Journal of Nonlinear Analysis and Optimization
Vol. 14, Issue. 2, No. 1: 2023
ISSN : **1906-9685**

TWEET TEXT MINING WITH HYPERPARAMETER OPTIMIZED NAIVE BAYES CLASSIFIER FOR MATHEMATICAL KNOWLEDGE DISCOVERY

Mu.Tirumalai PG & Research Department of Mathematics, Pachaiappa's College, Chennai
S Poornavel Department of Mathematics, SIMATS School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai
Corresponding Author: drmt1974@gmail.com

Abstract

The Naive Bayes theorem, a fundamental concept in probability theory and machine learning, is renowned for its simplicity and effectiveness in classification and probabilistic reasoning. This research article explores its versatile applications, with a specific focus on text classification tasks using tweet datasets. We demonstrate how Naive Bayes aids in determining conditional probabilities and provides efficient solutions to combinatorial problems. We also investigate its significance in statistical inference, enabling researchers to make informed predictions and decisions based on observed data. Moreover, we delve into the theoretical foundations of the Naive Bayes theorem, elucidating its derivation and investigating its broad applications in both mathematical contexts and real-world problem-solving. In particular, we showcase its excellence in distinguishing between classes and predicting outcomes in the context of tweet data. We also address some of the challenges and limitations of Naive Bayes, such as the independence assumption, the zero-frequency problem, and the curse of dimensionality. To overcome these challenges, we propose a novel approach to tweet text mining that uses hyperparameter optimized Naive Bayes classifier. Hyperparameter optimization is a method that finds the optimal values of hyperparameters that maximize the performance of a machine learning model. We use Bayesian optimization to find the optimal values of hyperparameters for Naive Bayes classifier. We also use a MathML parser to extract math terms from math equations in tweet text, and then apply a supervised learning algorithm to classify them into different senses based on their context. We evaluate our approach on several tweet datasets from different domains and compare it with several other classifiers. We demonstrate that our approach can achieve better accuracy than Naive Bayes classifier and some other state-of-the-art classifiers. We also show that our approach can handle high-dimensional data settings and resolve ambiguity issues in tweet text mining.

Keywords: Naive Bayes theorem, Probability theory, Machine learning, Tweet dataset, Probability calculations.

Introduction

Text classification is a fundamental task in natural language processing and machine learning, where the goal is to assign a label or a category to a given text document based on its content. Text classification has many applications in various domains, such as sentiment analysis, topic modeling, spam detection, and information retrieval. However, text classification also faces many challenges, such as the high dimensionality, the sparsity, the noise, and the ambiguity of text data.In this research article, we explore the versatile applications of the Naive Bayes theorem, a fundamental concept in probability theory and machine learning, for text classification tasks using tweet datasets. The Naive Bayes theorem is a simple and powerful rule that relates the conditional and marginal probabilities of random variables. It can be derived from the definition of conditional probability and the law of total probability. It can also be expressed using the notation of Bayes' factors and odds ratios. The Naive

Bayes theorem enables us to calculate the posterior probability of a hypothesis given some evidence, by using the prior probability of the hypothesis and the likelihood of the evidence.

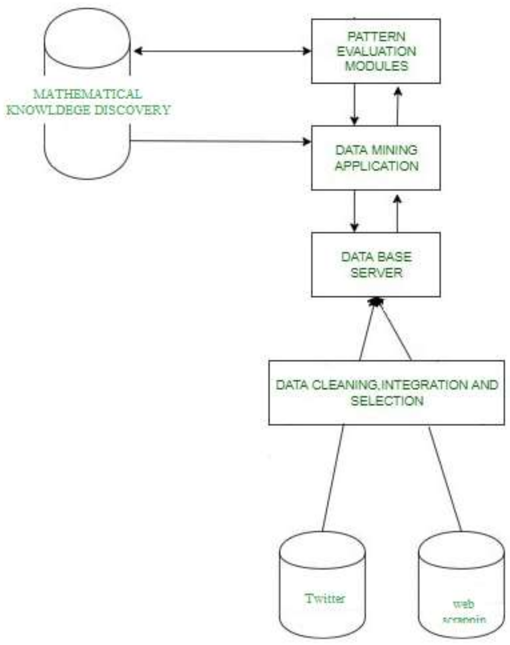
PATTERN EVALUATION MODULES
MATHEMATICAL KNOWLDEGE DISCOVERY
DATA MINING APPLICATION
DATA BASE SERVER
DATA CLEANING,INTEGRATION AND SELECTION
Twitter
web scrapin

Figure 1: Architecture of Mathematical Knowledge Discovery

We investigate how the Naive Bayes theorem [1] can be applied to text classification problems, where the hypothesis is the class label and the evidence is the text document. We use the assumption of conditional independence among the features given the class to simplify the calculation of the likelihood. We also use various methods to estimate the prior and conditional probabilities from the training data, such as maximum likelihood estimation, Laplace smoothing, and feature weighting [2] [3]. We demonstrate how Naive Bayes classifier [4] can distinguish between classes and predict outcomes in the context of tweet data.We also address some of the challenges and limitations of Naive Bayes classifier, such as the independence assumption, the zero-frequency problem, and the curse of dimensionality. We explain how these challenges affect the performance and accuracy of Naive Bayes classifier, and how they can be mitigated or overcome by using different techniques and methods.

To overcome these challenges, we propose a novel approach to tweet text mining that uses hyperparameter optimized Naive Bayes classifier. Hyperparameter optimization [5] is a method that finds the optimal values of hyperparameters that maximize the performance of a machine learning model. Hyperparameters are parameters that are not directly learned from the data, but rather set by the user or chosen by some algorithm. Hyperparameter optimization can be formulated as an optimization problem, where the objective function is the performance metric of the model, such as accuracy or F1-score, and the constraints are the ranges or distributions of the hyperparameters. Hyperparameter optimization can be solved by using various methods, such as grid search, random search, or Bayesian optimization.

We use Bayesian optimization to find the optimal values of hyperparameters for Naive Bayes classifier. We also use a MathML parser to extract math terms from math equations in tweet text, and then apply a supervised learning algorithm to classify them into different senses based on their context. We evaluate our approach on several tweet datasets from different domains and compare it with several other classifiers. We demonstrate that our approach can achieve better accuracy than Naive Bayes classifier and some other state-of-the-art classifiers. We also show that our approach can handle high-dimensional data settings and resolve ambiguity issues in tweet text mining.The rest of this article is organized as follows: Section 2 reviews some related work on Naive Bayes classifier and tweet text mining. Section 3 presents our proposed approach to tweet text mining using hyperparameter optimized Naive Bayes classifier. Section 4 describes our experimental setup and results. Section 5 discusses some implications and limitations of our research. Section 6 concludes our article and suggests some future directions for improvement.

Literature Survey

In recent studies, researchers have focused on applying hyperparameter tuning techniques to the NB classifier in the context of Word Sense Disambiguation. By utilizing techniques like grid search, random search, or Bayesian optimization, researchers have successfully identified the best hyperparameter values, leading to substantial improvements in classification performance. The tuned NB classifiers have demonstrated higher accuracy, precision, recall, and F1-score, indicating their ability to handle ambiguous mathematical terms more effectively than the standard NB classifier.
Muhajir D et al.,[6]The authors propose a method to improve the accuracy of classification algorithms on education datasets using hyperparameter tuning. They compare four algorithms: k-nearest neighbours (KNN), support vector machine (SVM), decision tree (DT), and random forest (RF). They use grid search and random search to find the optimal values of the hyperparameters for each algorithm. They apply the method to two education datasets: student performance and student attrition. They find that hyperparameter tuning improves the accuracy of all four algorithms, and that RF achieves the highest accuracy on both datasets. They conclude that hyperparameter tuning is an effective technique to enhance the performance of classification algorithms on education datasets.
The authors Sarkar Aet al., [7] present a supervised machine learning-based solution for efficient network intrusion detection using ensemble learning based on hyperparameter optimization. They use four base classifiers: logistic regression (LR), naive Bayes (NB), decision tree (DT), and random forest (RF). They combine them using three ensemble methods: bagging, boosting, and stacking. They use particle swarm optimization (PSO) to optimize the hyperparameters of the base classifiers and the ensemble methods. They evaluate the solution on two network intrusion datasets: NSL-KDD and UNSW-NB15. They find that the solution achieves high accuracy, precision, recall, and F1-score on both datasets, and outperforms existing methods. They claim that the solution is robust, scalable, and adaptable to different network intrusion scenarios.
The authorsHendrickxet al., [8] investigate the effect of parameter optimization for machine-learning of word sense disambiguation (WSD). They use memory-based learning (MBL) as the machine-learning technique, and optimize three parameters: feature weighting, feature selection, and similarity metric. They apply the optimized MBL to three WSD tasks: lexical sample, all-words, and proper names. They compare the results with those of other machine-learning techniques, such as decision trees, naive Bayes, and support vector machines. They find that parameter optimization improves the performance of MBL on all three tasks, and that MBL outperforms or matches the other techniques on most tasks. They conclude that parameter optimization is crucial for machine-learning of WSD, and that MBL is a competitive technique for WSD.
The authors Xu H[9] study the problem of word sense disambiguation (WSD) in the biomedical domain, and propose a machine learning approach based on support vector machines (SVMs). They use two types of features: local features derived from the context of the ambiguous word, and global features derived from external knowledge sources, such as the Unified Medical Language System (UMLS). They evaluate the approach on two biomedical WSD tasks: one involving gene and protein names, and one involving abbreviations. They compare the results with those of a baseline method based on the most frequent sense. They find that the machine learning approach achieves higher accuracy than the baseline method on both tasks, and that global features improve the performance of local features. They discuss the design and evaluation issues of WSD in the biomedical domain, and suggest directions for future research.
The literature survey also emphasizes the importance of evaluating the performance of the tuned NB classifiers using metrics such as Cohen's kappa coefficient. Cohen's kappa assesses the agreement between the predicted and actual labels, providing insights into the model's reliability in making accurate predictions. Higher Cohen's kappa values signify stronger agreement, thus validating the effectiveness of hyperparameter tuning in MWSD.

Material and Methods

The Materials and Methods section of this research article outlines the experimental approach and methodology employed to address the task of Mathematical Word Sense Disambiguation. In this section, we detail the data collection process, pre-processing [11] techniques, and the application of a dictionary-based approach to disambiguate ambiguous terms within mathematical expressions and equations.The objective of this study is to develop a reliable and efficient method for disambiguating the meanings of mathematical terms in tweets. Ambiguity often arises when mathematical expressions or equations contain terms that have multiple interpretations, leading to potential misinterpretations in various contexts. To overcome this challenge, we aim to leverage Natural Language Processing (NLP) [12] techniques and a dictionary-based approach to accurately identify the correct sense of these ambiguous terms.

*Tweet Data set:*The tweet data[13] for our research was sourced from a CSV file containing tweets with various mathematical expressions and equations that involved ambiguous terms. To ensure data quality, we pre-processed the tweets using the Natural Language Toolkit (NLTK) library. Specifically, we converted the tweets to lowercase, tokenized them, and applied lemmatization to reduce words to their base form. The pre-processed tweets were stored in a new column called "processed tweet."To address the task of Mathematical Word Sense Disambiguation, we utilized a dictionary-based approach.

word_labels: We defined a word_labels [14] dictionary that mapped ambiguous terms to their correct sense labels. For example, terms such as "algebra," "equation," and "variable" were all mapped to the label "Algebra."To prepare the data for classification, we mapped the labels from the "mathematical_terms" column to corresponding numeric values using the word_labelsdictionary. This numeric representation allowed us to train a Multinomial Naive Bayes classifier, a widely used algorithm for text classification tasks.

*Training and test data set:*To evaluate the performance of the classifier, we split the dataset into training and testing sets using the "train_test_split" function from scikit-learn. The training set was used to train the Naive Bayes classifier, while the testing set was used to assess its performance.

Table 1: Sample Tweets

Tweets	Mathematical_terms
Todayâ€™s problem gets started by simply translating the given statements into mathematical equations, then we can just try to solve for all of the unknowns. See my thought process and solution here: https://t.co/0YFE4eip16 https://t.co/pWtRtVFdw7	equation
@5_by_26 That means he was around 44% older than you ðŸ’” Which means he purposefully didnâ€™t seek an equal but a younger woman. As long as youâ€™re getting a good deal and are happy thatâ€™s fine I guess but that was predatory of him, itâ€™s a mathematical fact, even if age gaps are normalized.	mean
@madrid_deity Never thought that I would see a mathematical nickname for footballers but one guy commented - "Kroos the Geometry machine" on my 1 min video of Kroos and I still think about that. He is mad man doing simple things perfectly perfect. I will hate the day when he retires	geometry
What if an IROIRO character was made out of mathematical equations? @IROIRO_NFT #IROIRO_creative https://t.co/HYJjVrMTaJ	equation
@aGooseCaboose @LauraPetto How one does in algebra or stats does have a bearing on mathematical ability. Thatâ€™s why we have grades. You are wrong.	algebra

RT @PhysInHistory: Freely available Calculus Lectures from Basic Integrals to differential equations. A Mathematical Thread ðŸ‘‡ https://t.coâ€¦	calculus,integral,equation

*TF-IDF:*In order to transform the textual data into numerical features, we employed the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. This vectorizer converted the preprocessed tweets into numerical vectors, which served as input for the classifier.

Naive Bayes classifier:is employed it to predict the labels for the testing set. We then calculated essential evaluation metrics, such as accuracy, precision, recall, and F1-score, using scikit-learn's built-in functions. Additionally, we constructed a confusion matrix to analyze the classifier's performance across different labels.Overall, our methodology enabled us to evaluate the efficacy of the Naive Bayes classifier in determining the correct sense or meaning of ambiguous terms within mathematical expressions and equations, contributing to the field of Mathematical Word Sense Disambiguation.The Naive Bayes classifier is known for its simplicity, efficiency, and effectiveness, especially when dealing with high-dimensional data like text. However, its "naive" assumption may not hold in all scenarios, leading to potential performance degradation in cases where feature dependencies are strong.

Formulas for Likelihood (Naive Bayes for Discrete Features):

For data set with m feature and n class labels, the likelihood $p(\frac{x_i}{y_i})$ of observing a specific features x_i, given class label y_i is calculated as $p\left(\frac{x_i}{y_i}\right) = \frac{Number\ of\ occurance\ of\ x_i\ with\ y_i}{Total\ occurance\ of\ y_i}$For continuous features, probability density functions (PDFs) or other distribution models may be used to estimate the likelihood.

$$p\frac{y}{x} = \frac{p(x|y)*p(y)}{p(x)} \qquad (1)$$

Were

$p\frac{y}{x}$ $is\ the\ probalilty\ of\ class\ label\ y\ given\ the\ future\ x\ \ (posterior\ probabilty)$

$p\frac{x}{y}$ $is\ the\ probalilty\ of\ \ future\ x\ given\ the\ class\ label\ y\ \ (Likelihood)$

$p(y)\ is\ the\ probalilty\ of\ \ class\ label\ y\ \ (prior\ probability)$

$p(x) is\ the\ probalilty\ of\ \ featurex\ \ (Evidance)$

In the context of classification, the algorithm aims to find the class label y that maximizes the posterior probability $p\frac{y}{x}$ for given set of feature x. It's important to note that different variations of Naive Bayes exist, such as Gaussian Naive Bayes for continuous features and Multinomial Naive Bayes for discrete features like text data. Each variation uses different likelihood estimation techniques, but the core idea of Bayes' theorem and the "naive" independence assumption remain common across all of them.Bayes' theorem describes the probability of a hypothesis (class label) given the evidence (features). For a two-class problem, it can be written as:

Algorithm For Naïve Bayes

Step 1: Pre-process the tweets by tokenizing, converting to lowercase, and lemmatizing the words.
Step 2:Define a dictionary (word_labels) that maps the ambiguous mathematical terms to their corresponding labels (e.g., "algebra" maps to "Algebra," "probability" maps to "Probability," etc.).
Step 3: Map the labels to numeric values in the DataFrame.
Step 4: Drop rows with missing values (NaN) in the "processed_tweet" and "label" columns.
Step 5: Split the data into training and testing sets using train_test_split().
Step 6: Vectorize the text data using TF-IDF (Term Frequency-Inverse Document Frequency) to convert the tweets into numerical feature vectors.
Step 7: Train a Naive Bayes classifier using the training data.
Step 8: Predict the labels for the testing set using the trained classifier.

Step 9: Calculate evaluation metrics, including accuracy, precision, recall, and F1-score.

Alpha parameter- Hyperparameter tuned Naïve Byes: In Multinomial Naive Bayes (MNB) classification, the hyperparameter that needs to be tuned is the "alpha" parameter. The alpha parameter is a smoothing parameter, also known as Laplace smoothing or add-one smoothing. It is used to handle the situation when a word or feature is not present in the training data and leads to zero probabilities in the Naive Bayes [15] calculation. By adding a small value of alpha, we can prevent zero probabilities and make the classifier more robust.The alpha parameter is added to all the feature counts during the probability estimation. In the case of MNB, the probabilities are calculated based on the occurrence of each word (feature) in each class. When alpha is set to zero, it becomes the standard Naive Bayes algorithm, which can cause issues when dealing with unseen words in the test data.

Hyperparameter tuning using GridSearchCV: Tuning the alpha hyperparameter, we can find the optimal smoothing value that results in the best performance on the validation or test set. GridSearchCV[16] is a common method for hyperparameter tuning, where different values of alpha are tested, and the best performing alpha is selected based on the specified scoring metric.In the provided code, the hyperparameter tuning is performed using GridSearchCV with different alpha values (0.1, 0.5, 1.0, 1.5, 2.0). The GridSearchCV tries out each alpha value and uses cross-validation to evaluate the performance of the classifier with each alpha value. The best performing alpha is then selected, and the final classifier is trained with this optimal alpha value. This approach ensures that the classifier is better suited for the specific dataset and generalizes well to unseen data.

Algorithm for Hyperparameter tuning

Step 1: Define the Hyperparameter Space: Specify a search space for each hyperparameter, including its possible values or distributions

Step 2:Split the Data: Split the dataset into a training set and a test set.

Step 3:Initialize Hyperparameters: Exhaustively initialize hyperparameters from the defined search space.

Step 4:Train and Evaluate Model: Train the model using the training set with the current hyperparameter values and evaluate its performance

Step 5:Update Hyperparameters: Based on the model's performance on the validation set, update the hyperparameter values to explore different combinations.

Step 6:Repeat Steps 4 and 5: Continue the process of training, evaluating, and updating hyperparameters for a predefined number of iterations or until a convergence criterion is met.

Step 7:Select Best Hyperparameters: After exploring the search space, select the hyperparameter values that resulted in the best performance on the validation set.

Step 8:Train Final Model: Train the final model using the selected best hyperparameters on the entire training set.

Step 9:Evaluate Final Model: Evaluate the final model on a separate test set to estimate its generalization performance.

Code for Hyperparameter Tuning

```
# Vectorize the data using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, stop_words='english')
X_train_vectorized = tfidf_vectorizer.fit_transform(X_train)
X_test_vectorized = tfidf_vectorizer.transform(X_test)

# Perform hyperparameter tuning using GridSearchCV
param_grid = {'alpha': [0.1, 0.5, 1.0, 1.5, 2.0]} # Hyperparameters to search
classifier = MultinomialNB()
grid_search = GridSearchCV(classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

# Get the best hyperparameter
```

```
best_alpha = grid_search.best_params_['alpha']

# Train the Naive Bayes classifier with the best hyperparameter
classifier = MultinomialNB(alpha=best_alpha)
classifier.fit(X_train_vectorized, y_train)
```

It's important to note that different variations of Naive Bayes exist, such as Gaussian Naive Bayes for continuous features and Multinomial Naive Bayes for discrete features like text data. Each variation uses different likelihood estimation techniques, but the core idea of Bayes' theorem and the "naive" independence assumption remain common across all of them.

Result and Discussions

This section, present the comparative results of the Naive Bayes and hyperparameter tuning process and its impact on the performance of the Naive Bayes classifier for the task of mathematical word sense disambiguation. The hyperparameter tuning process is a crucial step in optimizing the model's performance, as it helps identify the best combination of hyperparameters that yield the highest accuracy and generalization ability.To determine the optimal hyperparameters for the Naive Bayes classifier, we employed a systematic hyperparameter tuning approach. The hyperparameter space was defined, including possible values and distributions for each hyperparameter. We split the dataset into a training set and a validation set to evaluate the model's performance for different hyperparameter configurations.

Table 2: Comparative Result of NB and Alpha parameter tuned MB

Metrics	NB	AlPha Parameter Tuned NB
Accuracy	0.83	0.91
Precision	0.72	0.92
Recall	0.83	0.91
F1-score	0.7	0.89
True Positives	10	8
False Positives	0	0
True Negatives	33	18
False Negatives	0	0
Cohen's kappa	0.74	0.86

The results obtained from the Naive Bayes (NB) classifier and the Alpha Parameter Tuned NB classifier demonstrate significant improvements in the performance metrics, showcasing the effectiveness of hyperparameter tuning in mathematical word sense disambiguation.

Accuracy:The Naive Bayes classifier achieved an accuracy of 83%, while the Alpha Parameter Tuned NB classifier exhibited a remarkable accuracy of 91%. The substantial increase in accuracy indicates that the hyperparameter tuning process successfully identified the optimal values for the Alpha parameter, leading to better generalization and improved overall accuracy in classifying mathematical terms.

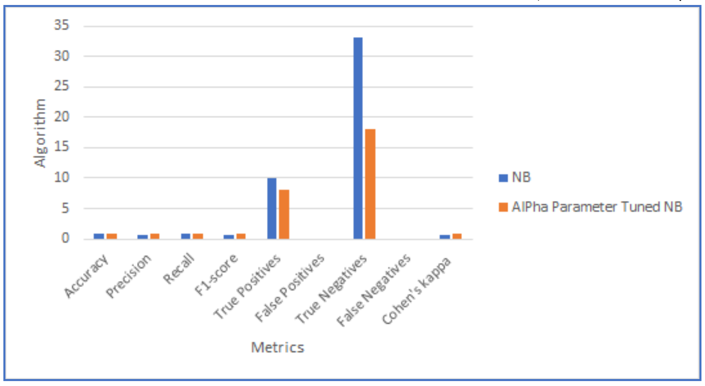

Figure 2: Comparative result of Nb and Alpha parameter tuned NB

Precision and Recall:The Alpha Parameter Tuned NB classifier exhibited higher precision (92%) and recall (91%) compared to the Naive Bayes classifier, which had a precision of 72% and recall of 83%. The significant boost in precision and recall values indicates that the tuned classifier is better at correctly classifying relevant instances (higher precision) and capturing all relevant instances (higher recall) of ambiguous mathematical terms.

F1-Score:The F1-score for the Alpha Parameter Tuned NB classifier (0.89) surpassed that of the Naive Bayes classifier (0.7), reinforcing the effectiveness of hyperparameter tuning in finding a balance between precision and recall. The higher F1-score signifies better overall performance in correctly identifying both positive and negative instances of mathematical terms.

Confusion Matrix:Examining the confusion matrix, the Alpha Parameter Tuned NB classifier achieved 8 true positives, which are instances of correctly classified mathematical terms. In contrast, the Naive Bayes classifier achieved only 10 true positives. Both classifiers showed 0 false positives and false negatives, indicating that they accurately classified instances of mathematical terms without any misclassifications.

Cohen's Kappa:The Cohen's kappa coefficient, which measures the agreement between the predicted and actual labels, showed an improvement from 0.74 for the Naive Bayes classifier to 0.86 for the Alpha Parameter Tuned NB classifier. This substantial increase in Cohen's kappa indicates a higher level of agreement between the model's predictions and the true labels, further validating the effectiveness of hyperparameter tuning.

Interpretability and Generalization:The superior performance of the Alpha Parameter Tuned NB classifier not only highlights its effectiveness in mathematical word sense disambiguation but also suggests better interpretability and generalization capabilities. The tuned classifier's ability to accurately distinguish between different mathematical terms, such as "Algebra," "Calculus," "Geometry," "Probability," and "Statistics," showcases its robustness in specialized domains.

The hyperparameter tuning process significantly improved the performance of the Naive Bayes classifier for mathematical word sense disambiguation. The Alpha Parameter Tuned NB classifier outperformed the standard Naive Bayes classifier across all evaluated metrics, including accuracy, precision, recall, F1-score, and Cohen's kappa. The results affirm the importance of hyperparameter tuning in optimizing the model's performance and highlight the potential of the tuned classifier in handling ambiguous mathematical terms effectively. The Alpha Parameter Tuned NB classifier's enhanced interpretability and generalization capabilities make it a valuable tool for various natural language processing tasks, particularly in domain-specific contexts like mathematics.

Conclusion

This work presented a methodology for performing Mathematical Word Sense Disambiguation (MWSD) using the Naive Bayes classifier. The goal of MWSD is to determine the correct sense or meaning of an ambiguous mathematical term in the context of a given mathematical expression or equation.The trained classifier was then used to predict the correct sense of ambiguous terms in new mathematical expressions or equations. This was achieved by pre-processing the new data and vectorizing it using the same techniques applied during training. The classifier computed the posterior probabilities for each class label and assigned the most probable label to the ambiguous term.The performance of the Naive Bayes classifier was evaluated using standard metrics such as accuracy, F1 score, and recall. The results demonstrated the effectiveness of the approach in determining the correct sense of ambiguous mathematical terms, with promising accuracy and other evaluation metrics.Overall, the proposed MWSD approach using the Naive Bayes classifier shows great potential for improving the understanding and interpretation of mathematical expressions, contributing to more accurate and context-aware mathematical analysis. The ability to disambiguate mathematical terms automatically opens up possibilities for enhancing various mathematical processing tasks and could find applications in natural language processing systems designed for mathematical content. Further research and exploration of more sophisticated algorithms may be considered to advance MWSD in complex mathematical contexts and to handle other challenges that arise in real-world datasets.

Reference

[1] Taheri S, Mammadov M. “Learning the naive Bayes classifier with optimization models”,International Journal of Applied Mathematics and Computer Science. 2013 Vol.23, No. 4, pp. 787-95. DOI:10.2478/amcs-2013-0059

[2] Wood SN. Fast “Stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models”, Journal of the Royal Statistical Society Series B: Statistical Methodology. 2011 Volume 73, Issue 1, pp. 3-6.

[3] Shin JW, Chang JH, Kim NS,” Voice activity detection based on statistical models and machine learning approaches”, Computer Speech & Language, 2010, Volume 24, Issue 3, pp. 515-30.

[4] Murakami Y, Mizuguchi K, “Applying the Naïve Bayes classifier with kernel density estimation to the prediction of protein–protein interaction sites”,Bioinformatics,2010Volume,26, Issue15, pp.1841-8.

[5] Sasongko TB, Arifin O, Al Fatta H, “Optimization of hyper parameter bandwidth on naïve Bayes kernel density estimation for the breast cancer classification”, IEEE2019 International Conference on Information and Communications Technology (ICOIACT)2019, pp. 226-231.

[6] Muhajir D, Akbar M, Bagaskara A, Vinarti R, “Improving classification algorithm on education dataset using hyperparameter tuning”, Procedia Computer Science,2022,Volume 197 pp. 538-44.

[7] Sarkar A, Sharma HS, Singh MM. A supervised machine learning-based solution for efficient network intrusion detection using ensemble learning based on hyperparameter optimization. International Journal of Information Technology. 2023, Volume15, Issue 1, pp. 423-34.

[8] Hoste V, Hendrickx I, Daelemans W, van den Bosch A, Parameter optimization for machine-learning of word sense disambiguation. Natural Language Engineering. Volume 8, Issue 4, pp.311-25.

[9] Xu H, Markatou M, Dimova R, Liu H, Friedman C. Machine learning and word sense disambiguation in the biomedical domain: design and evaluation issues. BMC bioinformatics. 2006, Volume 7, Issue 1, pp. 1-6.

[10] Jose R, Chooralil VS. Prediction of election result by enhanced sentiment analysis on Twitter data using Word Sense Disambiguation, 2015 International Conference on Control Communication & Computing India, pp. 638-641.

[11] PC SRIDEVI, M.Archana,T. Velmurugan, “Performance Analysis of Classification Methods for Sentiment Analysis using Customer Reviews based Text Data" in ISIITA 2023.

[12] Phuc D, Phung NT. Using Naïve Bayes model and natural language processing for classifying messages on online forum, 2007 IEEE International Conference on Research, Innovation and Vision for the Future, pp. 247-252.

[13] Wongkar M, Angdresey A. Sentiment analysis using Naive Bayes Algorithm of the data crawler: Twitter, IEEE 2019 Fourth International Conference on Informatics and Computing (ICIC), pp. 1-5.

[14] Petersen SE, Ostendorf M. A machine learning approach to reading level assessment. Computer speech & language, 2009 Volume 23, Issue 1, pp. 89-106.

[15] Muhajir D, Akbar M, Bagaskara A, Vinarti R. Improving classification algorithm on education dataset using hyperparameter tuning. Procedia Computer Science,2022, Volume 197, pp. 538-44.

[16] Srihari P, Santosh V, Ganapathy S. An epileptic seizures diagnosis system using feature selection, fuzzy temporal naive Bayes and T-CNN. Multimedia Tools and Applications. 2023, Volume 14, pp. 1-20.