CSC311 Final Project

Introduction

Improve the News is a non-profit organization that runs a daily and weekly email newsletter. The content includes the most important news stories that occurred that day. They have agreed to give us access to data for their email newsletters, for example things such as the content of the newsletters as well as anonymous user data, like whether a user has opened the newsletter or not. An important problem for the company is figuring out who will open their newsletters, since if they send a person too many emails that they do not open, the newsletter will be sent to that user's junk folder.

Literature Review

We found two relevant papers involving the use of machine learning on email clickthrough.

Predicting Email and Article Clickthroughs with Domain-adaptive Language models (Jaidka et al. 2018) [1] - This paper tries to predict clickthrough rate for various types of emails (finance, cosmetics, movies & TV). It uses the following features: N-grams (3000 features), general inquirer (GI) categorization (184 features), Word2Vec embeddings (100 features), topic modelling (2000 features), part of speech (PoS) tagging (36 features), and meta-feature extraction (13 features). To train the models, they use five-fold cross-validated weighted linear regression on the dataset. The main differences with their method and our method are the amount of features, clean-up, and weighting of their data. They also use k-fold cross validation, as well as a much larger dataset. We feel like these are all things we could implement given enough time but the scope would be much too large for the purposes of the project. In spirit, the first two tasks of our project is pretty much identical to what they are doing.

Predictive Analysis on Tracking Emails for Targeted Marketing (Luo et al. 2015) [2] - This paper uses some very basic features (email features: words of subject line categorized into six categories, day of week, and time of day; user features: country, state, OS type, device type, browser type, email domain type) as well as some basic models (decision tree and SVM). Most of these user features are not able to be replicated in our project due to anonymized user event data. Otherwise our project method is very similar to the first two tasks of this paper. The paper also provides a F1-measure (F-score) of 78% to compare our result to.

Problem Formulation

We want to see what useful questions we can answer from the data. Specifically, we want to answer the following questions: what machine learning model can most accurately predict which users open their emails? What email content prompts people to open their emails?

For example, a certain topic or wording in a headline may trigger more clicks. What features are most correlated to opening emails? Can we use Bidirectional Encoder Representations from Transformers (BERT) to predict headlines that have a high rate of opening?

The data we are given consists of three files: users, newsletters, and events (in CSV format). users_export.csv consists of user IDs along with a string of their email preferences (e.g., “ADWV”). Each letter represents one type of email preference, from the following categories: “Bounced”, “Unsubscribed”, “Active”, “Subscribed to daily newsletter”, “Subscribed to weekly newsletter” (there are actually eight categories in total, but an explanation for the last three codes was not provided by ITN Foundation). newsletters.csv consists of information about each email newsletter: newsletter ID, publishing date, title, HTML content. emailevents_export.csv consists of user ID, event type (“Sent”, “Soft bounce”, “Hard bounce”, “Deferred”, “Delivered”, “Loaded by proxy”, “First opening”, “Opened”, “Clicked”, “Unsubscribed”, “Abuse complaint”), newsletter ID, and timestamp.

To answer the questions, we will execute the following tasks:

1. What machine learning model can most accurately predict whether a user will open a given e-mail? For this task we will use models provided by the sklearn module, which provides an easy, standard interface for their models. We will use Gaussian Naive Bayes, Random Forest Classifier, Ada Boost Classifier, and Bagging Classifier. We will rank them based on accuracy. The data will be split into training, validation, and test sets. The features will be user email preferences, number of previous newsletters sent, number of previous newsletters opened, ratio of previous newsletters opened, time since first newsletter sent and the BERT embedding of the given e-mail. The validation sets will be used to tune hyperparameters so picking the best model does not introduce bias. The features we pick are as follows: user email preferences (8 types), number of previous newsletters sent, number of previous newsletters opened, percentage of newsletters opened, time since first newsletter sent (should be around the time since user signs up), and features from BERT. Refer to `create_npy_inputs.py` in the code for more details.

2. What user behaviour predicts whether a person will open an email? For this task, we will run a Random Forest Classifier on the same data and features as the previous task. Then, we will sklearn’s RandomForestClassifier.feature_importances_ to get the importance of each feature. To determine whether the feature is positively or negatively (or non-linearly) correlated with the rate of opening an email, we will use PartialDependenceDisplay.from_estimator from sklearn, which shows the partial dependence of a feature and the rate of opening an email (e.g. given a feature value, how often will a user open an email).

3. What email content prompts people to open their emails? For this task, the data will be the subject line of the email and the news story headlines in the email body, along with the ratio of people that opened the email. Using this data, we will fine-tune a pre-trained BERT model to predict what ratio of users open an email.

Results

For all of the tasks, the data was split into 70% training data, 15% validation data, and 15% test data using the sklearn train_test_split function. Task 1 did not require hyper-parameter tuning so the experiments were just run on the training and validation sets. Task 2 used the validation set to tune hyperparameters and then the final accuracy was determined using the test set.

Task 1

	Accuracy			Time (s)
	Overall	Unopened	Opened	
GaussianNB	0.74	0.7665	0.7244	20.09
RandomForestClassifier (max_depth=10, n_estimators=10)	0.8114	0.6032	0.9343	18.67
AdaBoostClassifier	0.8068	0.6602	0.8934	463.01
BaggingClassifier	0.8002	0.7228	0.8458	559.02

Note: what matters for runtime is the relative performance, as absolute runtime is system-dependent.

The random forest classifier (RFC) had the best overall accuracy and the best accuracy for the "opened" class, along with having the fastest runtime. However, the Gaussian naive Bayes (GNB) classifier had the best accuracy for the "unopened" class. GNB has a runtime of $O(nd^2)$ where n is the number of samples and d is the number of features. RFC has a runtime of $O(mdn \log n)$ where m is the number of trees. Therefore, for a fixed number of trees and features, the RFC will scale a bit more poorly for increasing amounts of data, which is an important consideration if the task is applied to a larger dataset consisting of thousands of newsletters and millions of users. The conclusion is that GNB is very fast and scales well, while providing decent overall accuracy for this task. However, since we had a high number of features (768 for the embedding and 12 for user preferences), RFC had a faster runtime (since the runtime has a factor of d versus d^2 for GNB) and should be used if attempting to get high accuracy. AdaBoostClassifier and BaggingClassifier scales poorly with number of samples and performance is not better than GNB or RFC, therefore it is determined that they are ineffective for this task.

Task 2

First, the hyperparameters for the random forest classifier (RFC) were tuned using the validation set. The RFC was tested with max depth of $[1, 5, 10]$ and the number of estimators was also tested with $[1, 5, 10]$. Unsurprisingly, the largest max depth and number of estimators yielded the best accuracy (0.81).

(*Note: For Task 2, we did not use the BERT embedding of the email as a feature since we were only interested in feature importances for features related to user preferences*)

		Max depth		
		1	5	10
No. estimators	1	0.6288	0.7967	0.8045
	5	0.7176	0.7941	0.8063
	10	0.7796	0.7924	0.8063

Then, the feature importances were extracted from the RFC (10 max depth and estimators) using a sklearn built-in function.

Ranking	Feature description	Value
1	Number of previous newsletters sent	0.3987
2	Number of previous newsletters opened	0.3971
3	Ratio of previous newsletters opened	0.1065
4	Time since first newsletter sent	0.0532
5	Email pref: Subscribed to weekly newsletter	0.0457
6	Email pref: Unsubscribed	0.0005
7	Email pref: Subscribed to daily newsletter	7.003e-5
8	Email pref: No description	0
9	Email pref: No description	0
10	Email pref: No description	0
11	Email pref: Active	0
12	Email pref: Bounced	0

Interestingly, email preferences were not revealed to be a big factor in determining email opening.

Then, the partial dependence of the features was computed (see Figure 1). There is a very strong positive correlation between the number of previous newsletters sent and the rate of opening emails. There is also a very strong positive correlation between the number of previous newsletters opened and the rate of opening emails. It makes sense that if a person remains subscribed to the newsletter, and they have a history of opening the newsletter, they will be very likely to open the next newsletter.

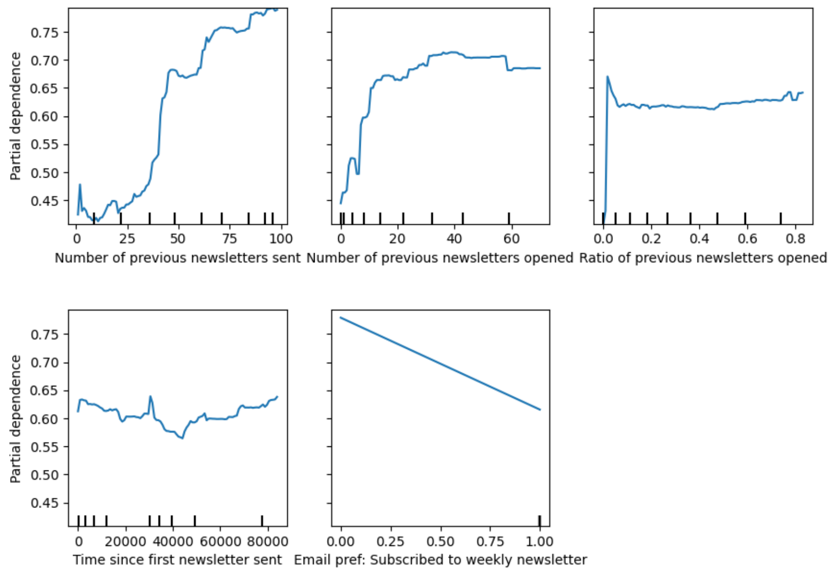

Figure 1: Feature Partial Dependence Plot

Task 3

For Task 3, as an extension to our current work, we were curious about using a newsletter's title and headlines and the proportion of users who opened the newsletter to predict for a given newsletter, what proportion of the users would open the given newsletter. In Tasks 1 and 2, we were looking to solve a classification problem (opened vs. unopened), in Task 3, we are attempting to solve a regression problem (predict proportion).

For this task, since we were specifically interested in using the newsletter's title and headlines to make this prediction, we chose to fine-tune a pre-trained BERT model with a dataset of 171 newsletters, where for each newsletter, we had its title, headlines, and proportion of users that opened it. For this task, we combined the titles and headlines together and used the proportion as our target variable for the regression. The code for this task can be found in `bert_finetune.py`.

In terms of architecture, the fine-tuning process involved adding **dropout** of 0.1 and a fully-connected linear layer on top of the pre-trained BERT model and then training on the dataset specified above. For the loss function, in order to measure the performance of the fine-tuned model, we used **Mean-Squared Error (MSE)**, a common loss function for regression problems.

After training this model on our dataset for 50 epochs, we got the following results:

Training Loss	Validation Loss	Test Loss
0.0012	0.0029	0.003139

Furthermore, provided below is a graph of the training loss and validation loss over the 50 epochs of training:

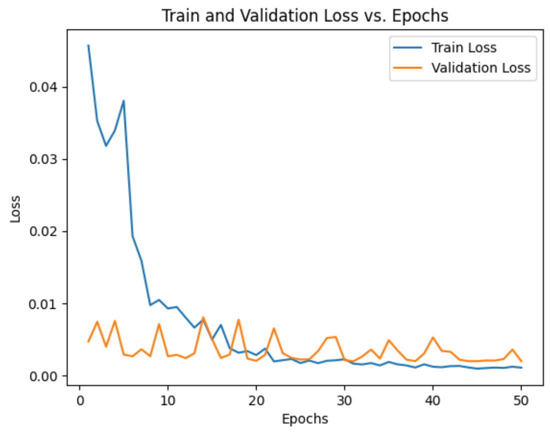

Figure 2: Training and Validation Loss

Overall, it is important to note that while were able to minimize the MSE during the training process and bring the test loss close to 0, indicating that our model was able to predict proportions relatively well, we could only have a proper understanding of our model's performance if we compared it to other models trained on the same problem. Furthermore, in order to have our model generalize well to a larger range of newsletters, we would need to train our model on significantly more emails and newsletters compared to the 171 we used on this task. The primary purpose of this task was to serve as an extension to our work in Tasks 1 and 2 and to see if we could use modern NLP techniques in attacking this problem.

Conclusions

The first two tasks produced results that matched or outperformed previous attempts at predicting email clickthroughs. Then, the third task went beyond existing research to provide specific information about what kind of language gets people to open emails.

Finally, it should be noted that both group members contributed equally to the project. Jonathan worked on Tasks 1 & 2 as well has half the write-up, and Arif worked on Task 3 as well as the other half of the write-up. Specific contributions can be determined using the Git repository.

Bibliography

References

[1] Kokil Jaidka, Tanya Goyal, and Niyati Chhaya. Predicting email and article clickthroughs with domain-adaptive language models. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '18, page 177–184, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Xiao Luo, Revanth Nadanasabapathy, A. Nur Zincir-Heywood, Keith Gallant, and Janith Peduruge. Predictive analysis on tracking emails for targeted marketing. In Nathalie Japkowicz and Stan Matwin, editors, *Discovery Science*, pages 116–130, Cham, 2015. Springer International Publishing.