Detecting Clickbait Hyperlinks Using Deep Learning

Brian Cui
The University of Texas at Austin
briancui@utexas.edu

Akshay Kumar Gupta
The University of Texas at Austin
akshaykg@cs.utexas.edu

Abstract

Clickbait titles are sensationalized, attention-grabbing headlines which entice users into clicking on and viewing content with little material value. Although clickbait has become increasingly popular on the Internet, negative user response to clickbait has motivated companies and researchers to develop automatic means of filtering clickbait on news feeds. In this paper, we present a deep learning model for detecting clickbait hyperlinks on webpages, enabling automatic clickbait detection on all Internet platforms. Our most powerful model achieved 98% classification accuracy, precision, and recall. Insight into our dataset reveals the first four words in a hyperlink are most critical for determining whether or not the hyperlink represents clickbait.

1 Introduction

Clickbait is a deliberately sensationalized and attention-grabbing headline that entices users to click on and view content with little material value. The low cost of publishing media online, and the potential for ad revenue has motivated the presence of clickbait on several Internet channels, including news websites, social media feeds, web advertisements, and email spam.

Although clickbait is popular enough to serve as a successful source of revenue for websites like Buzzfeed, survey data shows users have an overwhelmingly negative response to clickbait titles (Hu, 2017). The increasing prevalence of clickbait has motivated Facebook, Reddit, and other popular Internet social media platforms to take a stand against clickbait by demoting low-value content using simple pattern-matching strategies and crowdsourced reporting.

There is some existing research on performing automated clickbait detection without human intervention. (Chakraborty et al., 2016) uses feature engineering to evaluate the effectiveness of several heuristics for detecting clickbait. (Anand et al., 2016) uses a deep learning strategy combining both word and character vector embeddings to distinguish clickbait from non-clickbait news article titles.

While both feature engineering and deep learning methods were found to be effective, the scope of these methods were limited to purely titles of news articles. As such, the datasets used to train these models were limited in size and could not fully encompass the range of language expressions present on the Internet. Additionally, the applications of these models are restricted to filtering content on sites that present news article titles, but not other platforms like social media or email, which feature significantly greater diversity in content.

In this paper, we explore a deep learning strategy for automatically detecting and flagging clickbait hyperlinks. Rather than targeting exclusively article titles, we expand the scope of our model to classify all hyperlinks present on webpages. These can range from benign navigation links (e.g. ”Home”, ”About”, ”Next Page”), to listings of products (e.g. ”Low Profile Grey Cotton Baseball Cap”) to links to articles (e.g. ”18 Things That Will Unquestionably Make You Shake Your head And Sigh, ’LOL’”).

Our work extends the LSTM model with word embeddings by (Anand et al., 2016) to classify and distinguish clickbait article titles from other benign hyperlinks on the Internet. We combined a random selection of several thousand hyperlinks we scraped, with the datasets released by (Chakraborty et al., 2016), to produce a training set of nearly 50,000 clickbait and non-clickbait hyperlinks, almost triple the size of other datasets used in existing research. Our most powerful model achieved a 98% accuracy in classifying hyperlinks as clickbait vs. non-clickbait.

2 Problem Definition and Algorithm

2.1 Task Definition

The problem we seek to solve is the automated detection of clickbait hyperlinks on webpages. Our goal is to train an effective machine learning model that accepts the text encapsulated by a hyperlink, and then classifies the text as clickbait or non-clickbait. Such a model would enable detection (and prevention) of clickbait on the client side (in the web browser), enabling users to filter clickbait as they browse the web.

Formally defined, *clickbait* is a sensationalized, attention-grabbing headline that exploits human curiosity in order to entice readers ("clicks") into viewing opinionated articles with little insightful or material value. Examples of clickbait (hyperlink text) from our dataset include:

31 Valentines Gift Horror Stories That'll Make You Glad You're Single

You Will Never Be This Creeped Out By Anything In Your House Hopefully

21 Pictures That Are Way, Way Too Real For Every College Student

A common theme among clickbait titles are list-based articles ("listicles") that force readers to browse through several pages view the entire article's contents, garnering additional ad revenue for each click. In our dataset, 34% of all clickbait titles we collected began with a number. The ubiquity of this title format makes it possible for even our weakest models to classify clickbait with an appreciable level of accuracy, which we discuss in §3.2 and §3.3.

Non-clickbait is simply hyperlink text that is not clickbait. The vast majority of hyperlinks on the Internet are non-clickbait, corresponding to navigational links (e.g. "Home", "News", "Reviews"). However, non-clickbait also includes article titles that are similar in length and format to clickbait. Examples from our dataset include:

Study suggests 'sleeping sickness' parasite mutated to evade immune system defences

Indonesian suspected bird flu case dies; symptoms included encephalitis

These titles are generally objective and succinct in nature, and avoid directly targeting the reader like clickbait does. While some news publications stray away from clickbait (e.g. The New York Times) and others nearly exclusively feature clickbait (e.g. Buzzfeed, ClickHole), there are websites with a mix of clickbait and non-clickbait article titles (e.g. Huffington Post).

In the following section §2.2, we discuss how we collect hyperlink text for our dataset and distinguish clickbait and non-clickbait hyperlinks.

2.2 Dataset

To collect hyperlinks, we wrote a custom recursive web crawler to gather the text present in HTML anchor tags (`<a>`) from a sample of the top 50 most popular websites with English as a primary language. Anchor tags that surrounded non-text content (like an image) or triggered dynamic content via JavaScript were discarded.

Some websites, which feature carefully curated content, were deemed non-clickbait (e.g. WikiNews.org) and all their anchor tags were classified as such. Most sites featured a mix of content that we could not issue a blanket classification for; for these sites, we declared text tags less than or equal to three words in length (the vast majority of anchor tags) to be non-clickbait.

Specific adapters were written for sites known to contain clickbait content such as Buzzfeed or ClickHole. These adapters target specific anchor tags on the page where clickbait article titles expected, so they can be placed in the clickbait set.

In total, approximately 250,000 unique non-clickbait anchor tag texts were collected, and 9000 unique clickbait anchor tag texts were collected. We combined our clickbait set with 16,000 clickbait titles released by (Chakraborty et al., 2016) for a total of 25,000 clickbait texts. An equally sized random sample of the 250,000 non-clickbait texts was taken to create a training set of around 50,000 anchor tag texts, with exactly 50% of them clickbait and 50% of them non-clickbait.

2.3 Algorithm Definition

Our machine learning model was greatly inspired by the work of (Anand et al., 2016), which experimented with various Recurrent Neural Net (RNN) architectures and input strategies for clickbait classification. We chose the Long Short Term Memory (LSTM) RNN architecture, with each word input represented as a 300-dimension vector

word embedding. A fully connected output layer consisting of a single node classifies its input as either clickbait or non-clickbait.

The diversity of hyperlinks on the Internet meant that several words were unique (such as usernames, or proper nouns in news article titles), and only ever present once in the entire dataset. This led us to believe that training word embeddings from scratch would have been ineffective. Instead, we used pretrained word2vec embeddings published by Google consisting of approximately 3 million words trained on part of the Google News dataset, consisting of about 100 billion words.

Word tokenization was performed by splitting sentences by space and newline characters. Tokens (words) which did not have vector representations available in the word2vec database were mapped to a special unknown word vector. We expected many tokens containing special characters (e.g. a full URL contained in an anchor tag, such as `https://www.utexas.edu`) to be treated as unknown words.

(Anand et al., 2016) used character-level word embeddings to extract features and meaning out of unknown words. While we acknowledge the possible benefits of this strategy on a dataset with only English words, the presence of several possible nonalphanumeric characters in our dataset made it too difficult to train character embeddings from scratch. For example, the Twitter "hashtag" represented as `#example`, is distinct from using the hash # symbol as part of a numerical listing (e.g. `#1, #2, #3`).

In practice, unknown words are rarely present in clickbait, and upon seeing unknown words in text, our model almost always classified the text as non-clickbait. Thus, representing all unknown words as the same vector did not create issues with overgeneralization.

3 Experimental Evaluation

3.1 Methodology

We designed and trained our model using TensorFlow on Python 2 running on Ubuntu 16.04 LTS. Our dataset was randomly split into 75% training data, 12.5% validation data, and 12.5% test data. We used a batch size of 128 across 4 epochs (iterations across the entire training set). Cross entropy was our loss function, and gradient descent was performed using the Adam algorithm.

Our model was configured to truncate sentences exceeding a specified maximum length L, in tokens (hyperparameter). We trained and tested our model for $L \in [1, 20]$ to evaluate how well we could predict clickbait vs. non-clickbait using only the first L words (tokens) in a hyperlink text. The same training, validation, and test data were used for every model trained.

3.2 Results

Table 1 displays the training time (in seconds), accuracy, loss, precision, and recall of trained models for $L \in [1, 20]$ on the test set. The accuracy values are displayed in a chart in figure 1. Training and validation accuracy during training for model $L = 10$ are graphed in figures 2 and 3, respectively.

Our most powerful model, $L = 20$, achieved 98% accuracy, precision, and recall in classifying clickbait vs. non-clickbait hyperlink texts on our test set. This is comparable to the 98% accuracy scores achieved by (Anand et al., 2016) and exceeds the accuracy of the baseline (Chakraborty et al., 2016) by over 5%.

As expected, performance increased as L increased, as did training time. By $L = 10$, the model has achieved 98% classification accuracy. Accuracy, precision, and recall scores plateau past $L = 10$, and we see diminishing returns throughout with increased L.

Surprisingly, our model achieved respectable accuracy for even small values of L. The model trained on only the first word ($L = 1$) achieved 70% classification accuracy, though its precision (84%) and recall (52%) indicate a high false positive/negative rate and leave much to be desired. The model trained on the first four words ($L = 4$) reached 96% accuracy and had much better precision (95%) and recall (97%) scores.

3.3 Discussion

We weren't expecting to see such strong results with the weaker models. Even with $L = 2$, the trained model achieved more than 90% accuracy, precision, and recall on the test set. Accuracy scores converge by $L = 10$, and larger values of L do not lead to any significant increases in accuracy. This suggests only the first 10 words of a hyperlink text are necessary and relevant for determining whether the hyperlink embodies clickbait.

A more thorough investigation of our dataset revealed some insight on why this could be the case.

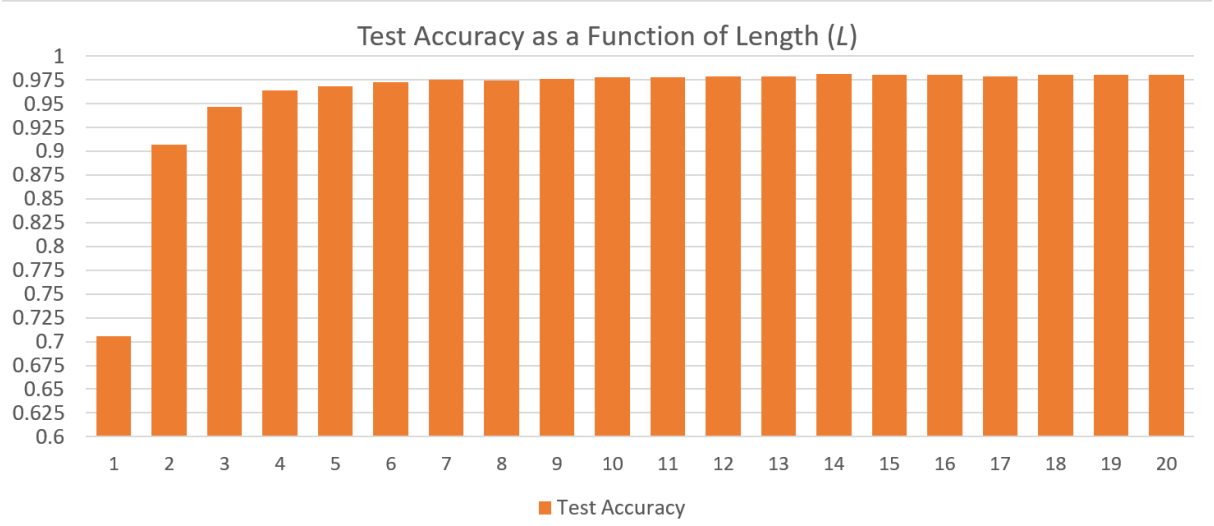

Figure 1: Classification accuracy on the test set for models trained with maximum token length L. The horizontal axis is L and the vertical axis is accuracy. By $L = 4$, the classifier achieves 95% accuracy. By $L = 10$, the classifier achieves achieved 98% accuracy.

L	Train Time (sec)	Accuracy	Loss	Precision	Recall
1	924	0.706	0.469	0.841	0.516
2	1013	0.907	0.228	0.914	0.902
3	1096	0.947	0.137	0.938	0.96
4	1162	0.964	0.092	0.958	0.973
5	1256	0.968	0.084	0.962	0.974
6	1336	0.973	0.071	0.972	0.975
7	1429	0.975	0.068	0.973	0.978
8	1514	0.974	0.065	0.974	0.975
9	1619	0.976	0.061	0.978	0.975
10	1701	0.978	0.057	0.981	0.975
11	1807	0.978	0.058	0.98	0.975
12	1877	0.979	0.054	0.977	0.983
13	1956	0.979	0.054	0.979	0.98
14	2052	0.981	0.05	0.979	0.984
15	2144	0.98	0.054	0.983	0.977
16	2238	0.98	0.052	0.98	0.98
17	2325	0.979	0.056	0.982	0.977
18	2445	0.98	0.052	0.981	0.978
19	2541	0.98	0.054	0.983	0.977
20	2633	0.98	0.054	0.983	0.977

Table 1: Test Results

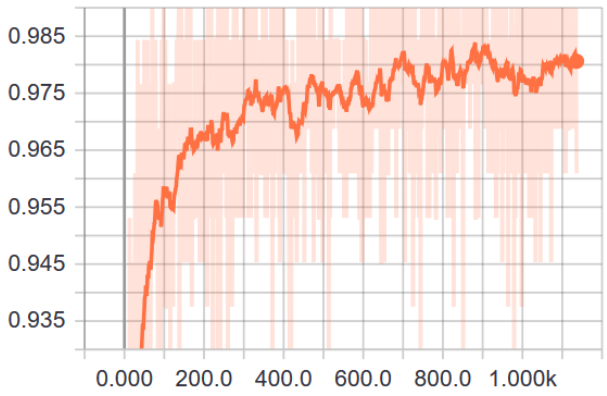

Figure 2: Training accuracy over time for $L = 10$. The horizontal axis represents training steps; the vertical axis represents accuracy on the training set.

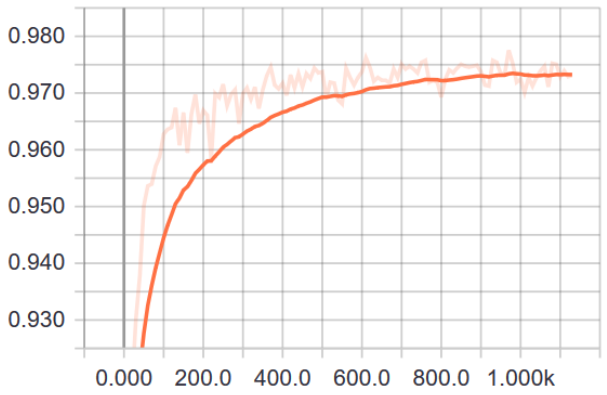

Figure 3: Validation accuracy over time for $L = 10$. The horizontal axis represents training steps; the vertical axis represents accuracy on the validation set.

The mean length of a non-clickbait hyperlink text was 2.74 tokens, whereas the mean length of a clickbait hyperlink was 12.49 tokens. The median length of non-clickbait hyperlinks was 1 token, whereas the median length of clickbait hyperlinks was 10 tokens.

With such a significant length difference between non-clickbait and clickbait, the presence of more than one or two words in a hyperlink text may be sufficient to indicate to the model that the text corresponds to clickbait.

Further inspection of the distribution of words in clickbait vs. non-clickbait text reveals additional insight into how the model distinguishes the two classes through word inputs. Figure 4 displays a histogram of the top 20 most common first words found in clickbait text. Together, these 20 words make up the first word of 20% of all clickbait text. Additionally, 34% of first words are numbers (tokens "1" and "5" are visible in the histogram) used in titles for "listicles" (list-based articles).

We observe that these top 20 first words are common to statements which invoke human curiosity by *suggesting* the presence of enticing content, rather than stating the content outright. Consider, again, the distinction between clickbait and an objective news headline. The two following examples (of similar token length) were present in our dataset, the first clickbait, the second non-clickbait.

How Everyone In The Audience Reacted When Kanye Pulled A Kanye

FDA recalls foreign meat used in aid for victims of Hurricane Katrina

The first headline makes a *promise* to provide information - an audience reaction to perhaps a celebrity scandal - but otherwise only provides the context that makes it enticing, creating a carrot-on-a-stick situation where the reader must make a deliberate action (clicking) to receive the reward (the audience's reaction). The second second headline provides the information immediately and makes no attempt to grab the reader's attention.

The word vectors provided as input from word2vec position words in vector space based on surrounding context. Thus, we expect the model has found distinct clustering of words that correspond to these carrot-on-a-stick sentences click-

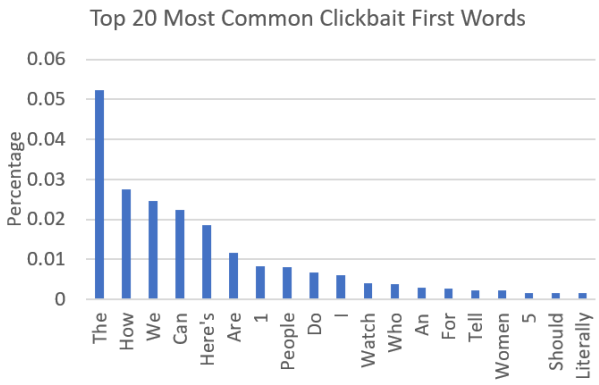

Figure 4: Top 20 most common first words used in clickbait titles. Together, these 20 words make up the first word of 20% of all clickbait text.

bait sentences, classifying them accordingly.

All of these insights suggest the first few words of a hyperlink are most important in deciding whether or not it contains clickbait. Correspondingly, our model was able to achieve 90% and 95% accuracy for $L = 2$ and $L = 3$ respectively.

4 Related Work

We came across several existing papers related to automated clickbait detection - two involving simple feature engineering and two utilizing deep learning.

(Chakraborty et al., 2016) evaluated the effectiveness of 14 structural, word level, N-gram and linguistic features like sentence length, hyperbolic words and PoS-Tags combined with three prediction models (SVM, Decision Trees, and Random Forests) in detecting clickbait, with a dataset of 7,500 sentences of each type (which we re-use in our dataset). They found that SVM with all 14 features performed the best, with 93% accuracy and 97% ROC-AUC scores. They also attempted to personalize the classifier for individuals, obtaining 81% accuracy on a hybrid topic-based and pattern-based classifier.

(Potthast et al., 2016) collected 215 features spread over 5 categories, evaluating their clickbait detection performance on three types of classifiers (Logistic Regression, Naive Bayes and Random Forest). Using all features with the Random Forest architecture gave an ROC-AUC score of 74%, and solely using N-Gram features seemed to outperform most other combinations of features used. The baseline ROC-AUC obtained was 54%, so there was a significant improvement.

(Anand et al., 2016), which much of our work is inspired by, compared three different Bi-RNN architectures (RNN, LSTM and GRU), each with character embeddings, word embeddings and a combination of both. Their model consisted of an Embedding Layer, a Hidden Layer and an Output Layer. Their dataset also re-used the 7,500 sentence dataset provided by (Chakraborty et al., 2016). After 10-fold cross validation, the BiLSTM (CE + WE) model obtained the best Accuracy, F1-Score and ROC-AUC (98.19%, 98.19% and 99.8% respectively).

(Agrawal, 2016) also attempted to solve the problem of clickbait detection using a deep learning model. (Agrawal, 2016)'s model model consisted of a Convolutional Neural Network (CNN) with a single layer of convolution, and the first layer was used to generate vector embeddings for the words (experiments were run using both embeddings developed from scratch as well as those obtained from word2vec). Evaluation was performed using 5-fold cross validation and Accuracy as well as ROC-AUC of 90% were obtained using word2vec embeddings.

All of the above papers produced effective models, but they were only applied to classifying news article headlines. The datasets used hence also reflected this limitation, and are not representative of the plethora of language expressions found on the internet. In addition, the intended applications of these papers were either undiscussed or restricted to specific websites, further limiting their applicability.

5 Future Work

In the future, we plan to develop a browser extension that incorporates our model so end users may automatically filter clickbait hyperlinks on the pages they browse. Currently, our model may not be lightweight enough (requiring the pretrained word2vec embeddings) to run exclusively locally; instead, a web service could be provided that performs classifications for the end user.

There are also several opportunities for us to improve our model. Currently, our model represents unknown words as all the same vector. We intend further experiment with how these words are represented, to determine if we can obtain better results with different representations. Incorporating support for sentence-level orthographic features (such as whether a title corresponds to a "lis-

ticle") into the input or output layer may also potentially yield better results.

6 Conclusion

In this paper, we developed a Bidirectional LSTM Deep-Learning model to detect clickbait hyperlinks on the web, using pre-existing distributed word embeddings (word2vec) to represent word inputs. Along with the dataset released by (Chakraborty et al., 2016), we built a recursive web scraper to collect approximately 50,000 clickbait and non-clickbait hyperlinks for training/testing. We found that our model performs extremely well in classifying hyperlinks as clickbait vs. non-clickbait, with our most powerful model achieving 98% accuracy, precision and recall. We also observed that the most significant information pertaining to a links classification can be found in the first four to ten words of the text.

In the future, we intend to develop a browser extension that uses our model to flag clickbait links on websites the user visits, in order to produce a better web-browsing experience. We also hope to extend our model with additional features and better representations of unknown words.

References

A. Agrawal. 2016. Clickbait detection using deep learning. pages 268–272.

Ankesh Anand, Tanmoy Chakraborty, and Noseong Park. 2016. We used neural networks to detect clickbaits: You won't believe what happened next! *CoRR*, abs/1612.01340.

A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. pages 9–16.

Xinlan Emily Hu. 2017. Clickbait and conscientiousness. https://www.stanforddaily.com/2017/03/20/clickbait-and-conscientiousness/. Accessed: 2018-05-07.

Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait detection. pages 810–817.