Email Management System

Celeste Carrasco Zúniga

1 Introduction

Motivated by the challenge of managing an inbox with a high number of daily emails, and by how tedious it is to create labels to classify them on Gmail, I decided to solve the issue by developing an ML system to automatically classify emails into categories such as `action_required`, `logistics`, and `fwd_chain`, with the goal of improving email organization. Using the Enron Email Dataset—a 500K publicly available dataset of corporate communications—I built an end-to-end classification pipeline and iteratively improved it through active learning and manual labeling.

This project is still in the very early stages, however, the goal is for it to become a productivity tool where models assist users in managing their inboxes. It combines classical NLP techniques, uncertainty-based sampling, and hands-on annotation to demonstrate how machine learning can drive productivity tools.

Although there are some projects and papers out there doing something similar, they either focus on identifying spam emails, use the OpenAI API key to take a look every email, use a dataset they created, or don't intend to connect to some email service of choice. Either way, as this project stands now, is more exploratory, but hopefully I can soon use it on my own email inbox.

2 Pipeline Overview

The pipeline for this project includes:

1. **Preprocessing** of email text (combining subject and body)
2. **Feature Extraction** using TF-IDF vectorization
3. **Model Training** using Logistic Regression with a One-vs-Rest strategy
4. **Active Learning Loop** that scores and ranks unlabeled emails by uncertainty
5. **Manual Labeling** of selected emails for the next round of training
6. **Threshold Tuning** to optimize per-label F1 scores
7. **Evaluation** using a held-out test set of 200 manually labeled emails
8. **Experiment Tracking** with MLflow

3 Labeling Schema

The classification model was trained to predict one of the following nine categories:

- `action_required`: Any email that clearly requests a reply, confirmation, rsvp, or next step

- `company_business`: Internal strategy, projects, financials, or formal memos
- `purely_personal`: Chat unrelated to work—holiday greetings, weekend plans, memes
- `logistics`: Scheduling, travel itineraries, meeting room bookings, IT tickets
- `employment`: Hiring, onboarding, referrals, performance reviews, HR policy
- `newsletter`: Subscription digests, marketing blasts, press releases
- `spam`: Unsolicited bulk mail, phishing, obvious advertising
- `empty`: Blank body or placeholder text like 'see attachment' with no attachment
- `fwd_chain`: Forwarded emails or emails that are part of a forwarded chain, often containing multiple replies or forwards

Although this is an inherently multi-label task, the model was trained in a single-label setting to simplify the problem for initial experimentation.

4 Training Experiments

The training utilized multiple active learning cycles, incrementally increasing the labeled dataset based on model uncertainty scores. Beginning with a small labeled subset, each iteration added approximately 200 newly labeled emails, enhancing model performance over time.

For comparative purposes, and out of curiosity, I fine-tuned a DistilBERT model on the best-performing dataset from earlier TF-IDF experiments, achieving decent improvements despite limited labeled data.

Run	**Training Size**	**Notes**	**F1 Macro (Test)**
Baseline	+800 emails	Initial phase with all labeled emails	0.15
Run 2	+1200 emails	Most balanced and with best performance	0.34
Run 3	+1400 emails	Skewed heavily toward `company_business`; hurt generalization on other classes	0.38
Run 4 (DistilBERT)	Same labeled set as Run 3	Transformer fine-tuning in PyTorch; best overall accuracy and macro F1 on test set	**0.39**

5 Evaluation on Held-Out Test Set

To evaluate generalization, I randomly sampled 200 emails from the test dataset — these had never been seen or scored by the model. I manually labeled this test set and used it to compute final evaluation metrics.

Baseline Performance

The initial baseline, trained on approximately 800 emails, exhibited limited effectiveness, establishing a foundation from which subsequent active learning rounds were able to improve from:

Metric	Baseline
Accuracy	24%
F1 Macro	0.15
F1 (`company_business`)	0.24
F1 (`logistics`)	0.44
F1 (`fwd_chain`)	0.21
F1 (`purely_personal`)	0.18

This baseline confirmed that more labeled data was needed to achieve useful performance, particularly on complex or highly variable classes like `fwd_chain`, `personal`, and `action_required`. It established a clear starting point for active learning to iteratively improve the model.

Performance Comparison Summary

Metric	Baseline	Run 2	Run 3	Run 4 (DistilBERT)
Accuracy	24%	31%	28%	**45%**
F1 Macro	0.15	0.34	0.38	**0.39**
Weighted F1	0.23	0.33	0.28	**0.43**
F1 (`company_business`)	0.24	0.17	0.17	0.26
F1 (`fwd_chain`)	0.21	**0.54**	0.37	**0.73**
F1 (`purely_personal`)	0.18	0.14	0.05	**0.22**
F1 (`logistics`)	0.44	**0.49**	0.45	0.44

Despite Run 3 being trained on more data, the performance dropped for key categories due to overfitting to `company_business` examples, emphasizing the necessity of balanced labeling in active learning workflows. Furthermore, examining the confusion matrices for both Run 2 (TF-IDF) and the DistilBERT run (Figure: 5), we clearly observe that DistilBERT made fewer incorrect predictions across several labels compared to the TF-IDF model.

6 Discussion

Looking into two different modeling approaches for text classification provided valuable insights into how different algorithms handle the same dataset. Although both TF-IDF and DistilBERT models used identical training data, the observed variations in their performance showed the strengths and limitations of each approach.

Initially, I went with TF-IDF due to its simplicity, speed, interpretability, and effectiveness with smaller datasets, making it ideal for rapid iterations and learning purposes, making it a good straightforward approach. In contrast, DistilBERT, a transformer-based model which is already known for its strong contextual understanding, naturally outperformed TF-IDF despite the small training dataset size. These results confirmed that transformer models excel in capturing nuanced semantic differences between similar texts.

When comparing confusion matrices (Figure 5) for TF-IDF Run 2 and DistilBERT, we see that DistilBERT demonstrates better class classification, especially for ambiguous labels such as `fwd_chain` and `logistics`. Right away, it is clear TF-IDF frequently confuses `fwd_chain` with `company_business` (24 examples in Run 2) while DistilBERT misclassified only 5 in the same way. This means that DistilBERT's contextual considerations do make a difference, even on small datasets, and thus provide an advantage compared to statistical only approaches. For example, there are less extreme misclassifications in the DistilBERT run.

7 Challenges and Learnings

- **Label Imbalance**: As shown in TF-IDF Run 3, training on unbalanced batches led to reduced generalization. Thus, it important to maintain class balance during active learning to avoid model bias toward more represented labels.
- **Hard-to-Distinguish Classes**: Categories such as `logistics` vs. `company_business` and `spam` vs. `newsletter` just naturally overlap semantically, which underlying multi-label nature of the data and the need for more carefully defined label criteria.
- **Context Matters**: Some classes were difficult to distinguish without considering context clues. For example, I could tell whether a message was a newsletter or spam based on textual cues and implications, but since the TF-IDF model relies only on word frequency, then it would often confused the two.
- **Preprocessing Decisions Can Backfire**: In an effort to simplify the dataset, I removed terms like "FWD" from subject lines and bodies, and created a flag if an email was a forwarded email or not. However, since my model only trained on subject and body fields, this unintentionally removed critical information for identifying forwarded chains. In other words, I should have preprocessed the data a little differently.
- **Engineering Overhead of Transformers**: Getting DistilBERT to run smoothly involved more setup than expected. Aligning library versions, enabling GPU support, and debugging Trainer errors required extensive searching and manual troubleshooting that I had not anticipated.

8 Next Steps

- Retrain the last run of the TF-IDF model with class-balanced batches or apply class weighting to future runs
  - Will not be going the transformer route as my access to uninterrupted GPU resources is limited.
- Build a more complex interface for labeling and predictions
- Fine-tune on personal email inbox for real-world use
- Explore multi-label classification

9 Conclusion

This project taught me how to take an applied NLP project from data cleaning through deployment and evaluation. I learned the importance of thoughtful data annotation, the challenges of real-world imbalances, the trade-offs between model complexity and interpretability, and an overall exposure to all these models and processes. The project, as it is right now, while imperfect, offers a strong foundation for continued improvement and real-world adaptation.

Repository and Documentation

For full source code, setup instructions, and implementation details, see the README in the project repository.

10 Appendix

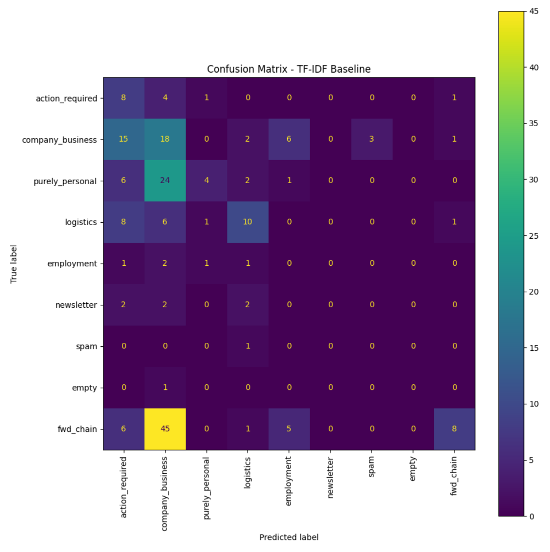

Figure 1: TF-IDF Baseline

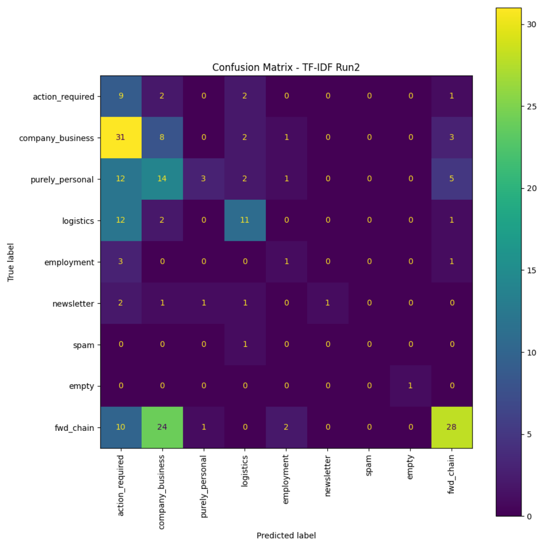

Figure 2: TF-IDF Run 2

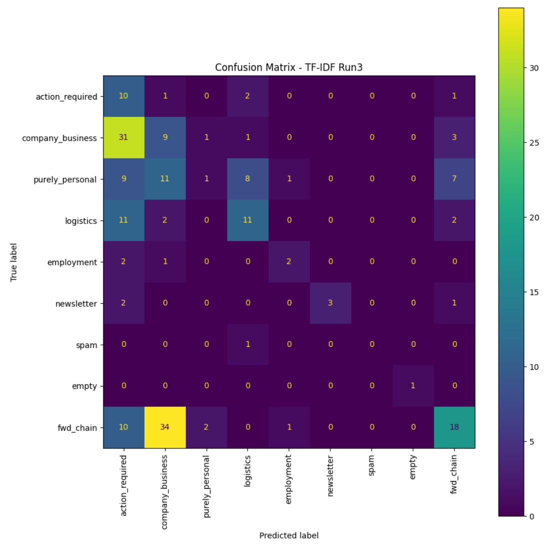

Figure 3: TF-IDF Run 3

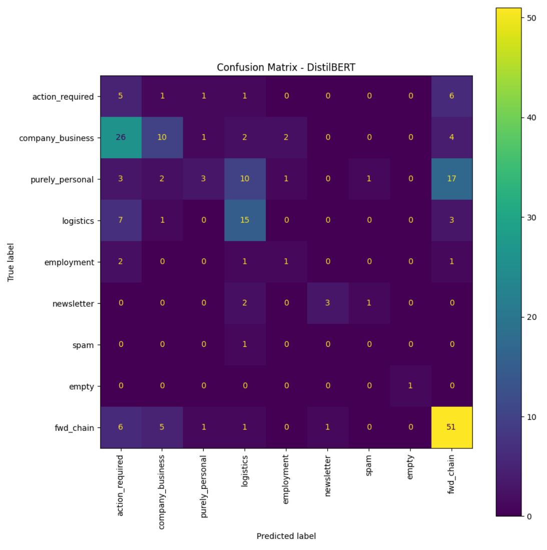

Figure 4: DistilBERT Run

Figure 5: Confusion Matrix for All Runs