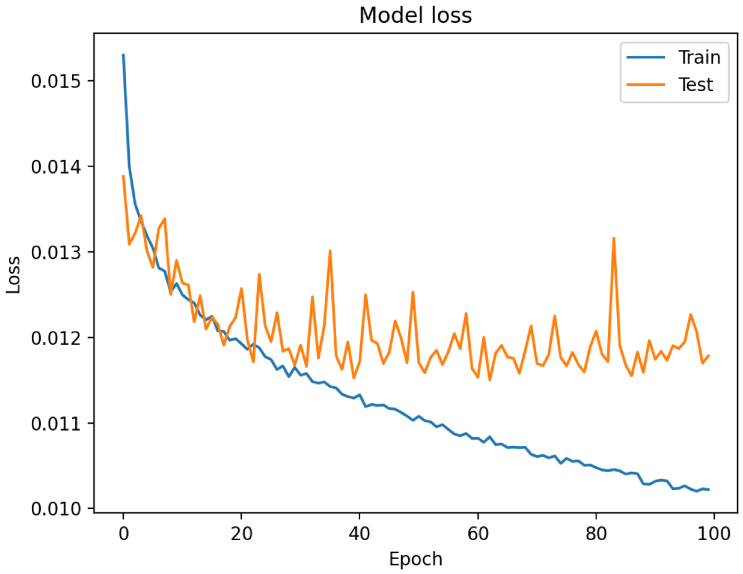


Figure 17: Training and test loss during training.

Our model is performing well on the test set, as both the MSE and RMSE values are relatively low. An MSE of 0.0118 and an RMSE of 0.2598 were obtained.

6 Conclusion

Clickbait titles are more and more prevalent in online media, and this should be addressed. In this master thesis, various machine learning and deep learning techniques were utilized for the clickbait classification on three datasets. Three datasets used in this work are the dataset by Chakraborty et al., the Webis-Clickbait-17 dataset and the merged dataset consisting of these two.

Three text representations were used, including TF-IDF, word2vec and FastText. FastText representation proved to be the best one since it provided us with the highest overall performance. Among the tested machine learning methods (NB, RF, SVM and linear perceptron), SVM is the preferred one because it achieved the highest average performance with F1 score of 0.8797 on the merged dataset. Still, SVM has a substantial downside – it takes, by far, the most time to execute.

Deep learning techniques included the use of LSTM and GRU architectures. Due to the similarities between these two architectures, their results (F1 scores of 0.8361 and 0.8321 on the merged dataset) are comparable. It is worth noticing that SVM with FastText representation outperforms both neural networks by approximately 4%.

Finally, title-text similarity analysis was conducted. The results, expectedly, suggest that non-clickbait titles are more similar to the texts. Therefore, they better represent the article texts (average similarity of 0.6024) than the clickbait titles (average similarity of 0.5411).

The work in this thesis provides insights into different methods and models for clickbait/non-clickbait classification. The results indicate that machine learning models are slightly preferred solutions when dealing with limited training corpora. The results confirm that classifiers could be used to actively detect clickbait titles and possibly enhance content quality and improve user experience by combating clickbait sensationalism, which can also be the first indicator of misinformation.

There are a few limitations in this work that can be addressed in the future. First, models are trained only on English titles. The datasets in this work are in English because there is a lack of quality datasets in other languages online. Collecting non-English datasets would contribute to the field of clickbait classification in general. Second, other text representations like n-gram models [38], character-level representation [39], syntax-based representation [40] and graph-based representation [41] should be considered. Finally, other semantic similarity scores (i.e., similarity proposed by R. Mihalcea in [42]) in the DSSM network can be tested. Likewise, additional machine learning and deep learning techniques (e.g., transformers [30]) should be evaluated since they may achieve better performance for this particular task.

7 References

[1] N. Zuhroh and N. Rakhmawati, "Clickbait detection: A literature review of the methods used," *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 6, p. 1, 10 2019.

[2] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, "Clickbait Detection," in *Advances in Information Retrieval* (N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, eds.), (Cham), pp. 810–817, Springer International Publishing, 2016.

[3] G. Loewenstein, "The Psychology of Curiosity: A Review and Reinterpretation," *Psychological Bulletin*, vol. 116, pp. 75–98, 07 1994.

[4] 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?," *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 232–239, 07 2017.

[5] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, "Stop Clickbait: Detecting and preventing clickbaits in online news media," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 9–16, IEEE, 2016.

[6] A. Agrawal, "Clickbait Detection using Deep Learning," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pp. 268–272, 2016.

[7] A. Anand, T. Chakraborty, and N. Park, "We Used Neural Networks to Detect Clickbaits: You Won't Believe What Happened Next!," in *Advances in Information Retrieval* (J. M. Jose, C. Hauff, I. S. Altıngovde, D. Song, D. Albakour, S. Watt, and J. Tait, eds.), (Cham), pp. 541–547, Springer International Publishing, 2017.

[8] V. Kumar, D. Khattar, S. Gairola, Y. K. Lal, and V. Varma, "Identifying Clickbait: A Multi-Strategy Approach Using Neural Networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ACM, jun 2018.

[9] R. Falotico and P. Quatto, "Fleiss' kappa statistic without paradoxes," *Quality & Quantity*, vol. 49, pp. 463–470, 03 2014.

[10] M. Potthast, T. Gollub, K. Komlossy, S. Schuster, M. Wiegmann, E. P. Garces Fernandez, M. Hagen, and B. Stein, "Crowdsourcing a Large Corpus of Clickbait on Twitter," in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 1498–1507, Association for Computational Linguistics, Aug. 2018.

[11] M. Potthast, T. Gollub, M. Hagen, and B. Stein, "The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength," *ArXiv*, vol. abs/1812.10847, 2018.

[12] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[13] B. Mahesh, "Machine Learning Algorithms – A Review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[14] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Proceedings of Workshop at ICLR*, vol. 2013, 01 2013.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, 07 2016.

[17] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. Springer, 2013.

[18] M. Minsky and S. Papert, "An introduction to computational geometry," *Cambridge tiass., HIT*, vol. 479, p. 480, 1969.

[19] A. Kulkarni, D. Chong, and F. A. Batarseh, "5 - Foundations of data imbalance and solutions for a data democracy," in *Data Democracy* (F. A. Batarseh and R. Yang, eds.), pp. 83–106, Academic Press, 2020.

[20] J. Mohajon, "Confusion matrix for your multi-class machine learning model," Jul 2021.

[21] E. Chen, S. Qiu, C. Xu, F. Tian, and T. Liu, "Word embedding: Continuous space representation for natural language," *Shuju Caiji Yu Chuli/Journal of Data Acquisition and Processing*, vol. 29, pp. 19–29, 01 2014.

[22] A. Rajaraman and J. D. Ullman, *Data Mining*, p. 1–17. Cambridge University Press, 2011.

[23] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," *Journal of Machine Learning Research*, vol. 3, pp. 932–938, 01 2000.

[24] N. Birajdar, "Word2vec research paper explained," Mar 2021.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 10 2013.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[27] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*. O'Reilly Media, Inc., 2017.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[31] I. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Computer Science*, vol. 2, 08 2021.

[32] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014.

[34] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP), (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[35] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier Inc., 01 2012.

[36] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning Deep Structured Semantic Models for Web Search using Clickthrough Data," in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, ACM International Conference on Information and Knowledge Management (CIKM), October 2013.

[37] B. Vishwas and A. Patel, *Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques*. Apress Berkeley, CA, 01 2020.

[38] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992.

[39] C. D. Santos and B. Zadrozny, "Learning Character-level Representations for Part-of-Speech Tagging," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 1818–1826, PMLR, 22–24 Jun 2014.

[40] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A Novel Neural Source Code Representation Based on Abstract Syntax Tree," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 783–794, IEEE, 2019.

[41] S. S. Sonawane and P. A. Kulkarni, "Graph based Representation and Analysis of Text Document: A Survey of Techniques," *International Journal of Computer Applications*, vol. 96, pp. 1–8, 06 2014.

[42] R. Mihalcea, "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization," in *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, (USA), p. 20–es, Association for Computational Linguistics, 2004.

List of Figures

List of Tables