3244-1910-0018-1 - Project Final Report

Authors: Jin Minjia, Joyce Lim Li Jie, Lim Kian Hwee, Sanjukta Saha, Sarah Taaher Bonna
Emails: e0201063@u.nus.edu, e0202859@u.nus.edu, e0014608@u.nus.edu, e0260142@u.nus.edu, e0200838@u.nus.edu

Abstract

This project explores several approaches, including Traditional Machine Learning models and Deep Learning approaches, to detect clickbait articles. Two key questions were raised: which features are most important in doing so, and which approach is the best for identifying clickbait. From the Random Forest model with manually engineered features as inputs, the proportion of the number of proper nouns in the content is the most important feature for identifying clickbait. Out of the models implemented, Recurrent Neural Network using LSTM has the highest accuracy and F1-score, and is the best model.

1. Introduction

The advent of the Internet and social media has made it easy for news to be shared worldwide instantaneously. Along with all of its advantages, this has also made it easy for misleading "news" to be propagated in a sensationalised and convincing manner - a phenomenon commonly known as 'Clickbait'. Clickbait can be defined as such: it is when an online article has an attention grabbing and incredulous sounding title that heavily exaggerates content that is either vaguely true or totally false. Some examples are: "Man tries to hug a wild lion, you won't believe what happens next!"[1] and "The brain enhancing smart pills that are sweeping Silicon Valley"[2]

These articles wastes people's time by disappointing them with content that do not live up to their incredulous sounding titles, and also directing people's attention away from things that are actually factually correct and relevant. With our project, we want to make it easy to identify these clickbait articles.

Thus, this motivates us to use Machine Learning techniques to create models that can automatically assess if an article is clickbait or not. To that end, two questions must be answered:

1. What are the features of an article that make it clickbait?
2. What Machine Learning approaches are good for detecting clickbait?

To answer the above questions, 7 models were implemented, consisting of Traditional Machine Learning approaches as well as Deep Learning approaches. Traditional Machine Learning involves engineering features from the data manually, whereas Deep Learning refers to using a Neural Network to automatically generate features from the data using its hidden layers. The main metrics used in assessing the models are classification accuracy and F1-score.

2. Related Work

In [1], four sections of a newspaper were studied manually by a few people to find clickbait features that were important to capture people's attention. They found that features like "informal language" in headlines were used quite frequently.

In [2], RNN was used very successfully to detect clickbait articles. They passed the clickbait article headlines into a few different RNN architectures, including LSTM.

In [3], the authors found a few important differences between clickbait and non-clickbait articles. For example, they found that clickbait articles contain a greater number of stop words. Using these differences, they created a few features themselves, and fed them into an SVM, a Decision Tree and a Random Forest model.

We have noticed that not much work has been done to compare the performance of traditional Machine Learning models to that of Deep Learning models, when detecting clickbait. Thus we want to incorporate the findings from the previously stated studies, and determine whether a traditional Machine Learning Approach is better or a Deep Learning Approach.

3. Method

The dataset used for training was obtained from the Clickbait Challenge[3], and it consists of news articles and social media posts. Specifically, the dataset "clickbait17-train-170630" was used. Instances in this dataset contained data such as article title, article content, the social media post title, labels of whether the article is clickbait or not, among others. After filtering out instances where the article title was an empty string (these usually represented tweets), there are in total 19538 instances, with a breakdown of 14777 non-clickbait instances and 4761 clickbait instances. Whether or not an article is considered clickbait was "measured against a crowd-sourced test set"[1].

As the dataset is unbalanced in terms of the number of clickbait and non-clickbait instances, naive random undersampling and oversampling approaches were considered in order to balance the dataset. For the random oversampling approach in particular, the dataset was first split into the training and test set, before oversampling the training set to balance the number of instances in each class. For all our methods, the splitting was standardized to 80% of observations to be used in training set and 20% of observations to be assigned to the test set. Alongside oversampling/undersampling, the models were also trained directly on the imbalanced dataset.

Given the nature of the problem topic as well as the dataset, particularly the presence of labels, this problem is a supervised learning problem, in the domain of Natural Language Processing (NLP). From this, we narrowed down the models and approaches we can use to tackle the problem, with approaches from both traditional Machine Learning techniques as well as Deep Learning. For traditional Machine Learning approaches, we decided to manually engineer features that we believed were relevant to the problem of detecting clickbait articles, as seen in Table 1, from the instances in the dataset and use them as inputs to 4 models: Logistic Regression, Naive Bayesian model, K-Nearest Neighbours and Random Forest. In terms of Deep Learning approaches, we implemented 3 models: Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN); in these models, we use as input the article titles after they have been pre-processed, as described below.

The features that were engineered for use in the traditional Machine Learning approaches are shown in Table 1. NLP Libraries such as NLTK[4] and TextBlob[5] were used to generate them.

Table 1: Features engineered from dataset

S/N	Name of Feature
1	Proportion of exclamations in the article title
2	Proportion of exclamations in the article content
3	Proportion of question marks in the title
4	Proportion of question marks in the content
5	Proportion of all-capitalised words in the title
6	Proportion of all-capitalised words in the content
7	Proportion of contractions in the title

[1] http://earthporm.com/man-tries-hug-wild-lion-wont-believe-happens-next/
[2] https://www.inc.com/vicky-castro/the-brain-enhancing-drug-thats-sweeping-silicon-valley.html
[3] https://www.clickbait-challenge.org/
[4] https://www.nltk.org
[5] https://textblob.readthedocs.io/en/dev/

8	Proportion of contractions the content
9	Proportion of words in the title that appear in the content
10	Sentiment of the title
11	Whether the title starts with a number
12	Whether the content starts with a number
13	Length of the longest word in the content
14	Whether the title starts with a 5W1H[6] word
15	Whether the content starts with a 5W1H word
16	Proportion of proper nouns in the content
17	Proportion of stop words in the title

The article titles of the instances were used as input for the Deep Learning models. These were first pre-processed to remove punctuation and numbers, single-character letters and multiple spaces. Then, these were tokenized and padded so that each processed title had the same number of tokens. These padded titles were used as input, and passed into a pre-trained embedding layer. FastText[7] (for CNN) and GloVe[8] (for MLP and RNN) were used for the pre-trained embedding layers.

For each model, validation was done in order to determine the best hyperparameters to be used. The specific hyperparameters that were tuned are discussed for each model separately. The final hyperparameters were chosen such that the lowest validation loss was achieved.

4. Evaluation

The metrics used to evaluate a model were the model's accuracy and F-1 score. The use of F1-score is especially important due to the imbalanced original dataset, and the need to discriminate between clickbait and non-clickbait articles.

The baseline used for comparison is the baseline result published on the Clickbait Challenge website - an F1 score of 55%, and an accuracy of 83%.

To answer the first question, the Random Forest approach was used to determine the relative importance of all the features mentioned in Table 1.

To determine which model performed the best in order to answer the second question, all models used article titles as input information, in order to achieve a meaningful comparison. Thus, a subset of features, regarding the article title, was used as input for the traditional ML models. These features are listed in Table 2. The Deep Learning methods, such as MLP, CNN and RNN used pre-processed article titles to analyse whether an article is clickbait or not. The reasoning behind using the article titles is that such a process models after the decision-making process of people in deciding whether to read an article based on its title. Furthermore, if a suitable decision can be made based on the article title, people would not have to waste their time to read the rest of the clickbait article and be disappointed.

Table 2: Features used in traditional ML models for comparisons between models

S/N	Name of Feature
1	Proportion of exclamations in the article title
3	Proportion of question marks in the title
5	Proportion of all-capitalised words in the title
7	Proportion of contractions in the title
10	Sentiment of the title
11	Whether the title starts with a number
14	Whether the title starts with a 5W1H word
17	Proportion of stop words in the title

Traditional Machine Learning Approaches

[6] Who, What When, Where, Why, How
[7] https://fasttext.cc/
[8] https://nlp.stanford.edu/projects/glove/

Logistic Regression

Scikit-learn's LogisticRegression was used to build this model. L2 normalisation was used on the training values of the features, and liblinear optimizer was used in the training of the model. For this model, the training dataset was treated in 3 ways: nothing done to it, oversampling and undersampling. The hyperparameters to be chosen for each approach are C, which is the inverse of regularization strength, and max_iter, which is the number of iterations to train the model; these values were chosen through 10-fold cross-validation, where the combination of values that gave the best mean accuracy was chosen.

For the first approach, without oversampling or undersampling, the best C value obtained is 0.1. After training the model with this C value, a classification accuracy of 75.5% and an F1-score of 8.80% was achieved. The normalized confusion matrix shows that about 95% of clickbait articles are misclassified as non-clickbait, which showed that having a high classification accuracy does not necessarily mean that the model is good enough in discerning clickbait articles.

Using oversampling, the best C value obtained is 10. After training the model with this C value, a classification accuracy of 23.9% and a F1-score of 38.2% was achieved.

Using undersampling, the best C value obtained is 100. After training the model with this C value, a classification accuracy of 48.7% and a F1-score of 65.2% was achieved.

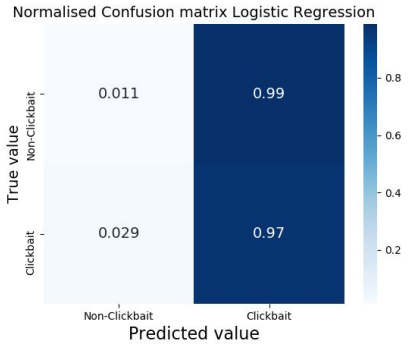

Figure 1: Normalised Confusion Matrix for Logistic Regression, with undersampling

Based on the F1-score, using undersampling is the best choice for Logistic Regression. The confusion matrix for this model is shown in Figure 1.

One thing to note about the Logistic Regression model with undersampling is that although it has a decent F1-score of 65.2%, it has a training accuracy of about 48.7%, which is the lowest among all the models. As such, it is not the best model for this problem.

Random Forest and Bagging

This model is used to answer both the questions raised. First, all 17 predictors in Table 1 were considered to answer the first question.

For random forest with oversampling, we first decided on the number of predictors to use at each split. As we have 17 predictors, we considered using 1 to 16 predictors at each split, and fitted a model for each number of predictors considered, resulting in 16 models. For the trees grown, the stopping criterion for the split was when the decrease in entropy was no longer significant. This resulted in trees of average depth 3, thus preventing overfitting. The model with the best prediction accuracy was picked as the final model, using just 1 predictor at each split (n=1). Ensembling 1100 trees gave the highest prediction accuracy of 69.0% with an F1-score of 63.5%. The trees grown in the final model were also diverse, as at each split any one of the 17 predictors could be chosen as the predictor to be considered for splitting.

Random forest was repeated again on the original dataset without oversampling, and it was found that considering n=4 features at

each split and ensembling 1400 trees gave the highest prediction accuracy of 78.4%. The resulting F1-score is 39.3%. Even though the prediction accuracy increased, the F1-score also decreased significantly.

Finally, random forest was performed on the undersampled dataset, and it was found that considering n=4 features at each split and ensembling 700 trees gave the highest prediction accuracy of 69.8%, and an F1-score of 68.9%.

Shown in Figure 2 is the normalized confusion matrix from undersampling for random forest.

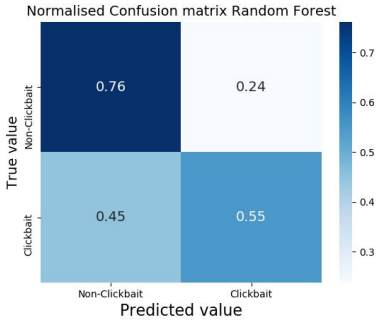

Figure 2: Normalized confusion matrix for random forest, undersampling

Thus, the random forest model from undersampling strikes a balance between F1-score and prediction accuracy and seems to be the best model out of the three.

For bagging, a variant of the random forest was performed with oversampling, and considering all 17 predictors (n=17) at each split. Again, the stopping criterion for the trees is when the decrease in entropy is no longer significant, resulting in trees of average depth of 3. We considered a range of values of trees to use, ranging from 100 to 1500, and fitted a model using those different number of trees. This resulted in 15 models. It was found that using 800 trees with all 17 predictors gave the highest prediction accuracy of 63.9% on the test, out of all 15 models. The F1 score obtained from this model was 50.9%.

We then repeated bagging again on the original dataset, and found that ensembling 600 trees from (100 to 1500) trees resulted in the model with the highest prediction accuracy of 78.3%. The F1-score obtained is 40.5%.

Finally, we repeated bagging on the undersampled dataset, and found that ensembling 300 trees gave the highest prediction accuracy of 68.9%, and an F1-score of 61.6%.

The bagged model from undersampling has the highest F1-score and the 2nd highest prediction accuracy; it seems to be the best model out of the 3 as it balances between F1-score and prediction accuracy.

Shown in Figures 4 and 5 are 2 examples of the trees grown from random forest and bagging.

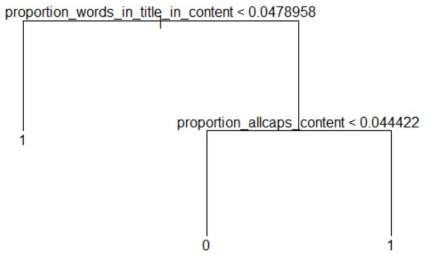

Figure 4: Example of tree grown from random forest, using undersampling

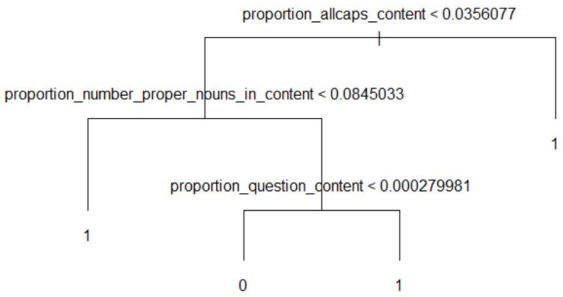

Figure 5: Examples of trees grown from bagging, using undersampling

From running the random forest, we were able to obtain the importance of each variable as shown in Figure 6. The variables are ranked in decreasing order of importance based on the Gini index, with the features that provide the highest average decrease in Gini index when used to split deemed to be the most important. From the results below, the 2 most important variables are the proportion of proper nouns in the article content and the proportion of words in the title that also appear in the content. Their relation with the types of clickbait are discussed in the Discussion section.

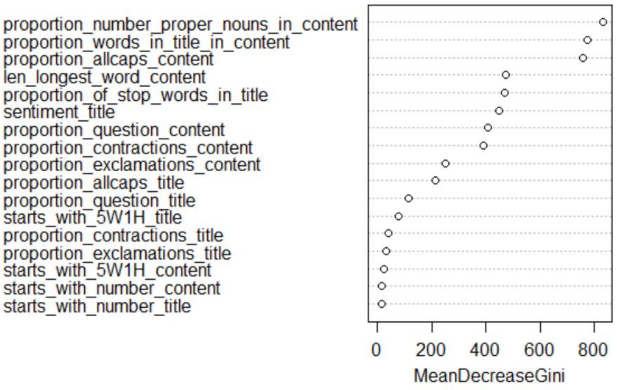

Figure 6: Importance of each feature based on the Gini Index

To allow for comparison between the deep learning approaches and random forest, random forest was performed on the 8 features in Table 2 that made use of the title only, and found that considering 2 features at each split and ensembling 200 trees led to the highest prediction accuracy of 60.5% and an F1-score of 58.4%.

Then, bagging was performed on the 8 features with undersampling, and obtain a model using all 8 features and ensembling 1100 trees. This resulted in a prediction accuracy of 58.5%, and an F1-score of 56.5%.

k Nearest Neighbours

For k-Nearest Neighbours (kNN), standardization of numerical features was needed as kNN uses a distance metric, so as to prevent the scales of the features to bias the kNN model. The dataset was first split into training and test sets before standardizing to prevent data leakage. The distance metric used is Euclidean distance.

First, kNN was performed on the oversampled dataset. As the number of neighbours k is a hyperparameter, k values from $k \in [1, 500]$ were chosen, starting with k=1. Then for each value of k, a kNN model was fitted on the training set, and then tested on the test set to obtain a prediction accuracy resulting in 500 kNN models with 500 prediction accuracies. The model with the highest prediction accuracy (k=40) was picked, which had a prediction accuracy of 66.8% and an F1-score of 65.2%. k=500 was chosen as the last value for k as the prediction accuracy was generally decreasing after k=40.

Then, kNN was repeated on the original dataset without any oversampling or undersampling for $k \in [1, 500]$, and it was found that k=11 gave the highest prediction accuracy of 76.9%, with an F1-score of 33.5%. The prediction accuracy has increased while the F1-score decreased when using the original dataset.

Finally, kNN was repeated on the undersampled dataset, and it was found that using k = 45 from k ∈ [1, 500] gave the highest prediction accuracy of 65.5%, and an F1-score of 61.6%.

The oversampled kNN model with k=40 seems to be the best model out of the 3, with it having the highest F1 score and 2nd highest prediction accuracy. Shown in Figure 7 is the normalized confusion matrix for kNN with oversampling.

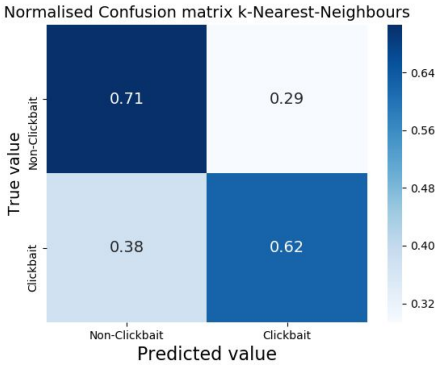

Figure 7: Normalized confusion matrix from kNN, using oversampling

Naive Bayes

For Naive Bayes, firstly a check was performed to see if the features were correlated.

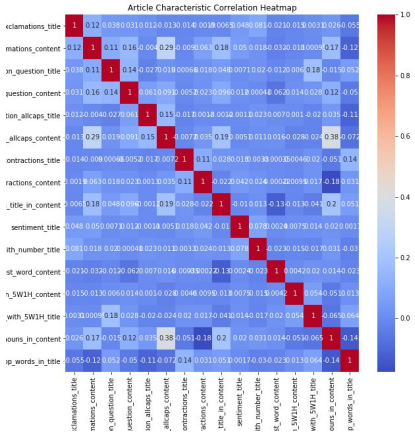

Figure 8: Feature correlation heatmap

As can be seen from Figure 8 above, the features were not correlated, which means the "naive" assumption in Naive Bayes holds well. For our problem, Gaussian Naive Bayes was used as our model. Gaussian Naive Bayes was chosen over Multinomial Naive Bayes because individually, some of the features exhibited a bell-curve shape, and one of the features (sentiment of the title) had negative values.

As usual, the Gaussian Naive Bayes model was trained on oversampled, undersampled and imbalanced training data. The best results were obtained for oversampled data, with an accuracy of 74.0%, and an F1-score of 0.380 when tested on the test set. For undersampled data, the accuracy was 57.0%, with an F1-score of 0.420. On imbalanced data, the accuracy was 74.0%, but with an F1-score of 35.0%.

One noteworthy thing is that the Gaussian Naive Bayes model had a training accuracy of about 72.0%. This means that the bias for the Gaussian Naive Bayes model is quite low, which means that the variance is high.

The normalised confusion matrix for the oversampled model is shown in Figure 9.

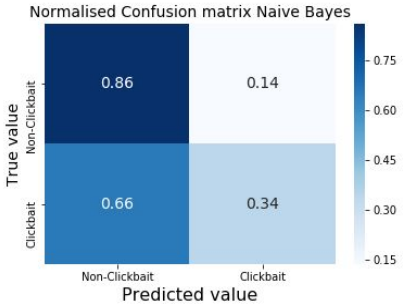

Figure 9: Normalized confusion matrix for Naive Baye, on oversampled training data

Deep Learning Approaches

Across the models, the article titles were pre-processed and padded, before being fed into the embedding layer. Binary Cross-entropy loss and Adam optimizer were used and the final layer used a Sigmoid function as activation in order to output the probability of an article being clickbait.

Multi-Layer Perceptron

The network architecture is as shown in Figure 10. For this model we used a single dense hidden layer of 64 neurons in the undersampled dataset version, and 128 neurons in the oversampled dataset version, and these values were chosen through validation. The intermediate dense layer used ReLu as the activation function, and the final layer used the Sigmoid function to output a probability of an article being clickbait. Regularisation in terms of dropouts and batch normalisation was applied. The model was trained for 100 epochs, and the actual model with the lowest validation loss was then used for testing on the test set.

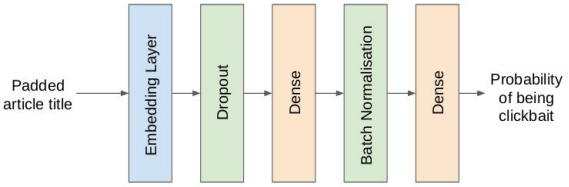

Figure 10: Architecture of MLP model used

The purpose of this relatively simple model is to provide a baseline in performance for the other Deep Learning approaches. Undersampled and oversampled data sets were used to train the MLP. It is worth noting that for this model, using undersampling yielded a better model due to its higher F1-score. Using undersampling, the accuracy (65.6%) was lower than with oversampling (71.4%). However, the F1-score using undersampling (65.6%) was much higher than when using oversampling (49.0%). The normalized confusion matrix achieved for undersampling is shown in Figure 11. The lowest validation loss in the model using the undersampled dataset is 0.623.

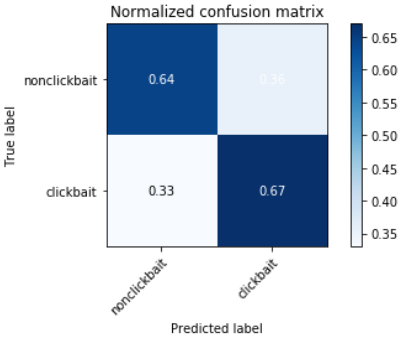

Figure 11: Confusion Matrix for MLP using an undersampled dataset

Convolutional Neural Networks

Convolutional Neural Networks have been used both in image data and text data. For sequential data like text, a 1-D CNN would be able to detect features in localised parts of a sequence. For example,

perhaps the 1-D CNN would be able to detect chunks of text like "You won't believe" in its hidden layers. The CNN architecture is shown below in Figure 12. Hyperparameters like the number of filters in each layer, the amount of dropout, size of the convolution kernel etc are chosen by validation. ReLu was used as the activation function of the convolution layers, while the Sigmoid function was used for the activation function for the final dense layer. Early stopping was used; the training of the model stopped when the validation loss increased for 4 consecutive epochs. No regularisation was imposed on the convolution layers.

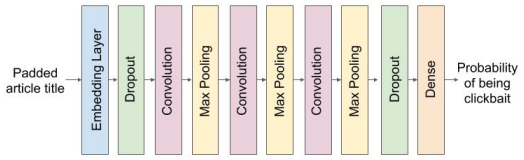

Figure 12: Architecture of 1D CNN model

For the 1-D CNN, oversampling gave the best results, taking into account both accuracy and F1-score. Oversampling gave an accuracy of 63% and an F1-score of 31%, as compared to undersampling which gave an accuracy of 55% and an F1-score of 55% and the imbalanced dataset which gave an accuracy of 76% with an F1-score of 1%. The lowest validation loss achieved using oversampling is 0.670. The normalised confusion matrix for the oversampled model is shown below in Figure 13.

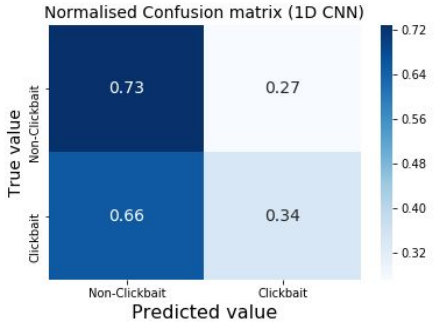

Figure 13: Normalised Confusion Matrix for CNN using an oversampled dataset

Recurrent Neural Networks

Recurrent Neural Networks are often used for sequential data such as text, because the sequence of words could convey important information in addition to the words themselves. This makes RNN an appropriate choice for this NLP problem. The RNN architecture used is shown in Figure 14. Using the undersampled dataset, a Long Short-Term Memory (LSTM) layer consisting of 64 cells and a Dense layer with 128 neurons were used; using the oversampled dataset, the LSTM layer had 128 cells and the Dense layer used 256 neurons. These hyperparameter values were chosen after validation. LSTM was chosen because it allows for the preservation of gradients. The intermediate dense layer uses linear activation, while the final dense layer used the Sigmoid function as the activation function. The model was trained for 100 epochs.

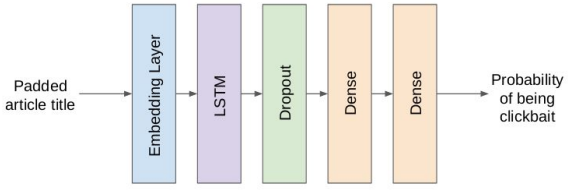

Figure 14: Architecture of RNN model

Undersampled and oversampled data sets were used to train the RNN. For RNN, undersampling yielded a better model due to its higher F1-score. Undersampling yielded a lower accuracy (68.9%) as compared to oversampling (72.7%). However, it led to a significantly higher F1-score (67.1%) as compared to oversampling (45.5%). The normalized confusion matrix for the undersampled model is as shown in Figure 15. The lowest validation loss in the model using the undersampled dataset is 0.604.

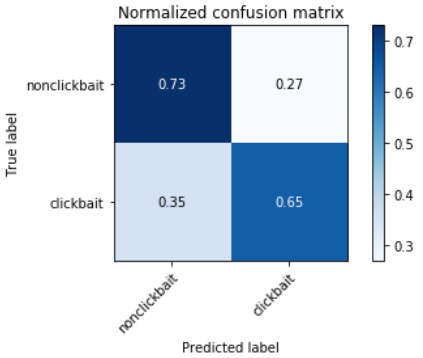

Figure 15: Confusion Matrix for RNN using an undersampled dataset

One thing of note is that the intermediate dense layer was crucial in getting the model to converge. From this, it can be inferred that combinations of sequences of words were important in determining whether an article is clickbait based on its article, and not just the sequences themselves.

Overall Results

The respective accuracy scores and F1-scores of the best-performing model in each approach is shown in Table 3. This table shows the results of the different methods when just the features from the title were used, so as to have a fair comparison between the different methods.

Table 3: Results of best-performing models using article title information as input

Model	Accuracy	F1-score
Logistic Regression	48.7%	65.2%
Random Forest	60.5%	58.4%
Naïve Bayes	74.7%	32.0%
kNN	59.6%	56.4%
Bagging	58.4%	56.5%
MLP	65.6%	65.6%
CNN	63.0%	31.0%
RNN	68.9%	67.1%

As seen from Table 3, the RNN model performed the best in terms of F1-score, and Naive Bayes performed the best in terms of prediction accuracy. This is likely due to how RNNs account for the sequences of words and the relationships between them, and in this specific model, how the combinations of sequences are considered in order to classify an article as clickbait or not based on the article title.

Between Traditional Machine Learning models and Deep Learning models, the performance was similar in terms of accuracy. However, Deep Learning models, especially MLP and RNN were better able to discriminate between clickbait and non-clickbait articles. Perhaps the performance of Deep Learning models could be improved with more data being used, especially when undersampling is used.

6. Discussion

To answer our question about which features are the most important for determining whether an article is click-bait, we have used the output from the Random Forest model to understand the importance of each of the 17 engineered features in Figure 6.

Figure 6 shows the features that are most important for determining whether an article is clickbait, in decreasing order of importance. The proportion of proper nouns in the article content and the proportion of words in the title that also appear in the content are deemed most important.

These results could be related to the types of clickbait articles. Clickbait articles usually leave out key details like names of people

or places (proper nouns), only giving brief descriptions without actually going into the specifics. Hence a higher proportion of proper nouns in the content indicates that the article is not clickbait.

Clickbait articles usually also have a sensational title that is attention-grabbing so as to attract clicks, and these titles may not match with the content. We have identified that the second most important feature for determining whether an article is click-bait is the proportion of words that appear in both the title and the content. Using random forest, we were able to pick out the more important variables, and the variables picked correspond to our understanding of clickbait intuitively.

To answer our question about what Machine Learning approaches are good for detecting clickbait, specific instances were used as sample points to evaluate each model.

2 articles, one clickbait and one non-clickbait, were picked from another dataset hosted by the Clickbait Challenge to test the best-performing version of each model from above. These instances' article titles are:

1. [non-clickbait] "Apple gives back gigabytes: iOS 9 'app thinning' feature will finally give your phone's storage a boost"
2. [clickbait] "U.S. Soccer should start answering tough questions about Hope Solo"

For traditional Machine Learning models, Logistic Regression with undersampling, which gave the best result, classified both articles as clickbait. Random Forest with undersampling classified both articles as non-clickbait. Bagging with undersampling classified both articles as non-clickbait. Naive Bayes with oversampling classified both articles as non-clickbait. kNN with oversampling classified both articles as clickbait.

On the other hand, the Deep Learning approaches MLP, CNN and RNN classified both instances correctly.

From the above, it is observed that the Deep Learning approaches are more appropriate for answering our problem statement. This corroborates with the general results obtained in Table 3, where Deep Learning approaches, particularly RNN, gave the best performance.

From working with the dataset, we found that the labels of the instances in the dataset could be subjective, due to their crowd-sourced nature. For instance, titles such as "Are You Paying Too Much for Your Tax Prep?" have been classified as non-clickbait in the dataset, even though they could subjectively be classified as clickbait. This highlights the subjectivity of the dataset labels and the nature of the topic. This subjectivity in determining whether an article is clickbait may not be replicable by Machine Learning models.

7. Conclusion

Random Forest was used to determine the most important features in deciding whether an article is clickbait or not. Out of the features in Table 1, it is seen that the proportion of proper nouns in the content was most important, followed by the proportion of words in the title that also appear in the content.

The performance of several models were compared in terms of their accuracy and F1-score, in order to determine which model and approach were the best in detecting an article to be clickbait, based on information from the article title. The Deep Learning approach using RNN was the best model to do so.

Areas for Improvement

Undersampling the dataset resulted in fewer data instances exposed to the model in training. Oversampling the dataset resulted in duplicates of the clickbait instances, and could have contributed to the overfitting of the models to the duplicate instances. Data augmentation could have been used to supplement the clickbait instances such that the dataset is balanced instead of oversampling and undersampling, which would eliminate the problems associated with oversampling and undersampling. This could have been done by searching for additional datasets that contained classified clickbait and non-clickbait instances.

To tackle the subjectivity of the dataset as well as the subjective nature of the topic, we could improve our models by using a larger dataset to train on. We could also consider outputting a clickbait index (a probability value) rather than a binary value, so as to determine which article is considered more "clickbait-y", thus accounting for the subjectivity of the subject itself.

Future Work

Deep Learning on article content

A combination of article title and content could be used to train our models. Using the article content in addition to the article title could enhance the accuracy and F1-scores of Deep Learning models, due to a greater wealth of material to train on.

Social Media and Clickbait

The relationship between the social media activity associated with an article, and the likelihood of it being clickbait, could be explored. From prior studies [4], social media activity looked to be good predictors for an article being clickbait or otherwise. Such social media activity could be in the form of the content of tweets that shared the article, reactions to the article on the Facebook page where it was shared, etc.

References

[1] Palau-Sampio, Dolors. (2016). Reference press metamorphosis in the digital context: Clickbait and tabloid strategies in Elpais.com. Communication & Society. 29. 63-79. 10.15581/003.29.2.63-79.

[2] Ankesh Anand, Tanmoy Chakraborty, and Noseong Park. We used neural networks to detect clickbaits: You won't believe what happened next! In European Conference on Information Retrieval, pages 541–547. Springer, 2017.

[3] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on, pages 9–16. IEEE, 2016.

[4] Amin Omidvar, Hui Jiang and An Aijun (2017). Using Neural Network for Identifying Clickbaits in Online News Media. Retrieved from https://www.clickbait-challenge.org/papers/omidvar18-notebook.pdf