News Quality Scoring: an Ensemble of NLP Methods to Assess News Quality

Oday Najad
oday.najad@studenti.unipd.it

Tommaso Di Fant
tommaso.difant@studenti.unipd.it

Wageesha Widuranga
wageeshawiduranga.waththeliyanage@studenti.unipd.it

Abstract

The goal of this work is to assess various qualities of a news article trough the use of NLP techniques, including detecting clickbait titles, scoring the quality of writing, assessing the amount of information present, detecting AI-generated content, identifying plagiarized content, and verifying the correctness of information. The relevance of this work stems from the increasing prevalence of poor quality news articles that are often misleading or lacking in substance.

1. Introduction

Considering the ever increasing quantity of content on the internet it is crucial to have ways to carefully gauge the quality of a certain content without legacy scores such as interactions and read time, as many times articles present clickbait titles or long trivial content designed to increase the reading time without giving any information in the process. Now with Generative AI in the mix it becomes an even greater issue when taking into account all the generated and mostly information-less content present on the internet.

We propose a combination of NLP methods to assess various qualities of a news article, in hope to give reasonable scores to the article and being able to see whether it does not meet certain quality criteria. The qualities are more or less subjective, but all of them are human-annotated and should represent a certain preference coming from human beings and not a pre-determined algorithm like PageRank.

We assess 6 main aspects of a news article, and approach each of them as a separate task during the execution of the project. The tasks are the following: Detection of clickbait titles, Information density scoring, Writing quality scoring, Detection of AI Generated content, Detection of plagiarized content and finally Fact checking.

2. Related Work

Clickbait detection has been extensively studied due to the increasing prevalence of misleading headlines. Several approaches leverage machine learning techniques, such as support vector machines (SVMs) and neural networks, to classify clickbait content based on textual features [3]. Recent works have employed transformer models like BERT for more accurate classification by capturing contextual information in headlines [7].

For information density and summarization tasks, neural architectures like sequence-to-sequence models (e.g., LSTMs and GRUs) have been widely used [10]. More recently, pre-trained transformer models such as BART and PEGASUS have achieved state-of-the-art results in generating concise and informative summaries [8, 12].

The early approaches in plagiarism detection were cosine and Jaccard similarity measurements, which focus on lexical matching. Cosine similarity is based on TF-IDF weighting and compares document vectors. However, Jaccard similarity measures trigram overlap between documents [6]. To improve this, containment measures were introduced, which assess overlap with a reference set [2]. Longest Common Subsequence (LCS) techniques have also been employed to detect reordered content, though they struggle with synonyms and are computationally intensive [1].

Fact-checking in NLP has advanced with the creation of datasets like FEVER claims derived from Wikipedia [11]. Transformer-based models such as BERT [5], RoBERTa, and XLNet have been used for these tasks. BERT captures bidirectional context, RoBERTa improves upon BERT by removing the Next Sentence Prediction task and using dynamic masking, while XLNet introduces permutation-based training.

The Writing Quality scoring task is not prevalent in literature, with most work revolving around automated essay scoring to be an auxiliary task to students or professors in university or other lower grade schools. One previous relevant work is Automated Essay Scoring[4].

AI Content Detection has become a major field of research and readily available tools on the internet since the introduction of Generative AI. There are even open source pretrained models that solve this task, as we'll se in the dedicated section, such as DetectGPT[9].

3. Methods and Results

Here we present each task separately, outlining the datasets, models and training methods used as well as presenting the results achieved and briefly reviewing them.

3.1. Clickbait Detection

This report provides a detailed explanation of using the DistilBERT model for the task of clickbait detection. The goal is to classify headlines as either clickbait or non-clickbait using a pre-trained language model.

3.1.1 Datasets

The dataset used for this task is loaded from a CSV file named `clickbait_data.csv`. The dataset is first processed by checking for any missing values, and the columns are renamed to ensure uniformity.

3.1.2 Method

Data Tokenization The tokenization process uses the `DistilBertTokenizer` which converts the titles into input tokens and creates attention masks to indicate the relevant portions of each sequence.

Model Architecture The model used for classification is `DistilBertForSequenceClassification`. It is a lightweight version of BERT, which retains its performance while being more computationally efficient. The model is initialized using the weights from the `distilbert-base-uncased` pre-trained model.

Training Process The training process is handled by Hugging Face's `Trainer` class, which abstracts much of the complexity involved in training transformer models.

Evaluation Metrics Once the model is trained, it is evaluated on the test set. Evaluation metrics are used (accuracy, percision, recall and F1-score) In addition, a confusion matrix is generated to give a detailed breakdown of the model's performance in classifying the headlines as clickbait or non-clickbait.

3.1.3 Results

The evaluation of the model on the test dataset produced the following metrics:

- **Accuracy**: 0.9897
- **Precision**: 0.9900
- **Recall**: 0.9894
- **F1 Score**: 0.9897

Confusion Matrix and evaluation metrics The confusion matrix below provides additional insight into the classification performance:

The following bar plot provides a visual representation of the evaluation metrics, illustrating the high level of accuracy, precision, recall, and F1 score achieved by the model.

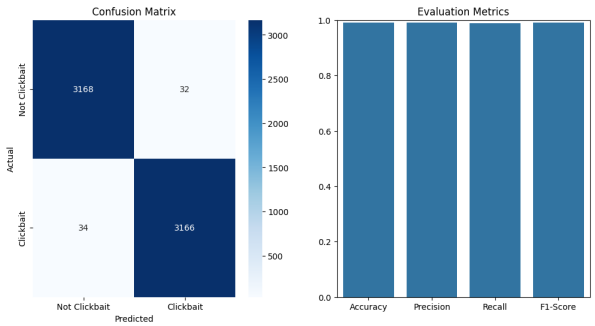

Figure 1: Confusion Matrix for the Clickbait Detection Model (left) and evaluation metrics bar plot (right)

3.2. Information Density

This report outlines the process of summarizing articles from the CNN/DailyMail dataset using the BART model for conditional text generation.

3.2.1 Datasets

The dataset used for this task is the `CNN/DailyMail` dataset, which is commonly used for summarization tasks.

3.2.2 Method

Summarization Using BART The summarization model used in this task is `facebook/bart-large-cnn`, a pre-trained version of BART specifically fine-tuned for summarizing news articles.

Evaluation Using ROUGE To evaluate the quality of the generated summaries, the ROUGE metric is employed. ROUGE measures the overlap between the generated summary and the reference summary by comparing the number of overlapping n-grams, word sequences, and word pairs. The ROUGE scores considered in this task are: ROUGE-1 (measures the overlap of unigrams between the generated

summary and the reference), ROUGE-2 (measures the overlap of bigrams between the generated summary and the reference and ROUGE-L (measures the longest common subsequence between the generated and reference summaries).

Visualization of ROUGE Scores The ROUGE scores are plotted for each article to visualize the distribution of the summarization performance. In addition, a bar plot is used to compare ROUGE-1, ROUGE-2, and ROUGE-L scores for a specific example summary.

3.2.3 Results

Summarization Output The BART model successfully generated summaries for the sampled dataset. Below is an example of an original article, its generated summary, and the corresponding ROUGE scores:

ROUGE Scores:

- **ROUGE-1**: 0.6500
- **ROUGE-2**: 0.3846
- **ROUGE-L**: 0.4000

Summary of ROUGE Scores The overall performance of the BART summarization model on the sample of 100 articles is summarized as follows:

- **Average ROUGE-1**: 0.3501
- **Average ROUGE-2**: 0.1527
- **Average ROUGE-L**: 0.2633

ROUGE Score Visualization Figure 2 shows the ROUGE-1, ROUGE-2, and ROUGE-L scores for each of the 100 sampled articles. The plot helps to visualize how consistent the model's performance is across different articles.

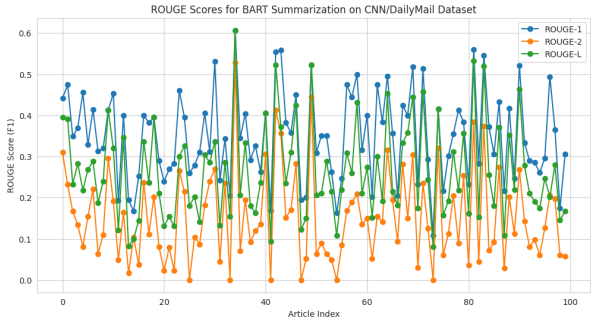

Figure 2: ROUGE Scores for BART Summarization on CNN/DailyMail Dataset

Additionally, a bar plot of the ROUGE scores for the generated summary of the example article is provided in Figure 3.

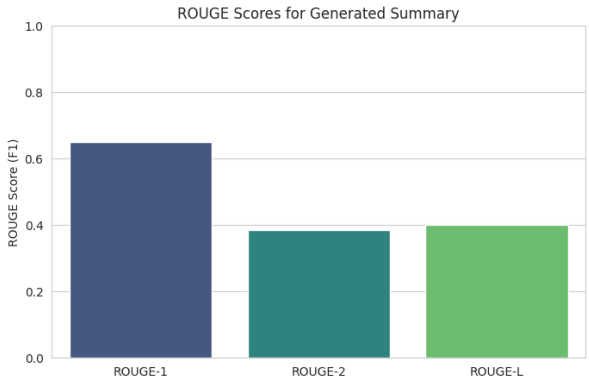

Figure 3: ROUGE Scores for Generated Summary (Example)

3.3. Writing Quality Assessment

To evaluate the quality of an article or essay we resort to datasets with human annotated scores, such as graded essays or scored text assessing some quality of it.

3.3.1 Datasets

The chosen datasets were *Automatic Essay Scoring 2.0 essay scoring* a 24k essay dataset each graded with a score from 1 to 6 based on its overall quality, *CLEAR* a 5k essay dataset each given a readability score, and finally *ELLIPSE* a 6k essay dataset where each essay is given 6 scores evalutating cohesion, syntax, vocabulary, phraseology, grammar, and conventions. These datasets are used together to have more general data over a single dataset, and they can provide meaningful scores for each of these measures (score, readability, cohesion, syntax, vocabulary, phraseology, grammar, and conventions) for any new text fed to the model.

3.3.2 Method

The *DistilBERT* model is used as a pretrained model and is fine tuned on the datasets presented. To train on these datasets a framework called MultiDataset-MultiTask training (called MDMT in the code), MultiDataset means concatenating all the datasets into one dataset and then sampling randomly from it until all samples have been exhausted.

MultiTask means that there are multiple target tasks to compute, for example estimating readability score, and each dataset is assigned on a task. Each task uses a different classification head and a potentially different loss from the other

tasks. In this case each dataset is assigned to a different task, the first and second dataset have a separate classification head with 1 output and both use Mean Square Error for their loss, the third dataset uses a 6-output classification head and also uses MSE loss.

For each batch the loss is computed separately for each sample according to their task specification and then averaged. Then since we are sampling the datasets together we will have a loss biased towards the more prevalent datasets so we inversely weigh the loss based on dataset size. Finally we add a final weight to balance the task's classification head output size with respect to the other task's heads, in this case the third task has 6 outputs compared to two 1-output heads for the other tasks, so we inversely weigh the loss on the task output width to balance the loss. When we refer to weighted mean square error for this task we mean the MSE loss calculated like this.

For fine-tuning we use a standard training setting with adamw as the optimizer, 1e-4 learning rate, 6 as the batch size and 1 gradient accumulation steps, we train for 2 epochs.

3.3.3 Results

The model is trained on 90% of the dataset and keep 10% for evaluation purposes. It reaches 0.044 weighted mean square error in the training set and 0.054 weighted mean square error on the evaluation set. These are much better compared to the 0.25 MSE we initially identified as the goal.

An important consequence of training on multiple dataset is that the model is able to generate all scores for all datasets, even if we didn't train to give a certain readability score (second dataset) to the samples of the first dataset, we now have a model that always outputs these 8 scores for all inputs, so we are able to give a readability score to the samples of the first or third dataset and viceversa.

This allows us to be rather confident that the model will generalize better on new text than a model trained on a single dataset, and averaging out the 8 scores will likely give a low variance estimate that can be useful in practice. Unfortunately there are no datasets available to check for generalization and that evaluation would require preferences given by human agents.

3.4. AI Content Detection

To detect whether a text is AI generated or human written is not an easy task, this is demonstrated by the fact that there are no very accurate detectors that work for any AI model available online, this is because each model is unique and there is no general fingerprint of an AI generated text. Moreover most AI models are trained to emulate human content so this task is even harder.

3.4.1 Datasets

To solve this task two datasets were used, first a small dataset *LLM — Detect AI Generated Text Dataset* comprising of 30k texts both AI generated or human written. Then a bigger dataset *artem9k/ai-text-detection-pile* is used, which contains 1.4 Million texts.

3.4.2 Method

For this task we use *DistilBERT* as a Sequence Classifier with a Binary Cross Entropy loss. We load the pretrained model and fine-tune it on the chosen dataset. First we train on the *LLM — Detect AI Generated Text Dataset* and as we'll see in the results section we get a good performance but little to no generalization on the same task in other datasets. So we also fine-tune another model on 5% of *artem9k/ai-text-detection-pile* dataset, reaching good performance on this dataset and improving performance on other datasets. For fine-tuning we use a standard training setting with adamw as the optimizer, 1e-5 learning rate, 1 as the batch size and 16 gradient accumulation steps, we train for 1 epoch.

3.4.3 Results

The first model trained on *LLM — Detect AI Generated Text Dataset* and reach a training BCE loss of 0.007 and validation loss of 0.026 showing slight overfitting but overall good performance on this dataset. Evaluating the model on the evaluation set gives us a F1 score of 0.99, but when evaluating on the evaluation set of a different dataset (*artem9k/ai-text-detection-pile*) the F1 score is very low.

The second model, trained on 5% of *artem9k/ai-text-detection-pile* achieves 0.187 BCE loss on the training set and 0.145 BCE loss and 0.95 F1 on the evaluation set. This model, without any further fine-tuning achieves 0.52 F1 score on *LLM — Detect AI Generated Text Dataset* which is only slightly better than random chance, but training on more data showed an increasing trend on this performance as training on only 1% of the dataset (instead of 5%) achieved 0.41 F1 score.

We also compared these results to an open-source pretrained AI Detector, that is 10x slower than this model, and that achieved 0.75 F1 on the first dataset and was too slow to calculate on the second one.

3.5. Plagiarism Detection

Plagiarism detection is a critical aspect of ensuring the originality of academic and professional writing. This section describes the integration of a plagiarism detection task into our existing framework.

3.5.1 Datasets

For the plagiarism detection task, the Webis-CPC-11 dataset, which is designed for specific tasks, was used. It contains 7,859 candidate paraphrases obtained from Mechanical Turk crowdsourcing. The corpus is made up of 4,067 accepted paraphrases, 3,792 rejected non-paraphrases, and the original texts. These samples have formed part of the PAN 2010 international plagiarism detection competition. Input text was tokenized and converted to lowercase, and the stop words and punctuation were removed to obtain a clean list of significant words. Trigrams were created from preprocessed tokens, which are sequences of three consecutive words capturing some contextual information within the text.

3.5.2 Method

Cosine Similarity In the field of Natural Language Processing (NLP), Term Frequency (TF) measures the frequency of terms in a document, while Inverse Document Frequency (IDF) measures the uniqueness or overlap of a word across the entire corpus. TF-IDF calculates a weight for each word by considering how important it is within a particular text and its rarity across the entire collection of documents. Cosine Similarity is a method that analyzes two documents by computing the cosine of the angle between their TF-IDF vectors in a multi-dimensional space. It emphasizes the direction of the vectors rather than their magnitude.

Jaccard Similarity The Jaccard Similarity measures the proportion of shared trigrams (intersection) relative to the total number of unique trigrams (union) between two sets. In NLP tasks, typically sets of trigrams extracted from text documents or sentences are used in this method. It considers both the common trigrams and the total number of unique trigrams in the two sets. The index is low if the sets have little overlap relative to their combined size.

Containment Measure This approach has similarities to Jaccard Similarity. It also uses the same set of trigrams, but instead of measuring the ratio between the union set, it measures the ratio in relation to a specific reference set. This measure highlights the extent to which one set is included by another. It is especially advantageous when one set is significantly smaller than the other or when you have a specific interest in determining the extent of overlap between the two sets.

Longest Common Subsequence (LCS) This is a technique used in text similarity analysis to identify the longest sequence of words that appear in the same order in both texts, even if they are not consecutive. The LCS method uses dynamic programming to construct a matrix for comparing sequences and identifying the longest matching subsequence. This makes it particularly useful for applications like plagiarism detection and text alignment, where the order of words is important. However, LCS is computationally intensive and only matches exact words, making it less effective with synonyms or flexible phrasing.

3.5.3 Results

For each set containing original and paraphrased or non-paraphrased documents, the above-mentioned similarity scores were calculated.

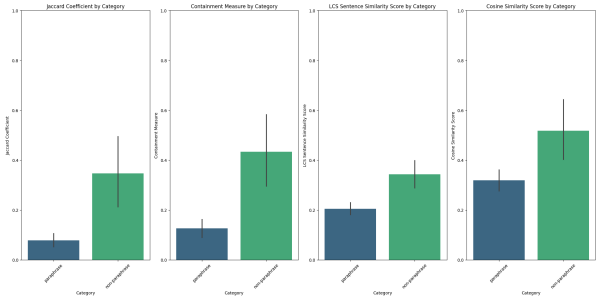

Figure 4: Plagiarism detection for two categories

The average similarity scores for each category were plotted in order to compare the performances of the different methods. The results suggested that the similarity scores of the non-paraphrased documents were higher compared to paraphrased documents across all methods. The highest average similarity scores were obtained by the Cosine Similarity method, while the lowest were obtained by the Jaccard Similarity method.

3.6. Fact Checking

3.6.1 Datasets

The set of 185,445 claims known as FEVER (Fact Extraction and Verification) was created by modifying statements taken from Wikipedia and then verified without the knowledge of the original text. The statements are categorized as Not Enough Information, Refuted, or Supported. The sentence or sentences providing the essential support for the annotators' decision were also noted for the first two classes.

Data Preprocessing Text data must be cleaned and normalized by standardizing whitespace, changing it to lowercase, and eliminating non-alphanumeric characters. For consistency among datasets, it also entails encoding category labels into numerical values. In order to enable efficient training and evaluation, this preparation is essential for

ensuring that the text data and labels are appropriate for machine learning models. For the model training ,claims were taken with relavent evedence sentences and Not Enough Information category was removed because there wasn't relevent evedence to support of refuse the claim.

3.6.2 Method

Three different models were used to compare the performance. BERT, RoBERTa, and XLNet are the three different models that were used in this task. These models all leverage the Transformer architecture, which enables them to capture complex patterns and relationships in text through attention mechanisms. They are pre-trained on large text corpora to learn contextual representations of words, which are then fine-tuned for specific NLP tasks.

BERT The BERT model prioritizes the left and right context of every word in a phrase because it is designed with a bidirectional transformer architecture. It can identify additional word meanings depending on the context in which they appear.

RoBERTa RoBERTa improves upon BERT's architecture in several ways. The Next Sentence Prediction (NSP) task is removed, and more data with larger batch sizes and learning rates are used for training. Moreover, RoBERTa employs dynamic masking during training, meaning that every epoch has a different masking pattern.

XLNet XLNet combines the ideas of the BERT model with a permutation-based training model, where it learns to predict the order of the words in a sentence instead of just masked words. This approach captures bidirectional context and maintains autoregressive properties.

3.6.3 Results

The results were compared across the above-mentioned methods, and the accuracies were nearly equal for every model. The best accuracy was shown by BERT, which was around 0.96. The accuracy was improved with some adjustments to the training parameters, such as learning rate, weight decay, batch sizes, and training epochs. However, running these models on larger datasets with limited computational power can be challenging, as it becomes more time-consuming and resource-intensive. The confusion matrix for the above-mentioned methods is provided below.

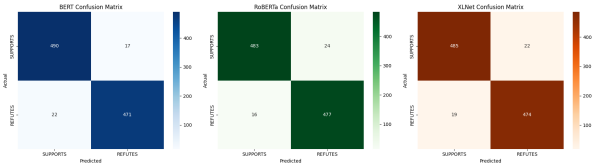

Figure 5: Confusion matrix for BERT, RoBERTa and XLNet models

4. Conclusion

Therefore, this paper offers an eclectic of NLP techniques for the assessment of several aspects of the quality of news articles such as clickbait, information density, other compositional aspects, artificial content identification, plagiarism check, and factual accuracy check. The intent of integrating these methods is to come up with a multiple and comprehensive analysis of news content, given that it is more challenging for those who analyze news to get high quality and credible articles in current media. As shown from the previous models and approaches, it yields promising results, but for best generalization and application more work is required on optimizing and perhaps introducing new approach on diverse data sets. It sets the base for further improvements in the development of automated news quality assessment.

References

[1] L Bergroth, H Hakonen, and T Raita. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, A Coruna, Spain, 2000*, pages 39–48. IEEE, 2000.

[2] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, pages 21–29. IEEE, 1997.

[3] Abhijnan Chakraborty, Bhargavi Paranjape, Sneha Kakarla, and Niloy 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)*, pages 9–16. IEEE, 2016.

[4] Hongbo Chen, Jungang Xu, and Ben He. Automated Essay Scoring by Capturing Relative Writing Quality1. *The Computer Journal*, 57(9):1318–1330, 10 2013.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[6] Wahid H Gomaa and Aly A Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 2013.

[7] Ankita Gupta, Gurjot Kaur, Gagan Dhiman, and Monika Juneja. Deep learning-based clickbait detection using bert. *Journal of Emerging Technologies and Innovative Research*, 7(7):156–162, 2020.

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2020.

[9] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature.

[10] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.

[11] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, 2018.

[12] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*, 2020.

Table 1: Task Distribution

Name	**Related Work**	**Implementation**	**Tasks**	**Report**	**Presentation**
Oday Najad	13h	30h	Clickbait Detection Information Density	5h	5h
Tommaso Di Fant	15h	25h	AI-generated Content Detection Quality of Writing	5h	5h
Wageesha Widuranga	15h	25h	Fact Checking Plagiarism Detection	5h	5h