Smart Email Sorting

A Machine Learning Approach for Enhanced Productivity

Abstract

Smart Email Sorting is a production-grade machine learning system designed to automatically classify incoming business emails into seventeen distinct categories and highlight time-sensitive messages for rapid user action. The system integrates a context-aware transformer-based classifier, fine-tuned using BERT, with a custom Named Entity Recognition (NER) pipeline built using SpaCy to extract temporal entities and deadlines.

The user-facing dashboard follows a simplified top-to-bottom interaction model. It features an always-visible urgent message ticker, tab-based navigation for high-volume categories, and a read-only archive optimized for non-technical users. An administrative interface supports model monitoring, user management, and manual category overrides, enabling continuous system improvement through retraining.

The implementation roadmap includes dataset creation and labeling, model fine-tuning, backend API development using FastAPI, and a responsive frontend built with React and Tailwind CSS. Production considerations such as asynchronous processing, persistent storage, containerization, orchestration, and CI/CD pipelines are incorporated to ensure scalability and reliability. The primary objective of the system is measurable business impact through reduced cognitive load, faster response times, and increased task execution velocity.

Keywords:
Email classification, BERT, Named Entity Recognition, SpaCy, FastAPI, React, Dashboard, Deadline extraction, Enron dataset.

1. Datasets

The system uses a combination of publicly available and internal datasets:

- Enron Email Corpus consisting of anonymized business emails
- Internal organizational email logs
- A manually labeled dataset of at least 5,000 emails mapped to seventeen predefined categories
- Synthetic data augmentation for low-frequency classes when required

2. Literature Survey

AUTHORS	TITLE	DATASET	YEAR	ALGORITHM	MERITS	DEMERITS	ACCURACY
1. Klimt, Yang	Enron Email Dataset Description	Enron Corpus	1999	—	Real-world corporate data	No fine-grained labels	N/A
2. Devlin et al.	BERT: Pre-training of Deep Bidirectional Transformers	Wikipedia, BooksCorpus	2018	BERT	Strong contextual learning	High compute cost	80–90%
3. Lample et al.	Neural Architectures for NER	CoNLL-2003	2003	BiLSTM-CRF	High NER performance	Requires labeled data	88–92% F1
4. Honnibal, Montani	spaCy NLP Framework	Multiple corpora	1990 - 2019	SpaCy NER	Fast and production- ready	Needs domain tuning	80–90% F1
5. Liu et al.	RoBERTa	BookCorpus, CC-News	2024-25	RoBERTa	Improved pretraining	Higher resource usage	+1–3%
6. Zhang et al.	Email Classification Using Deep Learning	Enron + Internal	2001	CNN / LSTM / BERT	Robust feature extraction	Preprocess ing complexity	85–92%
7. Khandel wal et al.	Practical Deployment of NLP Models	Internal Corpora	2022	DistilBE R T	Faster inference	Slight accuracy loss	−1–5%

8. Proposed Methodology

8.1 System Architecture

The system follows a two-stage processing pipeline:

1. Email Classification
   A fine-tuned BERT (bert-base-uncased) model assigns each email to one of seventeen predefined categories.
2. Entity Extraction
   A customized SpaCy-based NER model extracts temporal entities such as dates, times, and action-related phrases.

Text preprocessing includes HTML stripping using BeautifulSoup, text normalization, removal of email signatures, and tokenization. Class imbalance is handled using oversampling techniques and focal loss strategies. The final model output is returned as structured JSON containing category labels, confidence scores, and extracted deadlines.

8.2 Operational Workflow

Incoming emails are processed asynchronously using Celery with Redis as a message broker to maintain frontend responsiveness. Low-confidence predictions are flagged for human review. Manual category overrides are logged and incorporated into the training dataset, enabling an active learning feedback loop for periodic retraining.

9. System Requirements

9.1 Hardware Requirements

- **Development Environment:**
  1–2 GPUs (NVIDIA T4 or equivalent), 16–32 GB RAM, 4–8 CPU cores
- **Production Environment:**
  2–4 GPUs (NVIDIA V100 or A100) or CPU-scaled inference using DistilBERT
- **Storage:**
  SSD-backed PostgreSQL database with daily backups

9.2 Software Requirements

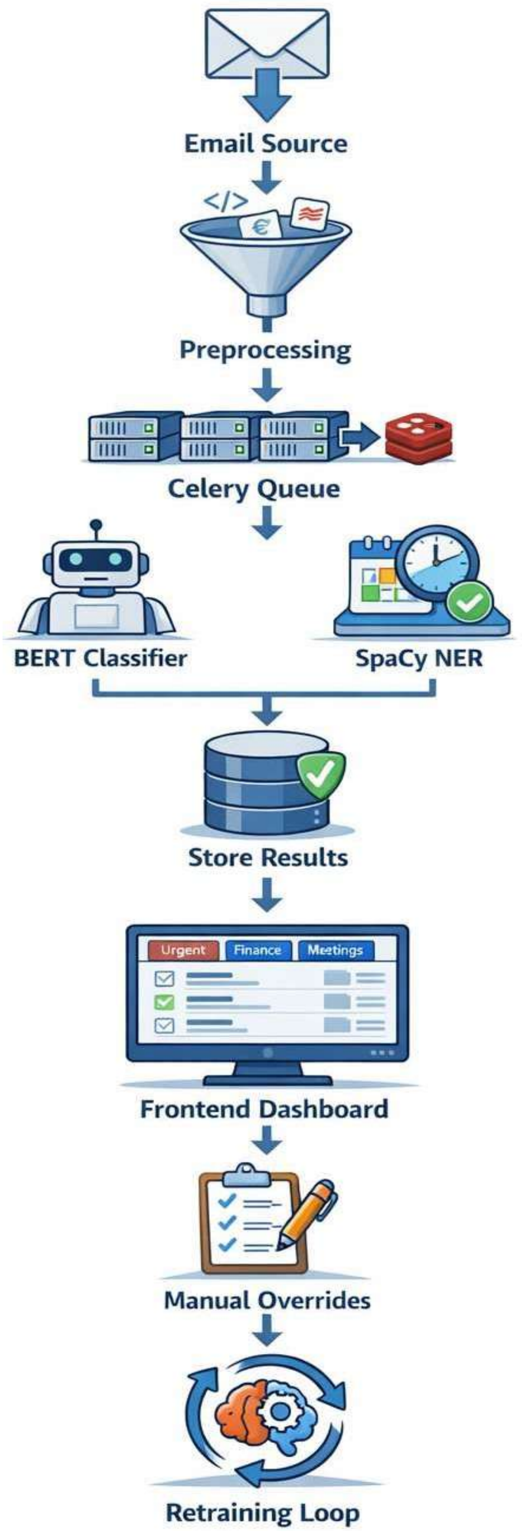
- Operating System: Ubuntu 20.04 LTS
- Backend: Python 3.9+, FastAPI, Celery, Redis, PostgreSQL
- Machine Learning Stack: PyTorch, Hugging Face Transformers, SpaCy, MLflow
- Frontend: React.js, Tailwind CSS, Recharts
- DevOps: Docker, Kubernetes, GitHub Actions or Jenkins

10. System Design

10.1 High-Level Flow

1. Email ingestion via IMAP or batch import
2. Text preprocessing and queuing
3. Email classification and NER execution
4. Storage and indexing of results
5. Visualization through user and admin dashboards
6. Feedback-driven retraining loop

- **System Flow Diagram**


Email Source
Preprocessing
Celery Queue
BERT Classifier
SpaCy NER
Store Results
Urgent
Finance
Meetings
Frontend Dashboard
Manual Overrides
Retraining Loop