Nazarbayev University Email Prioritizer

Madi Turgunov
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan
madi.turgunov@nu.edu.kz

***Abstract*—This paper investigates the use of RoBERTa, a powerful language model, for classifying university student email. The task focuses on addressing inbox overload caused by a mix of necessary and irrelevant emails, a problem inadequately solved by traditional spam filters. Data for the study came from my own personal university inbox (@nu.edu.kz) and was initially labeled using Google's Gemini API for efficiency. Human review ensured label accuracy. The dataset was imbalanced, with twelve defined categories. RoBERTa was trained on this data, using weighted classes to mitigate bias. The final model achieved an F1-score of 0.754 on an unbalanced test set and demonstrated practical utility in a human evaluation (42/50 accuracy). This work underscores RoBERTa's potential for specialized email organization while highlighting the need for larger datasets and further refinement of labeling to improve performance in underrepresented categories.**

I. Task Description

The task is relatively simple: a big problem of our university students is that the email inbox that we so desperately need for studying often gets clogged up by spam and overall useless emails, coming from the University itself! Normal spam detectors do not do this justice, since most of the email are not necessarily *spam* , but they might as well be, as they are useless. This is where RoBERTa comes in: by analyzing the sender, title and contents of the email, the classifier should be able to tell apart between 12 different categories: *Academic, Events, Updates, Club-Related Activity, High-Priority, Work-Related, Spam, Dormitory-Related, Out-of-university, Social and Non-Academic (other).*

II. Data Used

The data used for this task is the inbox from my personal @nu.edu.kz email address. It was received using *Google Takeout*.

The data consists of **5985** samples, which include the title, sender, timestamp and body. The ground truth for the data was obtained using Google's Gemini API [1], which is a large language model with a relatively easy to use API service. The data was processed in batches, and was sent, alongside it's body, to the API, which then provided the classification truth. After thorough human evaluation of the initial 100 samples, I have concluded that Gemini is trustworthy, as all 100 of the evaluations were accurate to my own understanding.

As you can see in 1, the labels are very unbalanced. Whilst this bias could hurt the performance of RoBERTa, which is why for training, I used weighted classes.

Label	Occurences
Academic	2561
Event	710
Club	536
Updates	451
High	375
Work	382
Other	261
Spam	252
Dorm	242
Outside	186
Social	74
Non-Academ	24

Fig. 1. Frequency of Labels

The emails were gathered into a dataframe of 4 columns, and an extra 5th column was added by Gemini to each entry. Most trailing spaces, trailing tabs and newlines are cleaned, but all of the text is still kept in UTF-8.

Typical Email Example
Title: you have submitted your assignment submission for machine translation
Author: "Do not reply to this email (via NU Moodle)" noreply@moodle.nu.edu.kz
Timestamp: "Mon, 19 Feb 2024 19:51:31 +0600"
Body: you have submitted an assignment submission for 'machine translation'. You can see the status of your assignment submission: https://moodle.nu.edu.kz/mod/assign/view.php?id=362854
Label: Academic

Further analysis of the data shows that the average number of words in an email input was **294.23**. In order to calculate the text length, the length of the title, author and the body was summed. The result can be seen in 3, where it shows that the majority of emails fall wihtin the range of 0-500 words. However, some particular outliers exist, going as far as 5000 words in length.

If we consider the mean length by label, the academic-related emails are the shortest at a mean of **199.5**, while the work-related emails tend to be the longest, at a mean of more than double - **490.6** 2.

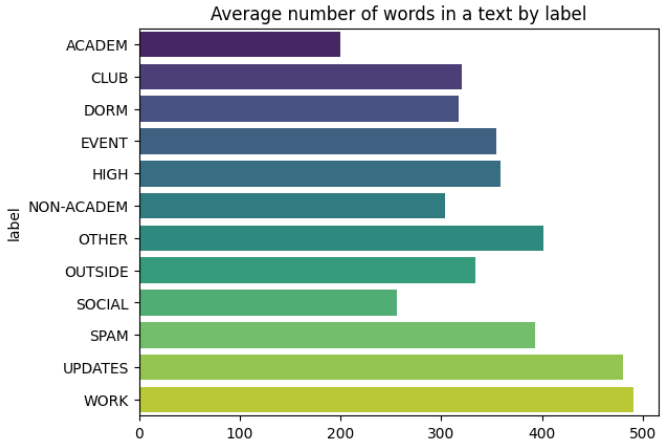

Fig. 2. Mean text length by label

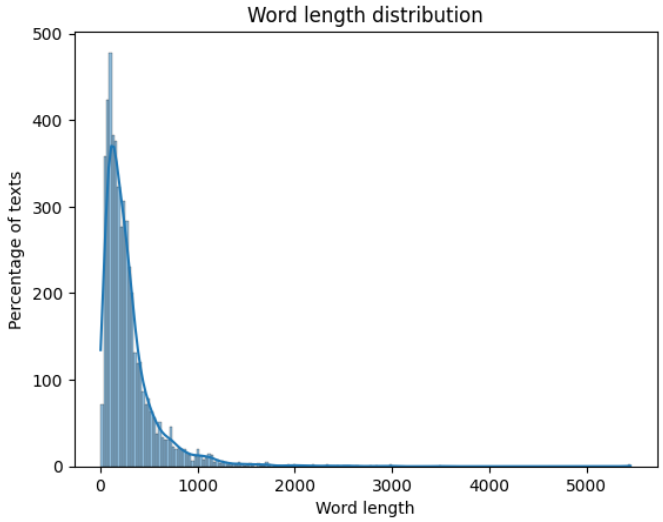

Fig. 3. Distribution curve of text length

III. Methods

The data was split into three datasets - Training, Validation and Testing. A dictionary was created in order to translate between labels and label IDs and vice-versa.

All three datasets were tokenized using a custom-written tokenizer. Since RoBERTa uses solid text, the four different fields were concatenated using [SEP] tokens. The tokenized inputs were then padded to get to 512 tokens, and labels were converted to their IDs.

For training, HuggingFace's Trainer API [2] was used. It offers a simple and reliable way to train big models such as RoBERTa. For training, the train and validation datasets were used. After four epochs of training, the models were evaluated using Trainer API's evaluate() function on the test dataset.

The training and evalutation was done on my personal computer using NVidia RTX 3060's CUDA cores on 12 GB of VRAM, which was enough to run the process at an acceptable pace.

IV. Experiments

In order to successfully label thousands of emails for ground truth values, Gemini API was used. For that purpose, a separate python script was written, that would convert the downloaded .mbox file into a .csv file, and provide Gemini with a prompt to label such emails. The results were gathered into a separate csv file, which were then merged together into **final_output.csv** , which was then used for training the RoBERTa model.

Since a lot of the data was dirty, and some were results of Gemini's hallucinations, rows with null values or rows without labels were removed from the dataset. Overall, out of the **8774** rows present in the .csv file, **2789** were discarded as 'unusable'.

The left data was then filtered further – in order to remove Gemini's hallucinated categories, any categories with less than 10 entries was removed from the dataset. Since Gemini's output table and the original table were merged, some values did not match, which is why the code also prioritizes real titles from the original .csv file (subject_x and author_x, as opposed to subject_y and author_y).

The dataset was split into a train and validation dataset, with 80% of the dataset becoming the training set, 10% going into the validation, and 10% going into the test dataset.

The 12 labels were then defined in the id2label and label2id dictionaries:

```
id2label = {
    "0": "ACADEM",
    "1": "EVENT",
    "2": "UPDATES",
    "3": "CLUB",
    "4": "HIGH",
    "5": "WORK",
    "6": "SPAM",
    "7": "DORM",
    "8": "OTHER",
    "9": "OUTSIDE",
    "10": "SOCIAL",
    "11": "NON-ACADEM"
}
```

The tokenizer, which uses RobertaTokenizeFast, was also customized by concatenating all of the fields into a single token string using separator tokens, as well as converting the labels into the IDs.

The datasets were, afterwards, tokenized using .map(), and their format was set to the torch format, which would adapt the columns to be more usable to the RoBERTa model (input_ids, attention_mask and label).

Since the data was very unbalanced, class weights were calculated, and implemented into the TrainerAPI's loss function

The training was done using Transformers' TrainerAPI. Custom compute metrics were defined to feature F1 score from sklearn's library. The training args used were:

```
    training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=4,
```

```
per_device_train_batch_size=8,
per_device_eval_batch_size=8,
evaluation_strategy="epoch",
learning_rate=[varies],
weight_decay=0.01,
warmup_steps=500,
save_strategy="epoch",
load_best_model_at_end=True,
)
```

The learning rate starts at 5e-5, but according to the learning rate scheduler, finds attempts to find the learning rate with the lowest loss.

During training, two strategies were used:

1) Training is done normally
2) Training is done with the usage of class weights to balance the dataset

After the training had finished, it was evaluated in two ways: once on the standard test dataset (596 entries), and once on a balanced dataset (with there being 100 of each label). The evaluation was done using TrainerAPI's.evaluate() method.

V. Results & Analysis

When evaluating the results, two test sets were used - one with typical distributions of labels, and one that was artificially balanced to have around 50 examples of each label. The results were evaluated using the F1 scoring function, which were 0.754 for the unbalanced test set, and 0.67 for the balanced, but smaller test set.

The balanced test suffers from having too little entry points, due to the unbalanced nature of the dataset, so the presented figures will be from the unbalanced test dataset. This is, in my opinion, okay, since this bias is consistent with real life NU Emails.

As we can see on the confusion matrix, 6, there was only one Non-Academic email in the test dataset, and it was misclassified. This is to be expected, because the label is, actually, quite vague, and can often be interpreted as a different label. We can also see from Figure 5 that all of the misclassified texts were shorter than 400 words, and that the probability of being misclassified rises dramatically for texts with wordcounts below 50. We can also see that the non-academic label has a 100% misclassification rate, for the reason mentioned before. Spam, Academic and Work have a much lower misclassification rate overall, but cateogires like *Social*, *Outside* and *Other* have a much higher misclassification rate, which is explained by the vagueness of the labels. This proves that, as an improvement in the future, the labels could be reworked or the label list may be shortened.

After this, the human evaluation consisted of me testing the inference on new emails. After testing over 50 emails from the inbox, the human tester - me - has judged an accuracy of 42/50.

VI. Conclusion

This project demonstrates the potential of using contextual language models like RoBERTa for classifying emails within specific domains like university inboxes. Although the dataset was imbalanced, the model exhibited acceptable accuracy in identifying various email categories. Notably, the use of weighted classes during training may need further refinement to improve performance on underrepresented categories. The human evaluation with a 42/50 accuracy highlights the model's practical utility. The main issue with the experiment was the size of the dataset - it was simply too small to find a good balance between the test and training sets. The labels were also a bit too vague, which showed detrimental, yet acceptable results.

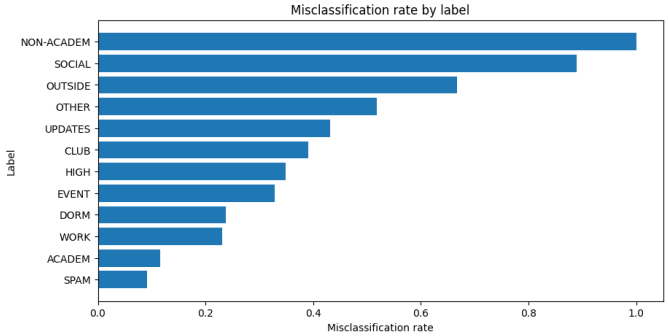

Fig. 4. Misclassification By Label

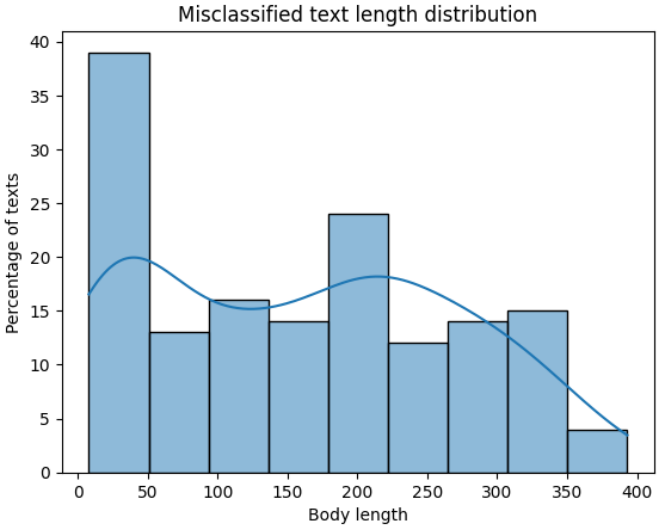

Fig. 5. Misclassification by Length

References

[1] Google, "Google generative ai api (python)." https://ai.google.dev/api/python/google/generativeai. Accessed: 01.03.2024.

[2] Hugging Face, "Transformers: State-of-the-art natural language processing." https://huggingface.co/docs/transformers/index, 2023. Accessed: 03.03.2024.

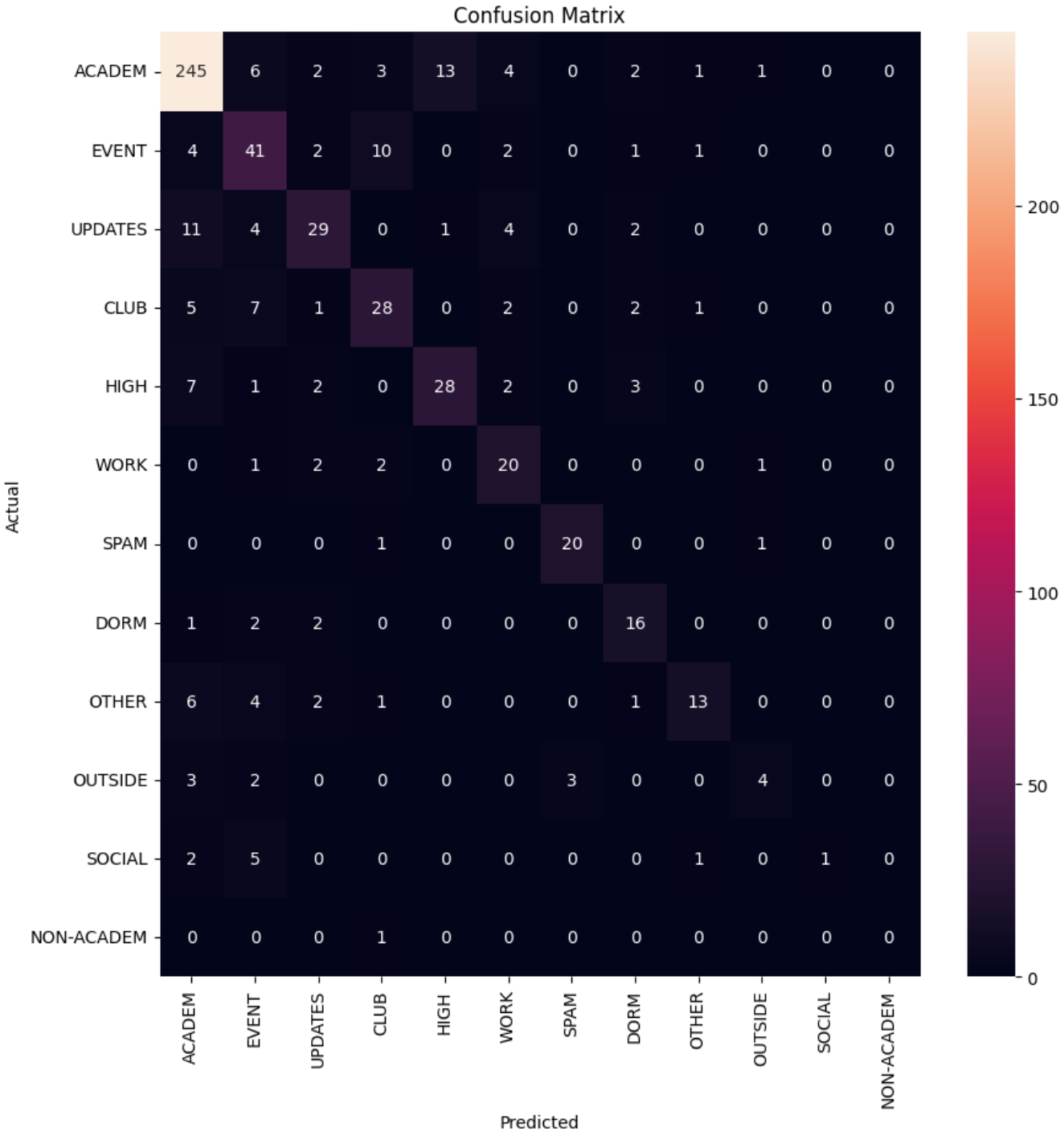

Fig. 6. Confusion Matrix