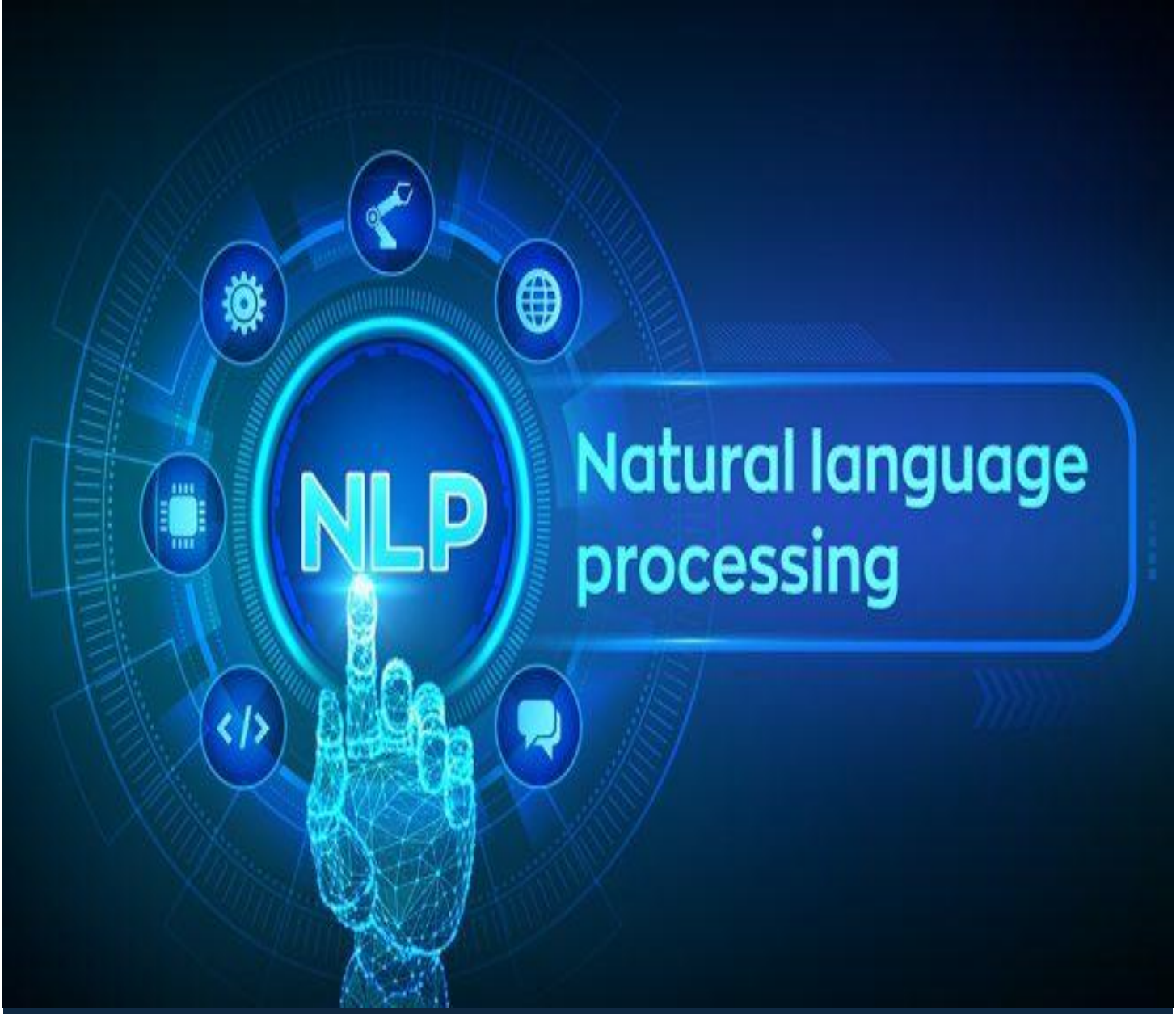

NLP Driven Automated Text Analysis for Topic Classification and Clarity Assessment

MIZANUR RAHMAN

Applied Machine Learning

Table of Contents

Executive summary

Task 1 involved topic classification via a Multilayer Perceptron (MLP) classifier. Hyper-parameters underwent optimization through grid search with 5-fold cross-validation. The results met the client's criteria with minimal misclassification. To evaluate used precision, recall, and F1 score provided a thorough assessment. Overall, the approach effectively fulfilled the task requirements, demonstrating the model's proficiency in accurately classifying paragraphs into predetermined topics.

Task 2 involved creating a classification prototype for text clarity using an MLP classifier. I optimized the hyper-parameters via grid search with 5-fold cross-validation. Results outperformed a trivial baseline, indicating effective model performance without overfitting. A comprehensive evaluation was conducted using confusion matrices, resulting in a thorough assessment. Moreover, the result was compared with a trivial baseline. Ultimately, the prototype fulfils the client's requirements.

Data Exploration and assessment

	par_id	paragraph	has_entity	lexicon_count	difficult_words	last_editor_gender	category	text_clarity
0	428209002237	Ramsay was born in Glasgow on 2 October 1852. ...	ORG_YES_PRODUCT_NO_PERSON_YES_	49	12.0	man	biographies	clear_enough
1	564218010072	It has been widely estimated for at least the ...	ORG_YES_PRODUCT_NO_PERSON_NO_	166	47.0	man	artificial intelligence	not_clear_enough
2	291401001672	He went on to win the Royal Medal of the Royal...	ORG_YES_PRODUCT_NO_PERSON_NO_	69	18.0	non-binary	biographies	clear_enough
3	31548004883	The changes have altered many underlying assum	ORG_NO_PRODUCT_YES_PERSON_NO_	76	27.0	non-binary	programming	clear_enough

Fig-1: Data table

Data Overview: The given dataset is filled with valuable insights ready for exploration. It contains a total of 9347 entries and 8 columns, including 'per_id' for paragraph ID, original paragraph content, 'has_entity' for the presence of entities, 'lexicon_count' for the number of lexicon counts, 'difficult_words' for number of difficult words, last editor gender, 'category' for paragraph classification, and 'text_clarity'.
It's important to keep in mind that this dataset consists of both numerical and categorical data types. 'per_id', 'lexicon_count', and 'difficult_words' are numerical columns, while 'paragraph', 'has_entity', 'last_editor_gender', 'category', and 'text_clarity' are categorical columns. There are some missing values in the 'category', 'difficult_words', and 'text_clarity' columns. Specifically, the 'category' column has 61 missing values, the 'difficult_words' column has 18 missing values, and the 'text_clarity' column has 9338 missing values. Additionally, the 'has_entity' column has 24 entries labelled as 'data missing'.

	par_id	lexicon_count	difficult_words
count	9.347000e+03	9347.000000	9329.000000
mean	3.568369e+11	81.981277	21.514203
std	3.221399e+11	63.533532	16.307358
min	8.500328e+07	0.000000	0.000000
25%	7.019601e+10	33.000000	9.000000
50%	2.684380e+11	64.000000	17.000000
75%	6.124310e+11	117.000000	30.000000
max	1.058779e+12	653.000000	143.000000

Fig-2: Summary statistics of numerical

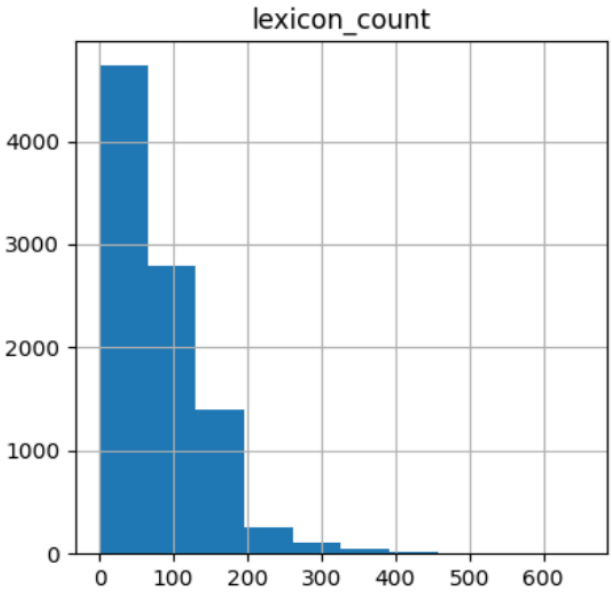

Fig-3: Histogram of lexicon count

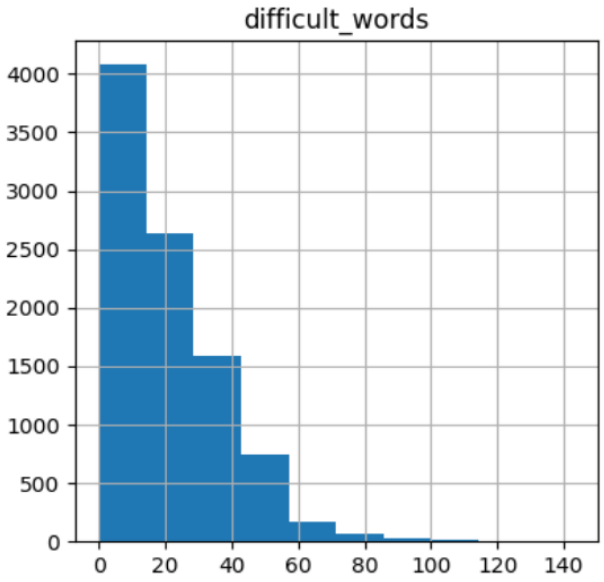

Fig-4: Histogram of difficult words

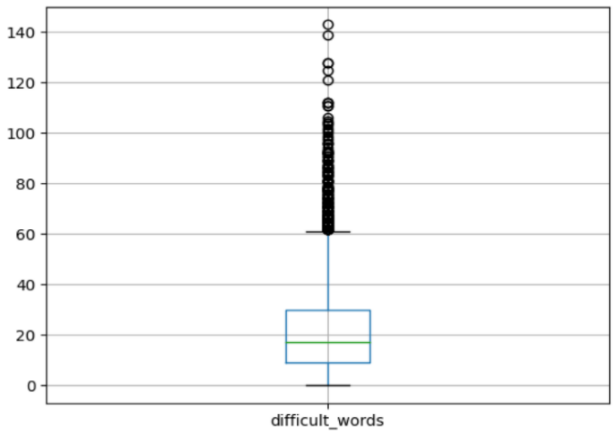

Fig-5: Boxplot of difficult words

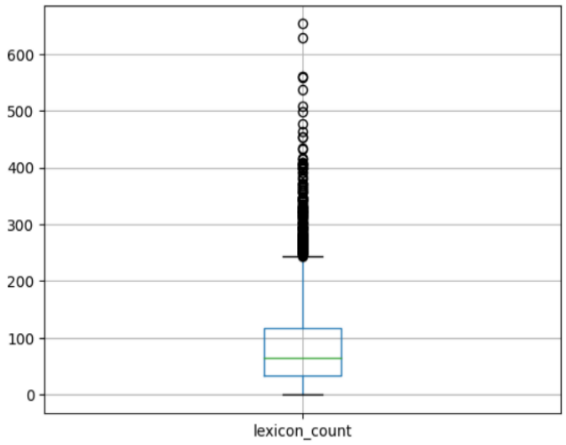

Fig-6: Boxplot of lexicon count

Numerical Features: Based on the summary statistics, it appears that the 'lexicon_count' column has a mean value of 81.981277, while the 'difficult_words' column has a mean value of 21.514203. The histogram plots and boxplot provide a visual representation of the data, showing that the majority of 'lexicon_count' values fall between 30-110, and 'difficult_words' values fall between 10-30. However, there are some outliers in both columns.

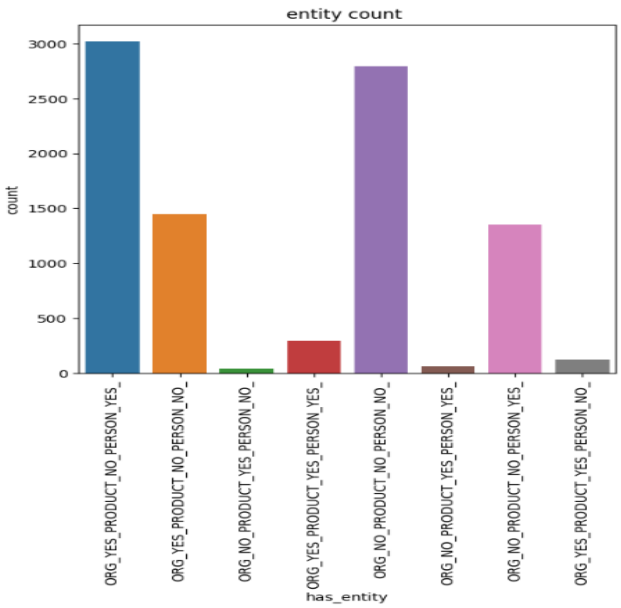

Fig-7: Distribution of 'has_entity' feature

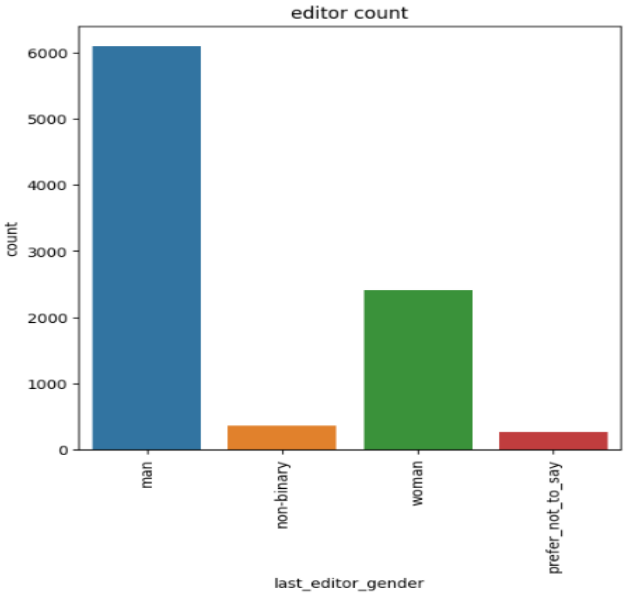

Fig-8: Distribution of 'last_editor_gender' feature

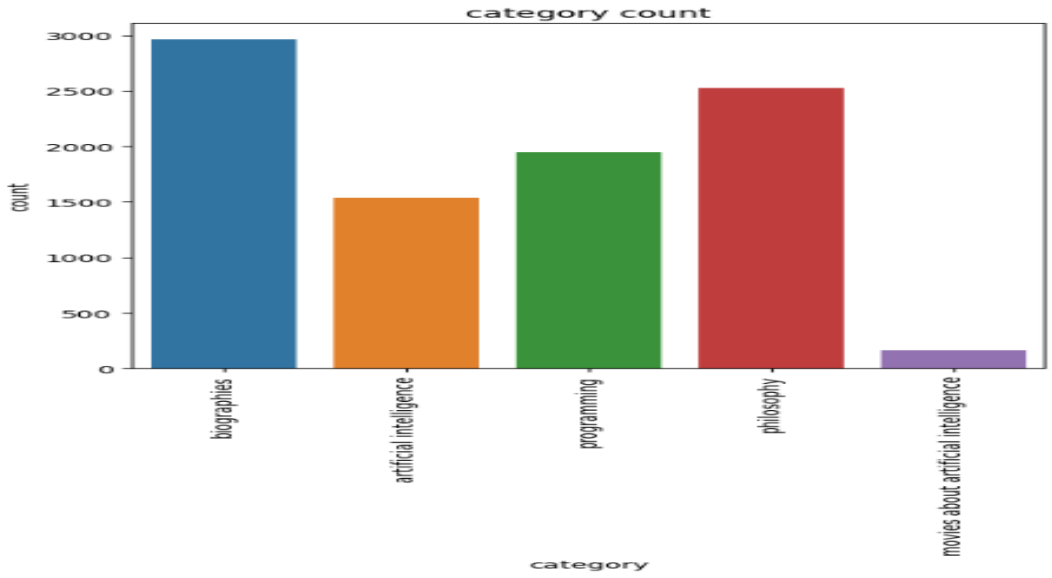

Fig-9: Distribution of 'category' feature

Categorial Features: Categorical features provide information about how data points are classified, grouped, labelled or categorized. For instance, the 'has_entity' feature categorizes the 'paragraph' column into 5 distinct categories. Based on the bar charts, it's evident that all three categorical columns contain imbalanced data. To achieve accurate clustering or

classification results, it's crucial to analyse and interpret the distribution of categorical features within categorical features.

Duplicates and Inconsistencies: Additionally, the dataset contains 215 duplicate entries which need to be removed. Furthermore, there are inconsistencies in the category names due to variations in capitalization, such as 'Biography' and 'biography'. Preprocessing of data can be made easier by addressing missing values and duplicates.

Data splitting and cleaning

Data cleaning: During the data analysis step, I found that there were some missing values in the 'category', 'has_entity', and 'difficult_words' columns. To work with the data, I had to either remove these missing values or fill them. However, removing these values would not be a good choice as it would lead to an incomplete dataset. Therefore, I decided to fill these missing values using an imputation technique. This way, the dataset would remain complete and usable while maintaining its integrity and statistical properties. One of the techniques I used for this is the Simple Imputer.

To fill the numerical column 'difficult_words', I used the mean value. For the categorical columns 'category' and 'has_entity', I used the most frequent value. This approach helps to preserve the distribution of categories, which is important to ensure the data remains representative.

Furthermore, I noticed that some attributes in the 'category' column started with a capital letter. To standardize the column, I transformed all the category names into lowercase.

In order to effectively work with text data, it is essential to perform cleaning and preprocessing. For this I remove special characters, punctuation, URLs, emails, and phone numbers from the text, as well as converting it to lowercase for consistency and to avoid redundant categories.

Cleaning text data has many benefits, including making it easier to analyse using natural language processing techniques and improving the quality of text-based features.

Data splitting: To ensure a fair and accurate model, I divided the dataset into two segments: the training and test sets. By using the 'train_test_split' function from Scikit-learn, I allocated 90% of the data to the training set and the remaining 10% to the test set. To maintain the class distribution across subsets, I stratified the division based on the target variable. This approach effectively eliminated any potential biases during model training and evaluation.

Data encoding

As the dataset has some categorical features. The categorical features ‘category’ and ‘has_entity’ are encoded by the LabelEncoder from the scikit-learn library. This library assigns a unique numerical value to each category.

Moreover, dataset has a text column which containing paragraph. I apply word embeddings vector. It is a Spacy library and representation words or text that capture semantic meanings and relationships between words.

In this step I first break down the text into its constituent parts by tokenization, then lemmatize these tokens to analyse the token as a single item, and finally calculate the average embedding of the tokens to fit them for modelling.

Topic classification

Model building

The model I use is a Multilayer Perceptron (MLP) classifier with a specific hyper-parameter.

The reason for the choice of this MLP classifier for topic classification is its adeptness at handling non-linear relationships and achieving commendable results on a variety of tasks. In natural language processing (NLP) tasks, language is complex and exhibits non-linear patterns and interdependencies. MLPs can be able to determine these complex relationships. It can learn meaningful representations of sentences or words. In addition, it gives the flexibility in case of structuring neural networks. Anyone can select the number of hidden layers, number of neurons, and activation functions. MLPs are the best choice for large-scale NLP tasks, as it can handle large amounts of data. (Windeatt, 2006)

The hyper-parameters I use in the model are hidden layer sizes and learning rate init.

hidden_layer_sizes	This hyper-parameter determines the architecture of the MLP classifier by selecting the number of neurons in each hidden layer.
learning_rate_init	This hyper-parameter determines the initial learning rate for the classifier.

Table1: Model hyper-parameter

To optimize the model, I employed the grid search approach using 5-fold cross-validation, where the dataset is divided into five subsets for training and evaluation. The optimal hyper-parameter combination is determined by selecting the highest accuracy score. For this GridSearchCV class from scikit-learn is used.

Let's compare the results on the validation folds and training from the grid search.

Mean test score	Std test score	Mean train score	Std train score	Hidden layer sizes	Learning rate init
0.864457	0.008083	0.894352	0.002551	(20, 20)	0.0001
0.863229	0.008157	0.964886	0.010004	(20, 20)	0.001
0.868017	**0.005824**	**0.899171**	**0.002064**	**(30,)**	**0.0001**
0.863352	0.003168	0.965500	0.005736	(30,)	0.001

Table2: Results of hyper-parameter

Here the results of training dataset and validation dataset are almost same. So, there is no over/under-fitting problem.

GridSearchCV is able to perform an exhaustive search over all possible combinations of hyper-parameter values. It methodically explores the hyper-parameter space to ensure that

no combination is unexamined, facilitating a comprehensive assessment of various hyper-parameter configurations. It aids mitigate bias during hyper-parameter selection systemically evaluating numerous combinations, thereby lowering the risk of choosing suboptimal hyper-parameters based on a single evaluation. It ensures a more objective and data-driven methodology for hyperparameter tuning. (B. H Shekar, 2019)

Model evaluation

As the target value is categorical using a confusion matrix is the best method for evaluating the model performance. Here I use multiple metrics (precision, recall, F1 score). There is a scikit-learn function named classification_report which gives all the metrics in one place.

	precision	recall	f1-score	support
artificial intelligence	0.87	0.82	0.84	153
biographies	0.90	0.91	0.91	293
movies about artificial intelligence	0.94	0.94	0.94	16
philosophy	0.88	0.88	0.88	249
programming	0.90	0.92	0.91	194
accuracy			0.89	905
macro avg	0.90	0.89	0.90	905
weighted avg	0.89	0.89	0.89	905

Fig-10: Classification report

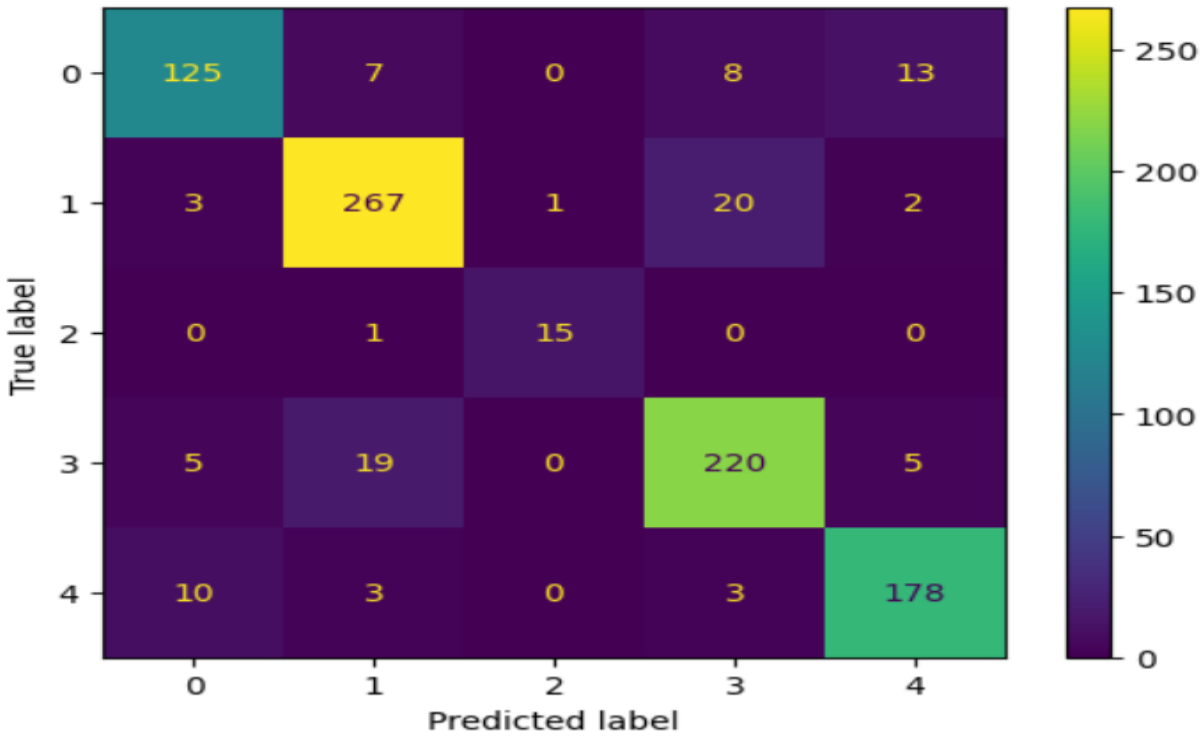

Fig-11: Heat map of confusion matrix

As the target levels are categorical and imbalanced I have to consider precision, recall, or F1 score results to check the model performance. We know that F1 score is calculated from the result of precision and recall. From fig-10 it is clear that all five categories give good results and macro average of F1 score is 90%. So, the model gives only 10% percent mismatch.

Comparison with 'trivial' baseline:

I can find how much accuracy I get calculating model random guess results. For this, I used the scikit-learn function DummyClassifier and it has an argument called 'strategy' which defines what trivial baseline we want to use.

	precision	recall	f1-score	support
artificial intelligence	0.13	0.14	0.13	153
biographies	0.31	0.20	0.25	293
movies about artificial intelligence	0.04	0.44	0.07	16
philosophy	0.28	0.19	0.23	249
programming	0.24	0.24	0.24	194
accuracy			0.20	905
macro avg	0.20	0.24	0.18	905
weighted avg	0.25	0.20	0.22	905

Fig-12: Classification report (for random guess)

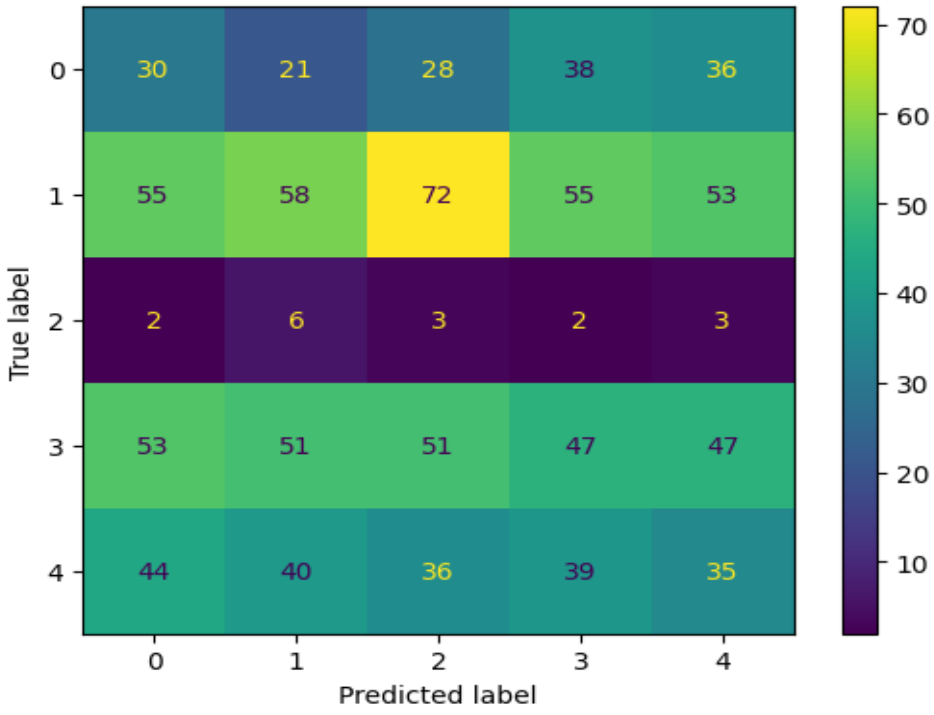

Fig-13: Heat map of confusion matrix (for random guess)

The random guess gives the macro average F1 score is only 18% which is much less than the actual result. So, the model is performing very well.

To address the client's requirements, we can use the results of the metrics and classification report. Precision is important as it ensures the predicted instances for every category are relevant. Recall is significant because it helps minimize the occurrence of missing relevant instances. The F1 score guarantees that the model's performance is not only accurate but also inclusive across all categories.

Conclusion

It is clear that the model is better than a trivial baseline (from section 4b) and it does not overfit the training dataset. Moreover, out of five classes, paragraphs are misclassified less than 10% in three classes ('biographies, 'movies about artificial intelligence' and 'programming'). In the case of the other two classes ('artificial intelligence' and 'philosophy'), paragraphs are misclassified by just over 10%. So, I can say that the model is successful according to the client's definition of success.

Although there are many metrics available to evaluate model's performance, but the F1 score is the most suitable for Clint to track the algorithm's performance. This score provides a balanced measure of the recall and precision, which are important for evaluating the classification performance of a model in imbalanced datasets.

Text clarity classification prototype

Ethical discussion

The utilization of an algorithm to automatically decline user submission depending on predicted text clarity introduces numerous ethical concerns and potential risks that need to be carefully considered. For instance:

Bias: The algorithm may unintentionally incorporate bias based on factors such as language, cultural references, or writing styles.

Data issue: If the data used is not representative or includes biased samples, the algorithm's predictions might not precisely mirror text clarity across all users. Such circumstances can lead to unfair rejections or approvals grounded on erroneous presumptions.

Performance Limitations: Automated algorithms for predicting text clarity may not always deliver accurate assessments. Depending entirely on the prediction of an algorithm may lead to false positives or negatives, leading to unfair acceptance or rejection of user contributions.

To reduce ethical risks associated with utilizing this algorithm, some steps can be considered. Firstly, make sure that the training dataset utilized in creating the algorithm is varied and accurately reflects the user base. Secondly, implement strategies to address fairness and mitigate bias during the algorithm's development and deployment phases. Additionally, it is important to provide users clear guidelines and feedback on enhancing text clarity.

Data labelling

In the provided dataset, there are only nine labelled data points in the 'text_clarity' feature. Out of these nine, four paragraphs are labelled as not clear enough. Interestingly, three of these paragraphs have more than 790 words, while only one has less than 300 words. On the other hand, most of the paragraphs labelled as clear enough have a text length of less than 600 words, with only one paragraph having more than 600 words. Based on these observations, I labelled the 'text_clarity' column according to the text length of the 'paragraph' column. If a paragraph has more than 800 words, it is labelled as not clear enough, whereas if the number of words is less than or equal to 800 words, it is labelled as clear enough. Total 9132 data points are labelled where 7241 data are clear enough and 1891 data are not clear enough. However, two paragraphs are mislabelled due to text length and do not meet the conditions, making it an inadequate way to classify paragraph clarity. So, this labelling system can impact on the final label due to the lack of clues for classify the paragraph's text clarity.

Model building and evaluation

First, I made a prototype with 100 data points. The model I use is a Multilayer Perceptron (MLP) classifier with a specific hyper-parameter. The rationale behind opting for this MLP classifier for topic classification lies in its capability to effectively manage non-linear correlations, yielding impressive outcomes across various tasks. In the realm of natural language processing (NLP), where language intricacies manifest in non-linear fashions and intricate interconnections, MLPs excel in discerning these complexities. They possess the capacity to grasp nuanced relationships within sentences or words, thereby acquiring meaningful representations. Moreover, MLPs offer flexibility in structuring neural networks, allowing users to customize parameters such as hidden layers, neurons, and activation functions. Given their adeptness at handling substantial datasets, MLPs emerge as the optimal choice for large-scale NLP undertakings. (Windeatt, 2006)

The hyper-parameters I use in the model are hidden layer sizes and learning rate
init.hidden_layer_sizes: This particular hyper-parameter plays a pivotal role in shaping the architecture of the MLP classifier, as it dictates the quantity of neurons within each hidden layer.

learning_rate_init: This hyper-parameter dictates the initial learning rate utilized by the classifier, which essentially denotes the magnitude of each step taken during the training iteration.

In refining the model, I utilized the grid search method coupled with 5-fold cross-validation. This technique involves partitioning the dataset into five distinct subsets for both training and evaluation purposes. The most effective hyper-parameter combination is identified by pinpointing the highest accuracy score. To facilitate this process, I employed the GridSearchCV class from the scikit-learn library.

GridSearchCV conducts a thorough exploration of all potential combinations of hyper-parameter values. By systematically traversing this space, it enables a comprehensive evaluation of diverse hyper-parameter configurations. This approach helps mitigate bias in hyper-parameter selection by meticulously assessing numerous combinations. It ensures a more objective and data-driven approach to hyperparameter tuning. The model undergoes iterative training and evaluation five times, each time with a distinct combination of training and validation sets. (B. H Shekar, 2019)

Let's compare the results on the validation folds and training from the grid search.

Mean test score	Std test score	Mean train score	Std train score	Hidden layer sizes	Learning rate init
0.9250	0.061237	0.959375	0.021195	(20, 20)	0.0001
0.9375	0.055902	1.000000	0.000000	(20, 20)	0.001
0.9250	0.072887	0.962500	0.015934	(30,)	0.0001
0.9500	0.046771	1.000000	0.000000	(30,)	0.001

Table-3: Results of hyper-parameter

Here the results of training dataset and validation dataset are almost same. So there is no over/under-fitting problem.

Model Evaluation:

	precision	recall	f1-score	support
clear_enough	0.94	0.94	0.94	16
not_clear_enough	0.75	0.75	0.75	4
accuracy			0.90	20
macro avg	0.84	0.84	0.84	20
weighted avg	0.90	0.90	0.90	20

Fig-14: Classification report of prototype

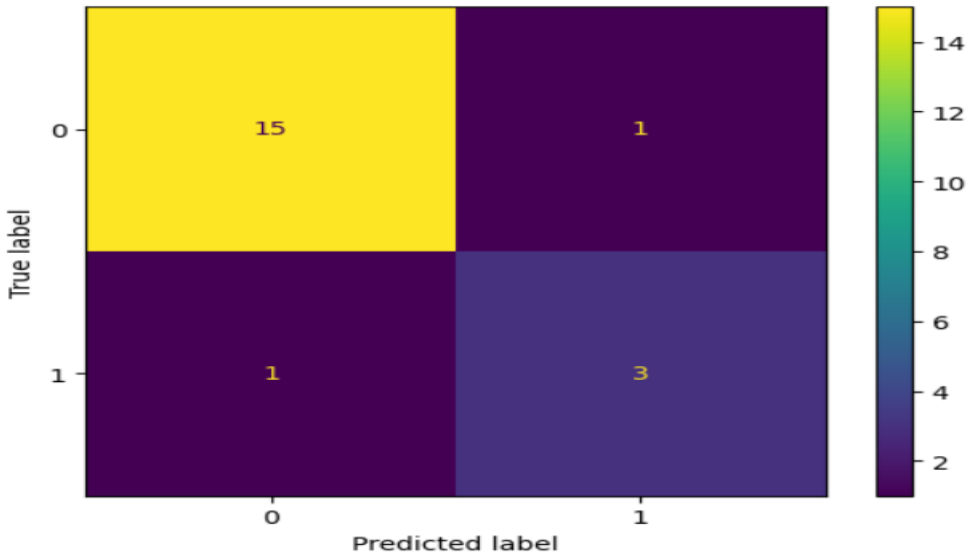

Fig-15: Confusion matrix (for prototype)

Comparison with ‘trivial’ baseline (for prototype):

	precision	recall	f1-score	support
clear_enough	0.75	0.38	0.50	16
not_clear_enough	0.17	0.50	0.25	4
accuracy			0.40	20
macro avg	0.46	0.44	0.38	20
weighted avg	0.63	0.40	0.45	20

Fig-16: Classification report of prototype (for random guess)

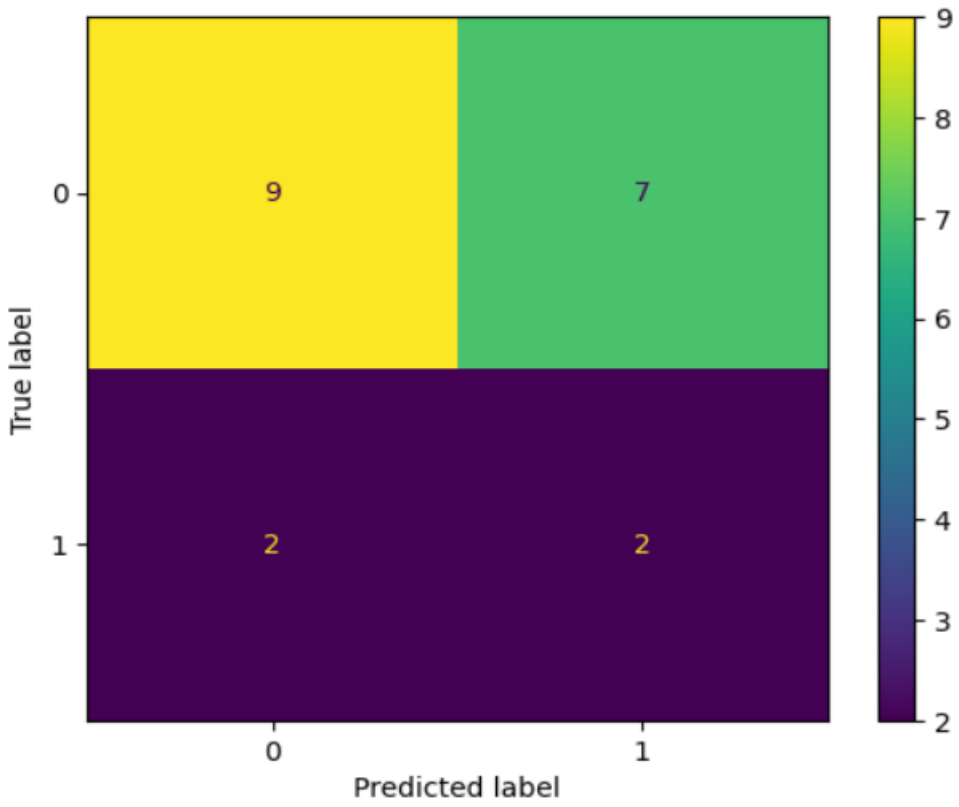

Fig-17: Confusion matrix of prototype (for random guess)

The random guess results are much less than the actual result. So, the model is performing very well. Then I labeled all the paragraph text clarity and checked the results.

Conclusions

It is clear that the model is better than a trivial baseline and it does not overfit the training dataset. So, I can say that the model is successful according to the client's definition of success.

I would recommend the client to consider using the F1-score as an additional scalar performance metric to track the algorithm's performance.

To improve text clarity, clients should provide more labeled data.

References

Terry Windeatt (2006). *Accuracy/Diversity and Ensemble MLP Classifier Design. IEEE, 17(5)*

Available at: https://ieeexplore.ieee.org/abstract/document/1687930/authors#authors

(Accessed 24 March 2024)

B. H Shekar, Guesh Dagnew (2019). *Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data. IEEE*

Available at: https://ieeexplore.ieee.org/abstract/document/8882943/citations?tabFilter=papers#citations

(Accessed 24 March 2024)