International Journal of Cyber Behavior, Psychology and Learning
Volume 13 • Issue 1

Comparison of Artificial Decision Techniques for Detection of Sarcastic News Headlines

Tarun Jain, Manipal University Jaipur, India
Horesh Kumar, G.L. Bajaj Institute of Technology and Management, Greater Noida, India
https://orcid.org/0000-0002-7783-122X
Payal Garg, G.L. Bajaj Institute of Technology and Management, Greater Noida, India
Abhinav Pillai, Manipal University Jaipur, India
Aditya Sinha, Manipal University Jaipur, India
Vivek Kumar Verma, Manipal University Jaipur, India*

ABSTRACT

Newspapers are a rich informational source. A headline of an article sparks an interest in the reader. So, news providing agencies tend to create catchy headlines to attract the reader's attention onto them, and this is how sarcasm manages to find its way into news headlines. Sarcasm employs the use of words that carry opposite meaning with respect to what needs to be conveyed. This leads to the need of developing methods by which we can correctly predict whether a piece of text, or news for that matter, truthfully means what it says or is simply being sarcastic about it. Here, the authors have used a dataset containing 55,329 tuples consisting of news headlines from *The Onion* and the *Huffington Post*, which was taken from Kaggle, on which they applied feature extraction techniques such as Count Vectorizer, TF-IDF, Hashing Vectorizer, and Global Vectorizer (GloVe). Then they applied seven classifiers on the obtained dataset. The experimental results showed that the highest accuracies among the ML models were 81.39% for LR model with Count Vectorizer, 79.2% for LR model with TF-IDF Vectorizer, and 78% for SVM model with Count Vectorizer. They also obtained the best accuracy of 90.7% using the Bi-LSTM Deep Learning Model. They have trained the seven models and compared them based on their respective accuracies and F1-Scores.

KEYWORDS

Classification, Decision Tree Model, KNN, Logistic Regression Model, Machine Learning, Naive Bayes Model, Random Forest Model, SVM Model

1. INTRODUCTION

Newspapers are a rich informational source which provides us with a clear idea and understanding of what is happening in and around the world, on a daily basis. It is one of the greatest means of

DOI: 10.4018/IJCBPL.330131 *Corresponding Author

communication between people and the world, due to its high availability and low cost. Even though they have very detailed articles, more often than not, it is the Headline of an article which sparks an interest in the reader. So, news providing agencies tend to create catchy headlines to attract the reader's attention onto them, and this is how Sarcasm manages to find its way into News Headlines.

Sarcasm employs the use of words that carry opposite meaning with respect to what needs to be conveyed. This may be done to criticize someone or something, to show annoyance or just to be funny. For instance, using "They're really on top of things" to describe a group of people who are disorganized, is sarcasm. Sarcasm when vocally expressed is easier to detect due to the added audiovisual cues which the speaker may showcase, such as a mocking tone, exaggerated expression using hands, or even a roll of the eyes. But the same task becomes significantly more difficult when presented in a textual format without these cues- when all the reader has as information to detect sarcasm lies in the text itself, often mentioned without any context. That is where genuine news tends to become fake news if the reader wrongfully guesses the content while skimming through the headlines of the newspaper- as most studies show that only a sliver of the population actually reads the full contents of a newspaper, and not just the headlines.

This leads to the need to develop methods by which we can correctly predict whether a piece of text, or news for that matter, truthfully means what it says or is simply being sarcastic about it. That is where Sarcasm Detection using Natural Language Processing comes in as an attempt to develop a data processing model which can correctly guess, whether a piece of text is sarcastic or not. There is no 100% accurate model as of now, and thus, researchers have been trying to create most accurate models using a variety of methods in various parts of the process. Our paper is one of them.

We agree with Pelser & Murrell (n.d.) on the fact that sarcasm in text tends to have no fixed structure as such, and so, implement a new approach to data pre-processing, in an attempt to improve upon whatever structure is already present. With respect to this, we do not perform stop-word removal during pre-processing and add on further to the structure by converting numbers to words instead of outright removing them. This gives us an increase in accuracy up to 11.24%. Our goal is to analyze the effectiveness of sarcasm detection with various models in order to find the ones that work best for a given dataset pre-processed as mentioned.

A dataset containing 55,329 tuples consisting of news headlines from The Onion and the Huffington Post, was taken from Kaggle (n.d.). A balanced subset of it was used for the ML based models- consisting of 10,000 tuples which were pre-labeled as 1 for sarcastic and 0 for non-sarcastic. Similar subsets having 15,000, 20,000 and 25,000 were derived for the Neural Network based models. The 55k tuple data was also used in the research.

2. RELATED WORK

There were 3 papers which made use of the same dataset as ours in their respective approaches. Onyinye & Afli (2020) used word level, n-gram level and character level TF-IDF embedding combined with Count Vectorizer for their research, concluding that supervised learning techniques fare better than deep learning methods for sarcasm detection. Jariwala (2020) used Part-of-Speech tagging to determine connections between words and utilized a rule-based approach to extract optimal features for their research. Their approach gave better results for SVM classifier compared to standard feature extraction. Pelser & Murrell (n.d.) used a Deep and Dense Neural Network to extract additional intrinsic information from the text for making better predictions for standalone utterances.

Sarcasm detection models implemented by different researchers primarily differ in the feature extraction and embedding process. As such, quite a few novel approaches have been invented which generate better results than traditional approaches. Babanejad & Davoudi (2020) developed novel deep learning models which extend the architecture of Bidirectional Encoder Representations from Transformers (BERT) by incorporating affective and contextual features extracted using a Bidirectional Long Short-term Memory (Bi-LSTM) model with multi-head attention. Khatri (2020) used suitable

pre-trained BERT and Global Vector (GloVe) word embedding techniques on their data. Bharti et al. (2020) extracted interjection words and intensifiers using Part-of-Speech tagging to build the feature set for 5-fold cross validation. Pan et al. (2020) utilized positive-negative incongruities, generally observed in sarcastic sentences, as a part of their feature extraction.

Patro (2019) proposed an interesting approach based on an observation. Sentences having sarcasm in them have words which belong to one of two clusters- sarcastic targets and non-sarcastic targets, depending on the values according to their Local and Organization named-entity distribution and Part-of-Speech distribution. They utilized these features using LIWC and Empath (Gupta, 2020) category fractional distribution dictionaries for their research. Dubey et al. (2019) utilized the incongruity of sarcastic sentences using exaggerated or comparative numerical values using rule-based, statistical machine learning and deep learning approaches for their classification. Rajadesingan (2015) categorized sarcastic sentences based on the function of sarcasm- a complex form of expression, a means of conveying emotion, a possible function of familiarity or as a form of written expression and evaluated them using the SCUBA framework. They procured higher performance than standard contrast-seeking frameworks. Jaiswal (2020) applied an ensemble strategy based on models trained on different length conversational texts to make more accurate predictions. Last but not the least; Gupta (2020) used a rule-based approach on the basis of the sentiment score of various features in data for their research.

3. METHODOLOGY

We acquired the labeled dataset on news headlines from Kaggle (n.d.) and utilized Jupyter Notebook as the stage to code. Our approach includes utilization of various classification systems such as Support Vector Machine (SVM), K-Nearest

Neighbors(K-NN), Logistic Regression, Decision Tree, Gaussian Naïve Bayes, Convolutional Neural Networks (CNN) and Bi-directional Long Short-term Memory.

3.1 Dataset

The datasets used for the proposed approaches were refined and unnecessary columns were removed. The refined datasets had the following attributes:

3.1.1 Attribute Information

1. Article link- A unique ID
2. Headline- Text to be classified
3. Is sarcastic- 1 for Sarcastic, 0 for Not

3.2 Data Pre-Processing

Information pre-processing is an information mining system involving the conversion of raw information into a workable form. The text in the datasets was first converted to lowercase form in order to achieve uniformity of words. The text contained lots of contraction terms and thus, they are expanded using necessary modules. Numerical data is usually cleaned out during pre-processing. However, in order to retain more of a structure for more accurate sarcasm detection, the numerical data was converted to words using the num2words module. Last but not least; non-alphanumeric characters were removed from the text. The removal of stop words and stemming or lemmatization of text significantly reduces the structure of the text and thus, they are not applied. This decision improves the final accuracies of implemented models by up to 11.24%, compared to that for conventionally pre-processed data. This increase in accuracy for each type of classification done through each type of feature extraction is shown in Table 2.

3.3 Feature Extraction

a) **Count Vectorizer:** The Count Vectorizer tokenizes and generates a vocabulary of unique words from the data. It produces an output for each tuple containing the number of times the words of the vocabulary appeared in it, in a 1-dimensional matrix form. It is implemented using the Scikit-Learn module.
b) **TF-IDF Vectorizer:** The TF-IDF Vectorizer tokenizes the words of the data based on their frequency and relevancy in the dataset. TF stands for Term Frequency and is the number of times a word appears in a document. In order to equalize the value across documents of differing lengths, the value is normalized:

$$\boldsymbol{TF}\left(\boldsymbol{t}\right)=\frac{\boldsymbol{N}}{\boldsymbol{T}}$$

Where,N = Total frequency of t in a document.T = Total number of terms in the document.

IDF stands for Inverse Document Frequency and is a measure of the relevancy or importance of a term. This is done to reduce the influence of words which repeat many times but have no real meaning or importance in the text. This value is also normalized for effective usage:

$$IDF\left(t\right)=log_{e}\frac{TD}{ND}$$

Where,TD = Total number of documentsND = Number of documents with t in them

It produces output values containing the product of the TF and IDF values for each word in the data and has been implemented using Scikit-Learn.

c) **Hashing Vectorizer:** The Hashing Vectorizer converts a collection of data into a sparse matrix based on token occurrence counts, by mapping the data to the fixed dimensions of the matrix. Unlike Count Vectorizer, it does not store the resulting vocabulary and thus, works faster for larger datasets, but loses the actual values of the tokens. It is implemented using Scikit-Learn.
d) **Global Vectors (GloVe):** GloVe is a word vector Technique that leverages both global and local Statistics of a corpus to come up with a principal loss function which uses both of them. Put simply, it generates a co-occurrence matrix from which the semantic relationships between words can be derived.

The cost function for GloVe is as follows:

$$J=f\left(X\_ij\right)(w\_i\hat{}Tu\_j+bu\_j-\ log\left(X_{i}j\right)^{2}$$

Where, $f\left(X\_ij\right)=\ (x\ /\ x\_\left\{max\right\})\hat{}\ a\ if\ x<x\_\left\{max\right\}else\ 0$ bw, bu = biases of the network

For this paper, a pre-trained 100-dimensional model with 6 billion words is implemented and fine-tuned accordingly.

3.4 Classification Algorithms

a) **Gaussian Naïve Bayes:** The Gaussian Naïve Bayes is a probabilistic classification algorithm based on the Bayes Theorem in statistics. It assumes that the continuous values associated with each extracted feature are distributed according to a Gaussian distribution.

In Gaussian Naïve Bayes, given the class designation, it is presumed that each feature operates independently of the others. This presumption makes computing probabilities easier and enables the algorithm to operate well even when dealing with enormous datasets. The procedure determines the likelihood that a data point belongs to each class, then assigns it to the class where the likelihood is highest.

Since the likelihood of features is assumed to be Gaussian,

The conditional probability for each x such that y occurs is given by:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}$$

This classifier is implemented using Scikit-Learn and works faster compared to its more complicated counterparts.

b) **Decision Tree Algorithm:** The Decision Tree algorithm predicts the class or value of the target variable by learning simple decision rules, inferred from prior data. It generates a decision tree by splitting the nodes on all available variables and then selecting the node which results in the most homogeneous sub-nodes, using a greedy approach, i.e., a function that returns a locally optimized solution. It is implemented using Scikit-Learn.

The algorithm works by recursively splitting the dataset based on the features that best separate the data points according to a certain criterion, such as Gini impurity or information gain. The leaf nodes in the tree reflect the final class or regression forecast, whereas each internal node represents a choice based on a particular attribute.

c) **Logistic Regression:** The Logistic Regression classifier employs a generalized linear model and a loss function that minimizes the results to values between 0 and 1. The input values are combined linearly using weights topredict the output values. The cost function is given by:

$$\mathrm{Cost}\left(\mathrm{h}_{\theta}\left(x\right), y\right) = \begin{cases} -\log\left(h_{\theta}\left(x\right)\right) & \text{if} \quad y = 1 \\ -\log\left(1 - h_{\theta}\left(x\right)\right) & \text{if} \quad y = 0 \end{cases}$$

It is implemented using Scikit-Learn and gives the Second highest accuracy for the data when the features are extracted using Count Vectorizer.

d) **Support Vector Machines:** The Support Vector Machine Classifier computes a hyperplane in an N-dimensional Space that best separates the two classes of data.

The fundamental principle of SVMs is to transform the input data into a higher-dimensional feature space, where a linear decision boundary may efficiently separate the classes. By using kernel functions, which intuitively translate the data into a higher-dimensional space without explicitly calculating the transformations, this transformation is made possible.

It is implemented in a linear fashion using Scikit-Learn.

e) **K-Nearest Neighbors:** The K-Nearest Neighbors is a non-parametric classification algorithm in the sense that it does not make any underlying assumptions about the data. It plots the classes on a graph and then tries to classify the test data based on the classes of its k nearest neighbors (where k = 1, 2, 3, …).

The basic idea behind KNN is to classify or predict the value of a data point based on its proximity to other known data points. The method determines how far each new data point is from every other point in the dataset whenever a new data point is provided.

The label or value of the new data point is then determined by using the labels (in classification) or values (in regression) of the K closest neighbors, which are selected based on distance.

f) **Convolutional Neural Network:** CNN is a Neural Network that has one or more convolutional layers i.e., filters over the input data that help make better predictions. These filters, when used appropriately, can convert complex, multi-dimensional input data into a simple 1-dimensional form.

The convolutional layer is a CNN's most important part. The network can automatically extract pertinent features from the images thanks to this layer's application of a set of trainable filters to the input data. The CNN can recognise complex patterns and objects at many scales thanks to these filters' ability to capture local patterns and spatial correlations.

It is implemented using Keras.

g) **Bidirectional Long Short-Term Memory:** Bi-LSTM is a Sequence processing model that employs two LSTMs- working in forward and backward directions. The back pass allows the neural network to effectively increase the amount of data available to it and also optimize its cost function at each iteration. It is implemented using the Keras module (Abercrombie & Hovy, 2016; Baruah et al., 2020; Fast et al., 2016).

Information only moves from past to future timesteps in a typical LSTM. However, BLSTM can offer significant advantages in jobs where future context is crucial, such as speech recognition or natural language processing. It consists of two independent LSTM layers, one of which moves the sequence forward while the other moves it backward (Campbell & Katz, 2012; Chaudhari & Chandankhede, 2018; Devlin et al., 2019; Ghosh et al., 2018).

By processing the sequence in both directions, BLSTM can leverage future information to enhance its predictions at each timestep. This bidirectional flow allows the network to capture long-term dependencies and understand the context in a more comprehensive manner.

4. EXPERIMENTS AND RESULTS

4.1 Evaluation Metrics

The evaluation metrics were derived from the confusion matrices generated by the models used. The structure of a confusion matrix is shown in Table 1.

Table 1. Structure of a confusion matrix

Actual Values / Predicted Values	Positive (1)	Negative (0)
POSITIVE(1)	TP	FP
NEGATIVE(0)	FN	TN

They are as follows:

Accuracy: It represents the number of correctly classified data instances over the total number of data instances. It is computed by:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Precision: It represents the number of positive class predictions that belong to the positive class. It helps evaluate models when the costs associated with false positives is high. It is computed by:

$$Precision = \frac{TP}{TP + FN}$$

Recall: It represents the number of negative class predictions that belong to the negative class. It helps evaluate models when the costs associated with false negatives are high. It is computed by:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: It is the weighted average of the Precision and Recall of a model. It is computed by:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4.2 Experiments

Each of the Machine Learning based models were fed features extracted through Count Vectorizer, TF-IDF Vectorizer and Hashing Vectorizer, and their corresponding confusion Matrices were generated for evaluation.

The Neural network-based models used pre-trained GloVe Embeddings for feature extraction, and were trained on 5 different pre-processed balanced datasets of sizes 10k (the same used for ML models), 15k, 20k, 25k and 55k. This process could not be followed for the other models due to hardware limitations.

4.3 Results

The comparison of the before mentioned evaluation metrics, for the ML based models has been shown in Table 2. The Increase in Accuracy column showcases the increased accuracy with the applied

pre-processing as opposed to that implemented with stop-word removal and removal of numeric data from the data.

a) Comparison of Machine Learning Models

Here we have used 5 Machine Learning Models named Gaussian Naïve Bayes, Decision Tree, Logistic Regression, Support Vector Machine and KNN.

Gaussian Naïve Bayes uses the Bayes theorem to determine the likelihood of each class and assumes that the features in the dataset are independent of one another. Decision Tree works by Recursively dividing the data into subsets according to the importance of the input features, it then bases its choices on the resulting tree structure.

Logistic Regression is a well-known method applied to classification issues. It functions by modelling the likelihood of the output class as a function of the features provided as input and making predictions based on a threshold value. SVM uses a kernel function to translate the input data into a higher-dimensional space, if necessary, then finds a hyperplane that maximally divides the various classes in the dataset. KNN makes predictions based on the most prevalent class or average value of the neighbors by locating the k closest neighbors of a given input data point in the dataset.

The highest accuracies among the ML models were 81.39% for LR model with Count Vectorizer, 79.2% for LR model with TF-IDF Vectorizer and 78% for SVM model with Count Vectorizer. Their corresponding F1-Scores were 81.61%, 79.28% and 77.94% respectively. Results of ML based models using different feature extraction models are shown in Table 2.

b) Comparison of Deep Learning Models

Table 2. Table showing results of ML based models (the best result has been highlighted ion green)

Classifier	Feature Extractor	Accuracy	Precision	Recall	F1 Score	Increase in Accuracy
GAUSSIAN NAIVE BAYES	COUNT VEC	66.52	50.23	74.92	60.16	3.40%
	TF-IDF VEC	63.83	54.46	68.27	60.59	1.11%
	HASHING VEC	57.8	60.73	58.31	59.5	4.68%
DECISION TREE	COUNT VEC	73.48	77.74	71.85	74.68	3.56%
	TF-IDF VEC	73.48	77.74	71.85	74.68	3.56%
	HASHING VEC	53.32	55.24	53.5	54.36	0.40%
LOGISTIC REGRESSION	COUNT VEC	81.39	82.03	81.19	81.61	6.07%
	TF-IDF VEC	79.2	79.09	79.47	79.28	7.40%
	HASHING VEC	55	54.84	55.33	55.08	1.17%
K NEAREST NEIGHBORS	COUNT VEC	62.4	48.72	67.51	56.6	11.24%
	TF-IDF VEC	49.72	0.08	100	0.16	0.04%
	HASHING VEC	56.87	57.23	57.14	57.18	2.87%
SUPPORT VECTOR MACHINES	COUNT VEC	78	77.26	78.64	77.94	8.33%
	TF-IDF VEC	77.24	76.39	77.93	77.15	7.96%
	HASHING VEC	55.96	58.93	56.58	57.73	2.16%

Here we have used two Deep Learning techniques(algorithms) named CNN (Convolutional Neural Network) and Bi-LSTM (Bidirectional Long Short-Term Memory).

In text based CNNs, instead of processing image pixels, the input is usually in the form of word embeddings, which are representations of words as vectors. The convolutional layers in the CNN architecture learn to extract features from these word embeddings, which capture important information about the meaning and context of the words in the text.

Natural language processing (NLP) activities including sentiment analysis, machine translation, and audio recognition frequently make use of bi-LSTMs. Bi-LSTM models are used in NLP to effectively model context and long-term dependencies in the data by capturing the temporal dependencies and relationships between words in a phrase. Bi-LSTMs can learn from input data in both the forward and backward directions since they are bidirectional, which enables them to learn from both the past and future contexts of each word in the sequence (Joshi et al., 2016; Kumar et al., 2020; Misra & Arora, 2019; Porwal et al., 2018).

The comparison between the Neural Network based models for the varying dataset sizes has been shown in Table 1. The highest accuracies of both Bi-LSTM and CNN were obtained for the dataset with 55k tuples. They were 90.7% and 85.9% respectively. Their accuracies were 83.28% and 76.4% for the dataset common to both ML and Neural Network models. The accuracy for these models increased by an average of 2% for the applied pre-processing. Results of CNN and Bi-LSTM when working with datasets of varying sizes are shown in Table 3. Scatter plot showing increase in accuracy with increase in size of dataset for CNN (blue) and Bi-LSTM (red) is shown in Figure 1.

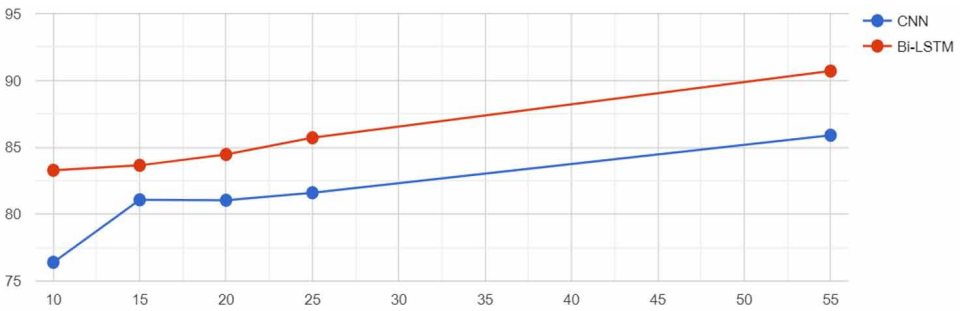
5. DISCUSSION

The detection of sarcasm in news headlines using NLP is a challenging task due to the complexity and ambiguity of language. However, by using various cutting-edge NLP techniques, researchers have made great advancements in this field. We've covered a variety of methods for sarcasm detection in this paper, including rule-based, machine learning, and deep learning techniques. The identification of sarcasm in news headlines requires feature engineering. We can significantly improve the performance

Table 3. Table showing results of CNN and Bi-LSTM when working with datasets of varying sizes. The best results have been highlighted in green.

	10k	15k	20k	25k	55k
CNN	76.4	81.07	81.04	81.6	85.9
Bi-LSTM	83.28	83.65	84.46	85.71	90.7

Figure 1 Scatter plot showing increase in accuracy with increase in size of dataset for CNN (blue) and Bi-LSTM (red)



of sarcasm detection models by including features like negation detection and sentiment analysis (Jones, 2021; Mikolov, Chen, Corrado et al, 2013; Mikolov, Sutskever, Chen et al, 2013; Son et al., 2019; Vaswani et al., 2017).

Our analysis showed that machine learning and deep learning approaches have demonstrated higher accuracy rates. Specifically, we explored the use of Bi-LSTM and convolutional neural networks (CNNs) for sarcasm detection in news headlines and found that they have shown remarkable performance in detecting sarcasm.

Sarcasm identification in news headlines using NLP has a wide range of possible uses. Political campaigns can employ sarcasm detection to recognize and respond to satirical news items, while media corporations can use it to enhance brand sentiment analyses and marketing strategies. Sarcasm identification can help social media analysts better comprehend the tone of user-generated material. However, there are some restrictions that we might face in our research that must be taken into consideration. Lack of a proper dataset for sarcasm detection in news headlines is one restriction, which may have an impact on the precision of findings. The development of larger and more varied datasets for sarcasm detection in news headlines can be the subject of future study.

In conclusion, sarcasm detection in news headlines using NLP is an important field of study with numerous real-world applications. We can anticipate seeing increasingly more advanced and precise approaches for sarcasm detection in news headlines as NLP algorithms and methodologies continue to advance.

6. CONCLUSION

Here we have used Natural Language Processing (NLP) methods for sarcasm detection in news headlines. Due to the complicated and frequently confusing nature of language, sarcasm detection in NLP is a difficult issue. However, academics have achieved great advancements in this field thanks to the creation of complex NLP algorithms.

We used various rule-based and machine learning techniques such as Naive Bayes algorithm, Decision Tree algorithm, Logistic Regression algorithm, SVM technique and KNN algorithm, for sarcasm identification. Additionally, we also obtained the results by applying deep learning methods, such as CNN and BI-LSTM, for sarcasm detection in news headlines, including recurrent neural networks and convolutional neural networks. The dataset consisted of 11 attributes and after applying all 5 algorithms, Logistic Regression gave the maximum accuracy of 96.5% using K fold method to calculate accuracy and K nearest neighbor with efficiency of 98% using confusion matrix, so it is good to use these algorithms for prediction.

The use of NLP for sarcasm detection in news headlines has an important effect on several businesses, including politics, media, and advertising. Sarcasm identification can be used to enhance brand sentiment analysis, spot bogus news, and support political campaign analysis. Additionally, the development of effective NLP algorithms for sarcasm detection in news headlines can advance sentiment analysis and natural language processing.

In conclusion, sarcasm detection in news headlines using NLP is a critical area of research with numerous real-world applications. We can expect to see increasingly more advanced and accurate approaches for sarcasm detection in news headlines as NLP algorithms and methodologies continue to advance.

REFERENCES

AbercrombieHovy. (2016). *Putting Sarcasm Detection into Context: The Effects of Class Imbalance and Manual Labelling on Supervised Machine Classification of Twitter Conversations*. ACL. https://aclanthology.org/P16-3016

Babanejad, N., & Davoudi, H. (2020). Affective and Contextual Embedding for Sarcasm Detection. *International Conference on Computational Linguistics.*

Baruah, A., Das, K., Barbhuiya, F., Dey, K. (2020). *Context-aware sarcasm detection using BERT*. 10.18653/v1/2020.figlang-1.12

Bharti, S. K., Naidu, R., & Babu, K. S. (2020). Hyperbolic Feature-Based Sarcasm Detection in Tweets: A Machine Learning Approach. IEEE Xplore.

Campbell, J. D., & Katz, A. N. (2012). Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, *49*(6), 459–480. doi:10.1080/0163853X.2012.687863

Chaudhari, P., & Chandankhede, C. (2018). Literature survey of sarcasm detection. *Proceedings of the 2017 International Conference on Communications and Signal Process Networking, WiSPNET 2017*, 2041–2046. doi:10.1109/WiSPNET.2017.8300120

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019-Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings Conference,* 1, 4171–4186.

Dubey, A., Kumar, L., Soumani, A., Joshi, A., & Bhattacharyya, P. (2019). Where Numbers Matter! Detecting Sarcasm in Numerical Portions of Text. ACL.

Fast, E., Chen, B., & Bernstein, M. S. (2016). Empath: Understanding Topic Signals in Large Scale Text. CHI.

Ghosh, D., Fabbri, A. R., & Muresan, S. (2018). Sarcasm analysis using conversation. *Context (Ibadan)*. Advance online publication. doi:10.1162/coli

Gupta, R. (2020). *A Statistical Approach for Sarcasm Detection using Twitter Data*. ICICCS.

Jaiswal, N. (2020). *Neural Sarcasm Detection using Conversation Text*. ACL.

Jariwala, V. P. (2020). Optimal Feature Extraction based Machine Learning Approach for Sarcasm type Detection in News Headlines. International Journal of Computer Applications.

Jones, S. (2021). A statistical interpretation of term specificity and its application in retrieval. *The Journal of Documentation*, *60*(5), 493–502. doi:10.1108/00220410410560573

Joshi, A., Bhattacharyya, P., & Carman, M.J. (2016). *Automatic sarcasm detection: A survey*. .10.18653/v1/2020.figlang-1.10

Kaggle. (n.d.). https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasmdetection?select=Sarcasm_Headlines_Dataset.json

Khatri, A. (2020). *Sarcasm Detection in Tweets with BERT and GloVe Embeddings*. ACL.

Kumar, A., Narapareddy, V. T., Srikanth, V. A., Malapati, A., & Neti, L. B. M. (2020). Sarcasm detection using multi-head attention based bidirectional LSTM. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 6388–6397. Advance online publication. doi:10.1109/ACCESS.2019.2963630

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013-Workshop Track Proceedings,* 1–12.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

MisraR.AroraP. (2019). Sarcasm detection using hybrid neural network. https://doi.org/10.13140/RG.2.2.32427.3920

Onyinye, , & Afli, . (2020). *Detecting Sarcasm in News Headlines*. CERC.

Pan, H., Lin, Z., Fu, P., & Wang, W. (2020). *Modeling the Incongruity between Sentence Snippets for Sarcasm Detection*. ECAI.

Patro, J. (2019). *A deep-learning framework to detect sarcasm targets*. ACL.

Pelser, D., & Murrell, H. (n.d.). Deep and Dense Sarcasm Detection. arXiv.org

Porwal, S., Ostwal, G., Phadtare, A., Pandey, M., & Marathe, M.V. (2018). *Sarcasm detection using recurrent neural network*. Academic Press.

Rajadesingan, A. (2015). *Sarcasm Detection on Twitter: A Behavioral Modeling Approach*. ACM.

Son, L., Kumar, A., Sangwan, S., Arora, A., Nayyar, A., & Abdel-Basset, M. (2019). Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 23319–23328. Advance online publication. doi:10.1109/ACCESS.2019.2899260

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5999–6009s.

Tarun Jain is an Assistant Professor.

Horesh Kumar is an Assistant Professor.

Payal Garg is an Assistant Professor.

Abhinav Pillai is a student in CSE.

Aditya Sinha is an Assistant Professor.

Vivek Kumar Verma is an experienced Associate Professor with a demonstrated history of working in the Technical education industry. Skilled in Research, E-Learning, Lecturing, Teaching, and Higher Education. Strong education professional with a Doctor of Philosophy (Ph.D.) focused in Natural Language Processing from Manipal University Jaipur. Working as Associate Professor at the School of Information Technology Manipal University Jaipur.