Comp1804 Report

19.03.2024

Word Count: 2390

Executive Summary

I have solved the whole of both tasks. For task 1 I have used the Support Vector Machine algorithm, and the results met the client's definition of success as the model is better than a trivial baseline, it doesn't overfit to the training dataset and for each class, no more than 10% of paragraphs get misclassified into an unrelated class. For task 2 I have used the Long Short-Term memory (LSTM) algorithm. The results don't meet the client's definition of success as the model does perform better than the majority class baseline but it overfits on the training data.

1. Data Exploration and Assessment

After loading the dataset, we start with exploring that data to find out any issues as early as possible to prevent problems in the later stages of modelling. The table below describes the EDA (exploratory data analysis) performed:

EDA steps performed	Findings	Action taken / Justification
Finding and dropping duplicate entries	We find 210 duplicate data points.	We remove duplicate data points to prevent identical copies of data points from appearing in both training and test datasets.
Inspecting the dataset	We understand the dataset's basic structure, including dimensions & column names, noting a combination of numerical & non-numerical columns.	This improves data comprehension which helps in selection & determining necessary pre-processing steps.
	The "has_enitty" structure appears complex and implies a need for simplification.	This implies dividing the column into three numerical columns, each for organization, person, and product.
Statistical insight of numerical features	We observe statistical descriptions of the numerical features in the dataset.	NA
Analysing categorical features	"has_entity" displays 24 data points labelled "data missing".	These data points should be replaced with null values in the cleaning stage.

	"category", the target label shows an imbalance.	This suggests a stratified split to maintain the class distribution on both the training and test datasets as in the original dataset.
	We also find mismatched values in "category" in due to the difference in text case.	During the data cleaning stage, text case differences should be fixed for all classes.
	We find imbalance in "category" classes.	There is a bit of imbalance which seems alright to be worked with.
Finding missing values	We find many missing values (9057) in "text_clarity".	These can be ignored at this moment as it is part of task 2.
	We find 18 and 61 missing values in "difficult_words" & "category" respectively.	These data points should be dropped during the cleaning stage to avoid hindering the accuracy of our models.

Table 1:Exploratory Data Analysis

Main characteristics of the dataset:

- Textual data which is "paragraph" column represents the content to be added to the website.
- Categorial target variables which are "category" for task 1 and "text_clarity" for task 2.
- "has_entity" binary feature indicates the presence of entities like organisation, person, or product.
- Numerical features "lexi_count" and "difficult_words" represent lexical characteristics.
- Categorical feature "last_editor_gender" represents that gender of the latest editor.

2. Data Splitting and Cleaning

I have decided to first handle the missing values as splitting the dataset before doing so resulted in an error. First, I changed labels "data missing" in "has_entity" and gave them a null value.

Upon verification, I found 4 features that contain missing values. Out of these, I have dropped rows with missing values in "has_entity" and "category" as they are crucial variables for task 1. There are relatively few missing values compared to the total number of rows and their absence wouldn't affect the task in any significant way, thus, dropping them seems like a reasonable approach to me. This retains the integrity of the categorial information. I have imputed missing values in "difficult words" using the mean strategy. I find this appropriate as it provides a reasonable estimate without introducing bias. As for "text_clarity", I have ignored the column as it will be looked at in task 2.

For inconsistencies in "category" due to the labels' text case, I have converted all labels to lowercase to aid smooth mapping of old labels to new labels by defining a dictionary. After updating the "category" column, the updated distribution of classes is free from inconsistencies.

I have first split the input features ("paragraph"," has_entity") and the target variable ("category") for task 1 into 70/15/15 % for the train/validation/test datasets. I have used stratified splitting to ensure class distribution in the target variable is maintained across the 3 sets.

For task 2, I have created a subset of 80 datapoints including columns "paragraph", "lexicon_count", "difficult_words" (input features) and "text_clarity" (target variable). I have split this subset into 70/15/15 % for the train/validation/test datasets and used stratified split here as well.

3. Data Encoding

I have created a function for pre-processing of text in "paragraph" to convert the text to lower case, removed punctuation, special characters, tokenized the text into individual tokens, removed stop words and converted the tokenized text into numerical representations using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. I have created another function to apply one-hot encoding on "has_entity" to create one-hot encoded values for each entity type.

Following this, I have applied pre-processing, TD-IDF vectorization on the "paragraph" columns and one-hot encoding on the "has_entity" columns. Then I have dropped the original columns from all datasets of task 1.

For task 2, I have tokenized the test of "paragraph using a tokenizer form Keras and converted them into a sequence of integers. The sequences were then padded to ensure uniform length. I have also encoded the target variable.

4. Task 1: Topic Classification

4.A. Model Building

I have employed the Support Vector Machine model as my final solution to this task. I experimented with two other models: Logistic Regression and Random Forest. The general path followed for all models is as followed:

- I have scaled the TF-IDF transformed features and combined them with one-hot encoded features.
- I have defined hyperparameters to tune and adjusted ranges along the way.

- I have used GridSearchCV to find the best hyper-parameters for optimized performance in all three models.
- I have trained the model on my training set.
- Finally, I have performed predictions on the training, validation, and test sets.

I have adjusted different parameters and reduced cross-validation folds, but it did not reduce the runtime for the Logistic Regression model. This did not align with the limited time provided for this submission, hence, I decided to drop this model altogether.

Moving on to the other 2 models, I have experimented with changing the ranges of the parameters, following which I have found higher accuracy (88 %) by employing the SVM model. The random forest model was way more overfitting compared to the SVM model. The hyper-parameters of my final (SVM) model are mentioned in Table 2 below:

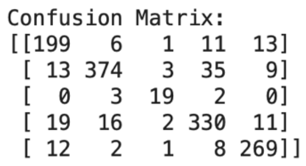
Hyperparameter	Value	Explanation
C (regularization parameter)	0.1	A smaller C value was chosen to increase the regularization strength, to prevent overfitting & improving generalization performance.
kernel	sigmoid	This kernel is fit for handlining non-linear relationships between the features
class weight	balanced	This class weight was crucial for handling the imbalanced dataset & to prevent bias towards the majority class

Table 2: Hyperparameters

4.B. Model Evaluation

Metrics	Results
Trivial Baseline: Based on the class distribution of my target variable “biographies” with 2889 instances of 9047 total instances being the majority class, makes the trivial baseline accuracy approximately 31.9 %.	My final model is better than the trivial baseline (accuracy of 31.9 %) with an overall accuracy of approximately 88 %.

Accuracy: Training accuracy: 89.5 % Validation Accuracy: 86.8 % Test Accuracy: 87.8 %	We find minor differences between the accuracy of training, validation, and test sets but the model's performance is reasonably consistent across all sets. The training accuracy is slightly higher but still similar to the validation and text accuracy. This implies that the model did not overfit the training dataset and it generalizes well to unseen data

Confusion Matrix:

`Confusion Matrix:`
`[[199   6   1  11  13]`
` [ 13 374   3  35   9]`
` [  0   3  19   2   0]`
` [ 19  16   2 330  11]`
` [ 12   2   1   8 269]]`
Figure 1: Confusion matrix
 	
Using the values of the confusion matrix, we calculate the misclassification rate of each class with the following results:
- artificial intelligence: 4.22 %
- biographies: 5.82%
- philosophy: 8.47 %
- programming: 3.09 %
- movies about artificial intelligence: 2.95 %
	Misclassification rate for each class is well under the 10% threshold of the client's requirement.

4.C. Task 1 Conclusions

- This model is successful as it is better than the trivial baseline (majority class), it does not overfit to the training dataset and for each class, no more than 10% of the paragraphs get misclassified into an unrelated class (all 3 requirements of the client).
- I would recommend using precision as a scalar metric to keep track of the model's performance as it is a crucial metric where datasets are imbalanced much like the one in consideration.

5. Task 2: Text Clarity Classification Prototype

5.A. Ethical Discussion

The use of algorithms to automatically reject user's work raises many concerns regarding ethical implications and risks. While machine learning aids ease of work in such cases, it comes with an equal possibility of potential bias, concerns of fairness & equity, questioning of transparency, amplification of bias and much more.

If the algorithm is trained on data that includes biased samples i.e., unfair proportions of samples from certain perspectives, it may reject inputs from such demographic groups. If the features used for training includes bias, even though the writing style and vocabulary used may be very clear but considered culturally or socially biased, then it may prove disadvantageous for their writers. If these rejected samples are not reviewed properly to make corrections, the algorithm may learn its own biases over time and therefore, keep rejecting inputs from the corresponding demographic groups. This may lead to users feeling discouraged from participation in the future. Reasons for automatic rejections not being made transparent may lead to frustration.

To lessen these risks, the organisation can consider the following approaches:

- The algorithm can be used to mark potentially unclear inputs instead to being rejected automatically and later evaluated by humans to consider the potential bias before making a decision.
- Ensuring that the training data and features which will be used by the algorithm are diverse can help to make sure fair treatment is given to all users.
- Regular monitoring of the algorithm to evaluate the results made for different user groups can help reduce the biases in the decision making process.

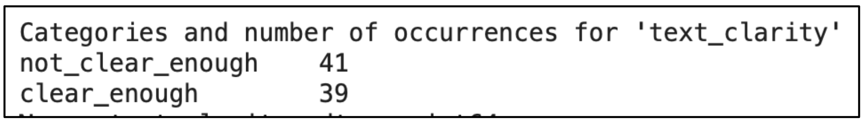
5.B. Data Labelling

I have considered a few aspects of the paragraph to build a criterion for labelling. I have manually read the paragraphs to judge coherence, familiarity and logical flow of the text. Presence of information in a structures and coherent manner, with tendency of being familiar concepts that are easier for readers to understand and a logical flow of ideas, make the text clear enough.

Apart from this I have looked at lexicon count and the number of difficult words, labelling paragraphs with a lower count as clear enough and those with a higher count, no clear enough.

I am fairly certain about my final labels (because I have used sensible measures to build my criterion) except for a few labels where paragraphs showed difficulty in understanding in spite of having a lower lexicon count & lower number of difficult words and visa-versa.

A total of 80 data points were labelled, with a fair balance between both categories as shown in



```
Categories and number of occurrences for 'text_clarity'
not_clear_enough    41
clear_enough        39
```

Figure 2: Occurrences of both categories in "text_clarity"

Example of “clear enough”:
Paragraph: *Herschel is reported to have cast, ground, and polished more than four hundred mirrors for telescopes, varying in size from 6 to 48 inches in diameter. Herschel and his assistants built and sold at least sixty complete telescopes of various sizes.*
Lexicon Count: 41
Difficult Words: 10

Example of “not clear enough”:
Paragraph: *Common Lisp , as described by Common Lisp the Language ‚Äì a consolidation of several divergent attempts to create successor dialects to Maclisp, with substantive influences from the Scheme dialect as well. This version of Common Lisp was available for wide-ranging platforms and was accepted by many as a de facto standard until the publication of ANSI Common Lisp . Among the most widespread sub-dialects of Common Lisp are Steel Bank Common Lisp , CMU Common Lisp , Clozure OpenMCL , GNU CLisp, and later versions of Franz Lisp; all of them adhere to the later ANSI CL standard .*
Lexicon Count: 93
Difficult Words: 25

5.C. Model Building and Evaluation

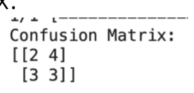
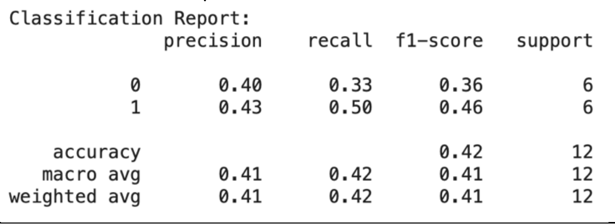
I have created a neural network architecture with the main input being the padded sequences and then converted the integer-coded words into embedding vectors. they were then fed into an LSTM (Long Short-Term Memory) layer. I have also used the numerical features using separate input layers.

The LSTM output was then concatenated with the numerical features. Finally, a dense output layer with sigmoid activation function is used. After this, the model was compiled using binary cross-entropy loss and the Adam optimizer was used to train the model. The hyperparameters used in the model are:

Hyperparameter	Value
Vocabulary size	Determined from data
Embedding dimension	100
LSTM units	128
Learning rate	0.001
Batch size	32
Number of epochs	10
Maximum sequence length	100

I chose this algorithm as its good choice of sequential & contextual understanding, is flexible in regard to model complexity and it has the capability to capture long-term dependencies.

Evaluation of this model is as follows:

Metrics	Results
Confusion matrix:  Confusion Matrix: [[2 4] [3 3]] Classification report:  Classification Report: precision recall f1-score support 0 0.40 0.33 0.36 6 1 0.43 0.50 0.46 6 accuracy 0.42 12 macro avg 0.41 0.42 0.41 12 weighted avg 0.41 0.42 0.41 12	After observing the confusion matrix and the classification report and accuracy results, I find that my model's test accuracy is around 67% which is higher than the majority class baseline (51.25 %)
Accuracy: Training accuracy: 78.5% Validation accuracy: 57.8% Test accuracy: 67.6%	Based on the accuracy percentages, I conclude that the model overfits to the training data.

5.D. Task 2 Conclusions

- I find my model unsuccessful according to the client's definition of success as it does perform better than the majority class baseline but it overfits the training data(the client's requirements for successful model).
- I would suggest using the F1 score to track the model's performance.
- Implementation of regularization techniques can prevent overfitting. Extensive experimentation with hyperparameters can help find optimal combination of hyperparameters.

6. Self-reflection

I could have improved in task 2 as my model overfits the training data. If I was provided with more time to complete the task, I would have taken up a detailed study of hyperparameter tuning to .understand them in detail which would help to improve the results.