Advanced Messaging Platform (AMP): Pipeline for Automated Enterprise Email Processing

Simerjot Kaur* **Charese Smiley*** **Keshav Ramani*** **Elena Kochkina**
Mathieu Sibue **Samuel Mensah** **Pietro Totis** **Cecilia Tilli** **Toyin Aguda**
Daniel Borrajo **Manuela Veloso**
JPMorgan AI Research
{name}.{surname}@jpmchase.com

Abstract

Understanding and effectively responding to email communication remains a critical yet complex challenge for current AI techniques, especially in corporate environments. These tasks are further complicated by the need for domain-specific knowledge, accurate entity recognition, and high precision to prevent costly errors. While recent advances in AI, specifically Large Language Models (LLMs), have made strides in natural language understanding, they often lack business-specific expertise required in such settings. In this work, we present Advanced Messaging Platform (AMP), a production-grade AI pipeline that automates email response generation at scale in real-world enterprise settings. AMP has been in production for more than a year, processing thousands of emails daily while maintaining high accuracy and adaptability to evolving business needs.

From: fixedincomegroup@client.com
Sent: 16-July-2024 13:17 (UTC-05:00)
To: opsteam1@firm.com
Cc: opsteam2@firm.com, opsteam3@firm.com
Subject: Status of my transaction
Attachment:

Hi Team,

Please find below the transactions. Can you please provide what is the status of my transactions? The below transactions are due to settle on 17-July-2024.

Client Identifier	Firm Identifier	ISIN	Transaction Date	Account No.	Portfolio
CIDTA12	F34GP5	US1234567892	16-07-24	A12345	P6763
CRTID23	F6756S	US2345678934	16-07-24	A65789	P9826
CTG45ID	F5738T	US3123456785	16-07-24	A98765	P6702

Best,
Jane Doe
Vice President, Operations Team, New York
Disclaimer: This email has been purely mocked up for proprietary reasons

Figure 1: Example email received by a financial firm.

1 Introduction

Email continues to be a key channel for communication between clients and firms (as shown in Figure 1), particularly in industries like financial services, where rapid, precise, and context-aware responses are critical. However, automating email processing in such environments presents unique challenges due to the proprietary nature of communications, especially in financial services where such data is extremely sensitive.

While LLMs have demonstrated remarkable progress in natural language processing, their generalist nature often limits their effectiveness in industry-specific applications. Financial services, for example, require nuanced handling of jargon and entity recognition, where off-the-shelf LLMs frequently fall short. This gap highlights the necessity of domain-tailored solutions, such as AMP, which meet the precise needs of such tasks.

A major challenge in developing AI-driven email automation is the lack of publicly available datasets for training and benchmarking. Since corporate emails are proprietary and highly sensitive, standard NLP datasets fail to capture the complexities of real-world business communications. This makes it difficult to train models that generalize effectively to industry needs and further underscores the need for custom-built solutions.

In this paper, we introduce Advanced Messaging Platform (AMP), an email automation pipeline tailored for financial services. AMP automates the email handling process from categorization to response generation. AMP is designed to process sensitive financial communications by combining automated workflows with industry-specific customizations. Though designed for financial services, our approach generalizes to other industries facing similar challenges. We discuss AMP's architecture, real-world deployment insights, and broader implications of domain-specific AI solutions in automat-

*These authors contributed equally to this work.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 813–825

ing corporate communications at scale.

2 Background

Emails are a distinctive form of communication (Dürscheid et al., 2013), that is semi-structured, due to their metadata (e.g. sender, recipient) and internal structure (e.g. signatures) (Lampert et al., 2009). They can be multi-modal, contain attachments, and can evolve into multi-threaded conversations involving numerous stakeholders.

Despite the widespread reliance on email, studying corporate email interactions remains difficult due to the lack of publicly available datasets. Most released corpora stem from legal disclosures, such as the Enron dataset (Klimt and Yang, 2004), Hillary Clinton email dataset (De Felice and Garretson, 2018), and Avocado dataset (Oard et al., 2015). Although, these datasets provide valuable resources for research, they are not representative of financial communications, which are heavily regulated, jargon-intensive, and inherently sensitive. The absence of high-quality financial email datasets makes benchmarking solutions a persistent challenge. Banking77 (Casanueva et al., 2020), a rare exception, focuses on intent recognition in conversational settings rather than email workflows.

Previous studies have largely tackled individual aspects of email automation, including subject line generation (Zhang and Tetreault, 2019), email parsing (Lampert et al., 2009), categorization (Lampert et al., 2010; Alkhereyf and Rambow, 2017), action items extraction (Corston-Oliver et al., 2004; Bennett and Carbonell, 2005; Scerri et al., 2010; Lin et al., 2018; Zhang et al., 2022), intent understanding (Wang et al., 2019; Shu et al., 2020), information extraction (Lahiri et al., 2017) and reply generation (Scheffer, 2004; Kannan et al., 2016). Although prior work such as UiPath (Khare et al., 2022) has developed automation tools for email workflows, these solutions do not address the domain-specific constraints of financial communications. In contrast, we introduce AMP, a fully integrated pipeline that combines all of these components into a cohesive system which meets the unique needs of enterprise email processing.

3 Pipeline Architecture

Figure 2 shows the AMP pipeline, which processes emails through multiple stages, including parsing, intent recognition, entity extraction, action implementation, and human validation. To enhance domain-specific understanding, AMP incorporates AMP-LM, a fine-tuned RoBERTa model specialized for financial emails.

AMP-LM: Email Language Model Automating email responses in financial firms require understanding complex, domain-specific jargon, where phrasing and entities vary across teams. Traditional methods struggle with this linguistic diversity, as financial terminology is rarely found in public datasets. Models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and FinBERT (Liu et al., 2021) are general-purpose and are not fine-tuned for email automation. While LLMs like GPT-4o (OpenAI et al., 2024), Qwen-2.5-72B(Qwen et al., 2025), Deepseek-R1(DeepSeek-AI et al., 2025) and PaLM2 (Anil et al., 2023) offer strong language capabilities, their computational costs and production environment constraints currently make them challenging for real-time email automation at scale. Given the volume of daily email traffic, a more efficient, domain-adapted solution is required.

To address these challenges, we further pre-train a Language Model (LM) using the Masked Language Modeling (MLM) objective (Devlin et al., 2019) on proprietary financial email data. MLM enhances contextualized word representations by predicting masked tokens in sentences, allowing the model to learn domain-specific linguistic patterns. Our pre-training dataset consists of a 250MB private corpus containing 92,764 email conversations with over 41M tokens and 2.2M sentences, collected from mailboxes of various operations teams. After exploring several LMs, we choose RoBERTa for its strong performance on downstream tasks. Further details are provided in Appendix A.1.

3.1 Message Parser

The pipeline begins by converting raw HTML into a structured format and splitting email chains into individual messages. A pre-existing legacy model then decomposes each email into key elements: header (sender, recipients, subject, date), greetings (salutations, introductory phrases), body (main content), tables (HTML tabular data), attachments, signature (name, title, contact details), and disclaimer.

3.2 Use Case Mapper

In the financial industry, various operations teams handle distinct tasks, follow unique practices, process specific information, and are entitled to access,

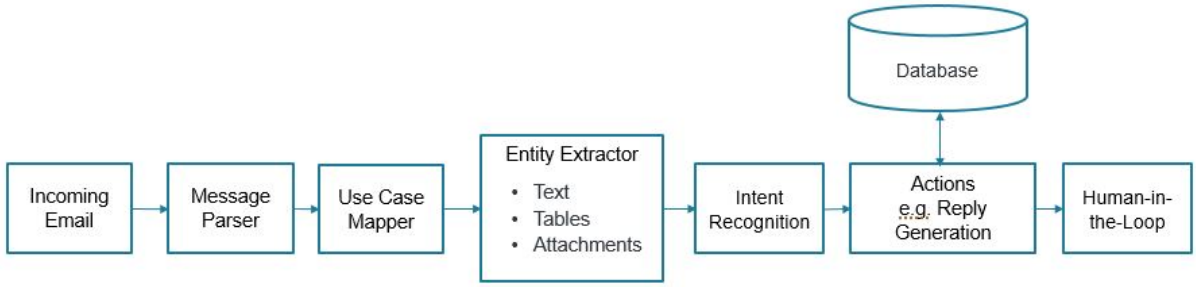

Figure 2: AMP Pipeline Architecture.

or update internal databases. Thus, a use case mapper tags emails based on predefined mappings between mailboxes and use cases. These tags define the scope of subsequent modules, such as intent recognition, entity extraction and actions.

3.3 Entity Extraction

Financial institutions often receive vast volume of emails that are either new inquiries or part of existing email chains. Extracting relevant entities is crucial for determining next steps. However, in multi-threaded email chains, crucial details may appear in earlier messages rather than the latest email. AMP intelligently searches prior emails to ensure no key entity is missed (see Appendix A.3). Entities extracted by AMP include unique identifiers (for teams, firms, clients), security IDs (CUSIP, SEDOL, ISIN[1]), trade economics (volume, amount, currency, dates), portfolio IDs, and account numbers. These may appear in the subject line, email body, tables, or attachments. In real-world scenarios, capturing all relevant entities is crucial. To address this, AMP prioritizes high recall, ensuring comprehensive entity extraction, with precision refined during database queries. AMP employs an ensemble approach, combining deep learning, rule-based techniques, and domain expertise to extract entities from text, tables, and attachments.

Extraction from Text AMP first parses the email body and subject, tokenizing the text, and generating deep learning-based vector representations. Tokens with predefined vectors, likely to be common English words, are filtered out, leaving potential candidates for entities. Financial domain knowledge is then leveraged to identify firm and client unique identifiers, account and portfolio information. Publicly available guidelines are used to detect security IDs. For general trade economics, AMP utilizes spaCy(Honnibal et al., 2020), while AMP-LM enhances the extraction of context-sensitive financial details, such as trade and settlement dates (see AppendixA.4).

Extraction from Tables When processing tables, AMP leverages both column headers and cell values. Headers (e.g., Trade Date, Volume) provide strong semantic signals for entity types. Cell values are extracted and processed using text-based extraction techniques. The entity types predicted from column headers and cell values are compared, and a confidence score is assigned based on the consistency of the predicted entity types across the column and the reliability of the column header as an indicator. In cases where the entity type is ambiguous, contextual information from surrounding cells and the overall table structure is used to validate the predictions.

Extraction from Attachments The extraction process for attachments varies by file type. For text and PDF, the module extracts text content from the original binary format. Then, it processes it using the text extraction methodology. For CSV files and Excel spreadsheets, the module relies on the table extraction methodology. For details on how compressed files are processed, see Appendix A.5.

3.4 Intent Recognition

AMP is designed to handle varying levels of labeled emails for intent recognition.

Semi-supervised Learning Most operation teams share a taxonomy of intents, making models transferable across different use cases. However, in low-label availability scenarios, a semi-supervised clustering-based solution that works at the sentence or email level is used, depending on the problem setting. The process involves obtaining a standardized email representation, extracting key features like verbs and specific nouns, and

[1] https://www.isin.com/

using a TF-IDF vectorizer (Salton and Buckley, 1988) to generate embeddings. The K-Means algorithm (MacQueen et al., 1967) clusters the emails, and a subject matter expert labels the clusters. Hyperparameters are tuned if users find clusters too heterogeneous.

Supervised Learning In cases where a large labeled corpus is available, we fine-tune the AMP-LM model to classify intents. Specifically, we stack a linear layer with softmax activation on top of the first token representation <s> of the AMP-LM pretrained backbone (as usually done with RoBERTa-based sentence classifiers) to map the model to predefined intent categories. Then, we fine-tune the full model on text elements extracted from each email and accompanying labels.

3.5 Actions

Once the intent has been recognized and the entities extracted, each email requires specific actions to be executed to fulfill the intent, including generating a custom reply, moving the email to a given folder (e.g., monthly reports), forwarding the email to internal teams, or initiating a certain workflow (e.g., accessing database to fetch or update information).

Reply Generation Among other actions, reply generation is the most elemental for a messaging system. To ensure consistent, controlled responses, and to avoid the reduced predictability and high costs of LLM-based generation (Kaddour et al., 2023), we opt for a template-based approach. More precisely, the response generation module receives intermediate outputs from upstream elements of the AMP pipeline, and applies use case-specific rules to generate the output HTML code. The rules for processing inputs are based on the business requirements linked to each use case and intent. For instance, if required, a draft requesting additional client information can be generated when no database records are returned in a previous action. Example emails shared by business stakeholders are also leveraged to manually tailor the language and format of the response.

3.6 Human-in-the-Loop

Once an action has been performed, validation by a human is crucial due to the pipeline's production nature involving client-facing teams. It ensures that all client queries are addressed, and information is accurately recorded. With the current pipeline implementation, it is possible to record and evaluate the human interaction with the reply generation action, by comparing the provided draft reply with the actual human reply. We identify three potential scenarios on human interaction with the drafts: (a) Total use: humans retain the full draft in their sent reply; (b) Partial use: humans use some information in the draft; and (c) No use: humans discard the draft entirely. Identifying total use and no use is straightforward, but detecting partial use is challenging due to the need for natural language understanding of response and insights in the actions performed to generate the response.

These challenges correspond to: (1) replies that reword at least part of draft, for example summarizing a draft table in text; and (2) replies that denote some action taken on the information of draft, for example conducting additional research to provide a more comprehensive answer to the client's inquiry. To address the former, we first extract and compare such transaction-related statements from the reply by means of POS tagging. Second, we check the overlap of the sent text with content words from the draft's additional information on the transactions. To address the latter, we perform clustering on the type (2) replies, based on similarity embeddings (Wang et al., 2020), then manually label each cluster with a reply type. This allowed us to identify 30 reply types, and discuss draft usage with users based on these types.

4 Evaluation

We evaluated the performance of AMP both at each module and pipeline levels. No evaluation is needed for the use case mapper and actions module, as both rely on rules predefined with the help of users. For the pipeline evaluation, we use a sample of 200 emails manually annotated.

4.1 Message Parser

We note that the message parser used in our pipeline employs a legacy code base that predates LLM adoption, therefore its implementation is not a contribution of this paper. However, as it is an important component of the AMP pipeline, we perform a thorough evaluation of it on our use-case specific examples. We evaluate it on two dimensions: (1) the accuracy of the segmentation into individual elements; and (2) the accuracy of the classification of each element. The parser produces a correct output for 59% of the examples. The

Entity Type	Client Id.			Firm Id.			ISIN			CUSIP			SEDOL			Other Eco.		
Model	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
AMP-EE	**97.6**	**93.5**	**95.5**	**96.7**	**100**	**98.3**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	67.3	**90.5**	**77.2**
Llama-3.1-8B	58.4	15.8	24.9	51.2	8.6	14.7	84.7	48.1	61.3	29.5	27.6	28.6	36.4	73.3	35.0	72.3	13.1	22.1
w/3-Shot	54.2	39.4	45.6	92.3	8.4	15.4	97.7	78.1	86.8	94.6	74.5	83.3	92.5	74.7	82.6	**90.0**	18.5	30.7
w/5-Shot	53.9	35.3	42.7	95.3	8.2	15.1	95.6	57.4	71.7	95.7	71.3	81.7	93.1	80.7	86.5	50.4	23.6	32.2
Qwen-2.5-7B	58.1	48.9	53.1	28.7	26.3	27.5	90.4	76.1	82.6	40.3	88.3	55.3	41.9	84.3	56.0	82.3	48.9	61.4
w/3-Shot	74.9	44.9	56.1	48.2	17.2	25.4	92.9	95.6	94.3	45.9	95.7	62.1	58.8	92.8	71.9	89.8	57.2	69.9
w/5-Shot	73.1	56.5	63.6	49.1	10.3	17.1	93.9	90.6	92.3	69.5	94.7	80.2	82.9	93.9	88.1	86.7	58.5	69.9

Table 1: **Entity Extractor**: Performance (in %) of the entity extractor on emails from operations teams.

most frequent segmentation error is classifying disclaimers as signatures. Similarly, the most frequent classification errors relate to signatures, which are often divided over multiple segments, and some of them are classified either as Body or Disclaimer. However, only errors on the segmentation and classification of the email body affect the performance of the downstream components, since the rest of the components is not used within the pipeline. We also test Llama-3.1-8B (Touvron et al., 2023) and Qwen-2.5-7B-Instruct (Yang et al., 2024) as an alternative solution for parsing HTML. However, we obtain a fully correct output only for 10% of the tests, with a much higher computational resources usage. More details can be found in Appendix A.2.

4.2 Entity Extraction

We test the entity extraction on manually annotated proprietary business emails. The entity types evaluated include client and firm identifiers, security IDs, and trade economics (dataset statistics in Appendix A.7). We also test the extraction properties of LLMs, specifically Llama-3.1-8B and Qwen-2.5-7B-Instruct in zero-shot and few-shot settings.

Table 1 demonstrates that AMP-EE significantly outperforms LLM-based approaches, achieving the precision and recall of ∼90% for firm and client unique identifiers. The results for security IDs were perfect, reflecting the robust rules and guidelines these identifiers follow within the financial industry. Finally, for trade economics, our recall-heavy entity extractor maintained a high recall rate of ∼90%, ensuring that almost all relevant entities were identified. In contrast, Llama and Qwen struggle in zero-shot settings, failing to generalize domain-specific financial entities. While their performance improves with few-shot prompting, they remain computationally intensive and less reliable than AMP-EE, which is optimized for efficiency, robustness, and real-world deployment. These results highlight the importance of using an ensemble of techniques for different entity types.

4.3 Intent Recognition

We evaluate the performance of our intent recognition methods using proprietary datasets from three operations teams (Ops-X, Ops-Y, and Ops-Z) and the publicly available Banking77 dataset (Casanueva et al., 2020) to assess generalization. Banking77 is chosen as it closely mirrors the structure and complexity of financial emails, which are typically confidential. Detailed dataset statistics are provided in Appendix A.8. We also benchmark the effectiveness of LLMs, specifically Llama-3.1-8B and Qwen-2.5-7B, in zero-shot and few-shot settings to explore their capability in handling domain-specific intent classification.

Performance The experimental results demonstrate that AMP-LM exhibits a significant advantage over RoBERTa in Ops-Z, primarily due to its pretraining on data that closely matches the distribution of Ops-Z. This allows AMP-LM to achieve an impressive F1 score of 97.1%. In contrast, clustering methods show relatively lower performance across the datasets, highlighting their limitations in handling complex intent recognition tasks.

LLMs like LlaMA and Qwen initially show low zero-shot performance, perhaps due to a limited exposure to the domain-specific language and jargon prevalent in the financial sector. However, they show considerable improvement in few-shot settings. However, this improvement still comes at the expense of utilizing larger models, which demand more computational resources. Overall, AMP-LM, a lightweight model, achieves state-of-the-art performance across the compared models, making it particularly suitable for processing the massive volume of emails encountered daily.

4.4 Human-in-the-loop

To assess the effectiveness of AMP-generated draft replies, we compute usage rate metrics using two complementary approaches. The first employs automatic recognition to classify drafts into total,

Model	**Ops-X** (7 classes)			**Ops-Y** (11 classes)			**Ops-Z** (3 classes)			**Banking 77** (77 classes)		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
AMP-LM	**71.2**	**71.2**	**71.0**	**69.2**	**69.4**	**69.0**	**97.1**	**97.1**	**97.1**	93.4	93.2	93.2
RoBERTa	71.2	70.8	70.8	68.3	68.0	67.9	94.3	94.2	94.2	**93.7**	**93.5**	**93.5**
Clustering	52.1	60.7	54.5	25.0	36.9	27.9	50.7	67.6	53.2	48.9	41.7	43.5
Llama-3.1-8B	28.4	29.7	26.3	42.8	38.5	38.4	52.6	57.5	53.3	66.0	59.2	56.7
w/3-Shot	67.0	64.0	63.9	65.7	63.6	64.1	82.6	81.5	81.9	87.1	81.6	82.8
w/5-Shot	68.1	65.8	65.7	68.2	66.8	67.2	86.3	86.0	86.1	89.0	86.4	86.5
Qwen-2.5-7B	28.5	34.8	28.8	41.9	36.3	35.2	76.8	71.2	72.0	70.5	62.0	61.5
w/3-Shot	65.0	64.2	63.6	62.8	60.0	60.1	84.8	83.4	83.7	92.0	91.2	91.2
w/5-Shot	65.9	66.0	65.3	64.6	63.1	63.0	87.9	87.3	87.5	92.9	92.4	92.4

Table 2: **Intent Recognition**: Performance of models (in %) across Ops-X, Ops-Y, Ops-Z, and Banking77. AMP-LM and RoBERTa results are mean values across three runs using different random seeds.

partial, or no use scenarios (Section 3.6), while the second relies on human evaluators assigning labels to these categories. Interestingly, we observed discrepancies between automated and human evaluations. The automated evaluation focuses on whether all relevant information was retrieved, whereas human annotators assess how well the draft aligns with the user's intent and inquiry. Additionally, during the early stages of adoption, users often reformulate, summarize, or tailor the draft to better match client-specific requirements. Due to proprietary constraints, we cannot disclose aggregate results. However, moving forward, we aim to incorporate user modifications into a feedback loop, enabling AMP to continuously refine its outputs. By analyzing added or removed entities and structural adjustments, we can enhance AMP's adaptability and response accuracy over time.

4.5 Pipeline Results

To evaluate performance at each stage, we assessed each module sequentially using a consistent set of 200 manually annotated emails. Among these, our entity extraction and intent recognition models identified 58 emails as either lacking entities or having intents outside the pipeline's scope. From the remaining emails, drafts were successfully generated for 58.5% of the test set. The primary reasons for the failure to generate drafts were as follows: (a) False positives in the entity extraction or intent recognition stages led to invalid database queries, as no corresponding records were found. (b) Some transactions were either outdated or canceled, resulting in an inability to locate them in the database. Finally, we observed that 67.5% of the generated drafts were used by humans. The reasons for less than full adoption are discussed in Section 4.4.

5 Conclusion

In this work, we introduced a pipeline for the automated processing of corporate email messages, detailing its core components: message parser, intent recognition, entity extraction, and the AMP-LM model. Through comprehensive evaluation, we demonstrated the strong accuracy and reliability of each module, as well as the overall pipeline, when tested against human-annotated datasets. These results establish the pipeline as an effective tool for streamlining email workflows, significantly reducing the time employees spend on routine tasks and enabling greater operational efficiency.

6 Limitations

A key limitation of this work is lack of publicly available datasets for financial email automation, making it difficult to benchmark across industries. While we use proprietary datasets for evaluation, data privacy constraints prevent public release. Banking77 offers insights into financial text processing but is not an email corpus and provides only directional guidance for email-related tasks.

While we compare AMP's performance against Llama-3.1-8B and Qwen-2.5-7B, due to compute and production environment constraints we have not been able to compare with larger LLMs. Additionally, the pipeline-based architecture introduces challenges such as cascading errors, where failures in early stages impact later stages, and higher maintenance complexity due to interdependent modules (see Appendix A.12 for production challenges).

Processing attachments adds another layer of complexity, as irrelevant or excessively large files can cause system timeouts. Lastly, AMP does not currently support image processing within emails, limiting its ability to extract insights from embedded screenshots. Future work could explore multimodal approaches to address this gap.

Disclaimer This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates "JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

Acknowledgements We would like to thank Xiaomo Liu for insightful review and discussions. We thank our business partners for their collaboration and invaluable human feedback for various evaluations.

References

Sakhar Alkhereyf and Owen Rambow. 2017. Work hard, play hard: Email classification on the avocado and enron corpora. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 57–65.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Paul N Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 585–586.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45.

Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *Text Summarization Branches Out*, pages 43–50, Barcelona, Spain. Association for Computational Linguistics.

Rachele De Felice and Gregory Garretson. 2018. Politeness at work in the clinton email corpus: A first look at the effects of status and gender. *Corpus Pragmatics*, 2:221–242.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing

Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Christa Dürscheid, Carmen Frehner, Susan C Herring, Dieter Stein, and Tuija Virtanen. 2013. Email communication. *Handbooks of pragmatics [HOPS]*, (9):35–54.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *Preprint*, arXiv:2307.10169.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 955–964.

Arpit Khare, Sudhakar Singh, Richa Mishra, Shiv Prakash, and Pratibha Dixit. 2022. E-mail assistant – automation of e-mail handling and management using robotic process automation. In *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, pages 511–516.

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, pages 217–226, Berlin, Heidelberg. Springer Berlin Heidelberg.

Shibamouli Lahiri, Rada Mihalcea, and P-H Lai. 2017. Keyword extraction from emails. *Natural Language Engineering*, 23(2):295–317.

Andrew Lampert, Robert Dale, and Cécile Paris. 2009. Segmenting email message text into zones. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 919–928.

Andrew Lampert, Robert Dale, and Cecile Paris. 2010. Detecting emails containing requests for action. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 984–992.

Chu-Cheng Lin, Dongyeop Kang, Michael Gamon, and Patrick Pantel. 2018. Actionable email intent modeling with reparametrized rnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2021. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 4513–4519.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Douglas Oard, William Webber, David A. Kirsch, and Sergey Golitsynskiy. 2015. Avocado research email collection.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai,

Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Lance A. Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora, VLC@ACL 1995, Cambridge, Massachusetts, USA, June 30, 1995*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Simon Scerri, Gerhard Gossen, Brian Davis, and Siegfried Handschuh. 2010. Classifying action items for semantic email. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Tobias Scheffer. 2004. Email answering assistance by semi-supervised text classification. *Intelligent Data Analysis*, 8(5):481–493.

Kai Shu, Subhabrata Mukherjee, Guoqing Zheng, Ahmed Hassan Awadallah, Milad Shokouhi, and Susan Dumais. 2020. Learning with weak supervision for email intent detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1051–1060.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wei Wang, Saghar Hosseini, Ahmed Hassan Awadallah, Paul N Bennett, and Chris Quirk. 2019. Context-aware intent identification in email conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 585–594.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Kexun Zhang, Jiaao Chen, and Diyi Yang. 2022. Focus on the action: Learning to highlight and summarize jointly for email to-do items summarization. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4095–4106.

Rui Zhang and Joel Tetreault. 2019. This email could save your life: Introducing the task of email subject line generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 446–456, Florence, Italy. Association for Computational Linguistics.

A Appendix

A.1 AMP-LM

We experimented with RoBERTa using two approaches to further pre-train it on financial email data. In the first approach, we treated entities in the corpus as whole units, replacing them with special tokens before performing MLM. These entities were those extracted by the entity extractor (Section 3.3). This was based on the assumption that the model will be encouraged to learn the surrounding context to infer what the masked entity could be, rather than attempting to learn the patterns of entities themselves. The second approach allows MLM to be performed on the data without replacing entities with special tokens. As a result, the model processes the entire data as it is originally represented in emails. Interestingly, the latter approach leads to satisfactory performance on downstream tasks, as shown later in experiments.

A.2 Message Parser

Experiments are run on a machine equipped with a 16-core AMD EPYC 7R32 CPU, paired with 128GB of RAM and a 24GB Nvidia A10G GPU. The expected output is a python list where elements are the emails of the chain. Each email is a dictionary with keys "Body" and "Header" to separate content from metadata, and the body is a list of segments, where a segment is a dictionary containing the content mapped to its type. Tables are dictionaries of columns mapping to rows and rows are dictionaries mapping row numbers to cell data.

AMP parser The most frequent discrepancies between the parser results and human annotations regard the classification of images and tables appearing in the signatures or disclaimers. The human annotators prioritize the semantic level, classifying them as signature or disclaimer, while the parser prioritizes the syntactic information assigning the class Image or Table. We did not account for errors on image classification because images are not yet supported, and thus they have no influence on the downstream components. Nevertheless, signatures proved to be the hardest part for the parser, with frequent errors on classifying parts of signatures as Unknowns or Disclaimers. Signatures proved to be difficult also for segmentation, where a typical error is incorporating in a signature segment short disclaimers like “Internal only”.

LLM parser We tested Llama-3.1-8B (Touvron et al., 2023) and Qwen-2.5-7B-Instruct as an alternative solution for parsing HTML. This approach is much more demanding in terms of hardware resources (in particular memory) and time, and provides overall worse performances than the AMP parser (Table 3). The LLMs struggle to produce the format required by the downstream tasks and shows poor segmentation performances, producing a correct segmentation and classification only in 10% (Llama) and 6% (Qwen) of the tests.

Large emails, typically carrying a long conversation history, struggle to fit in memory: with the model: 22.5% (45 emails) of the data cannot fit with Llama in the available memory, while for Qwen 18.5% of the data (37 emails). When Llama produces an output 49.5% (99) of them are in an incorrect format, that is, they cannot be converted into a valid python object. Qwen produces a valid python object for 32.5% (65) of the examples. About half of the Llama formatting errors (46) are due to the LLM adding extra text, e.g. “Here is the parsed output:...”. Manually removing such text reveals that the majority (35) would have been a valid python object. Qwen presents this type of behavior as well, but only on 9 examples. However, in 89 instances (44.5%) Qwen returns an invalid for-

mat because it fills the output string with the repetition of a short substring of the HTML input or simply an HTML tag like . Llama valid outputs, 28% (56) of the emails, show good classification performances, with 40 emails with all segments classified correctly (71.4% of the valid outputs), but low segmentation performances, with only 20 emails (35.7% of the valid outputs) correctly segmented. Similarly, Qwen shows stronger classification performances, with 41 correct class predictions (63% of the valid outputs), than segmentation performances, with only 12 emails (18.5% of the valid outputs) correctly segmented. In both cases several errors regard the isolation of the first email, a task where the AMP parser did not make mistakes.

	Segment	Class	Format	Time
AMP parser	81.5%	63.5%	100%	0.11s
Llama-3.1-8B	10.0%	20.0%	28.0%	73.3s
Qwen-2.5-7B	6.0%	20.5%	11.0%	365.5s

Table 3: **Parsing performance metrics** Segment: first email is isolated and segmented correctly. Class: all segments are assigned the correct class. Format: output is properly formatted. Time: average runtime per email.

A.3 Entity Extraction: Handling Multi-chain Emails

In the case of a chain of emails, the last email might not contain all the information needed to handle the client intent. In this scenario, AMP has to determine how far back to look into email threads to extract the necessary information and identify relevant queries. Additionally, the system must ensure that it only captures entities that need to be addressed, and does not act upon entities that have already been dealt with.

Our proposed solution to this problem involves implementing a "look-back" functionality that balances between not omitting important information, and not overwhelming the user with already processed entities. The system captures all the entities if there have been only external conversations, and the mailbox has received the query for the first time. In the remaining cases, the system will perform a look back into previous messages until an entity has been identified. This functionality enables AMP to capture relevant entities, which can be identified from the previous messages, thus maximizing the amount of emails in-scope for the system to handle.

A.4 Entity Extraction: Various Date Types Extraction

For context-specific trade economics, such as identifying various date types (trade, settlement, and payment dates), the AMP-LM model is employed due to its ability to learn context-aware representations of entities. This is typically achieved by treating the entity extraction task as a sequence labeling problem, where BIO tags (Ramshaw and Marcus, 1995) are assigned to tokens to identify the **B**eginning, **I**nside and **O**utside of entities in the text. This tagging system enables the model to learn to capture contextual information around each entity, allowing it to identify the specific type of entity based on surrounding words and phrases.

A.5 Entity Extraction: Compressed Files

In the case of compressed files, like zip or tar archives, the system decompresses the archive and processes each file individually. Text and PDF files within the archive are processed using the text extraction methodology, while CSV and Excel files are processed using the table extraction methodology.

A.6 Intent Recognition: Email Features

In the context of intent recognition in emails, several types of features accurately identify the underlying intent. Features derived from the email's metadata were found to be very useful in the scenarios discussed above. Examples include reports sent from a specific email address, and emails generated by an automatic email failure detection system. Some senders may consistently convey the same intents based on business logic, and automated emails may be part of a book-keeping process. Textual features, found in the subject, body and attachment of the email, are the most common and complex modes of instruction. Attachments are commonplace in financial settings, and can provide instructions or supplement the information already present in the email. Often, a mixture of all these features is used, requiring intent recognition to work with some or all of these features.

A.7 Entity Extraction: Evaluation Statistics

Statistics for the datasets used to evaluate each entity type are presented in Table 4.

A.8 Intent Recognition: Evaluation Statistics

Statistics for the datasets used to evaluate intent recognition are presented in Table 5.

EntityTypes	# of Texts	# of Entities
Client Id.	357	317
Firm Id.	357	81
CUSIP	357	34
SEDOL	357	24
ISIN	357	149
Other Econ.	540	237

Table 4: **Dataset Statistics**

Type	Dataset	Train	Test	Intents
Proprietary	Ops-X	2,920	730	7
	Ops-Y	3,465	612	11
	Ops-Z	1,512	379	3
Public	Banking77	10,003	3,080	77

Table 5: **Intent Recognition:** Dataset Statistics

A.9 LLM Prompts

Parser You are an email parser responsible for the segmentation and classification of emails. You will receive as input an HTML string and you are tasked with parsing the HTML as follows: 1) isolate the current email from the history of previous messages that may be present below the most recent content. 2) Segment the email into the different elements and paragraphs, each segment should represent a piece of information in the email of the same type. 3. Assign to each segment the corresponding type among: GREETINGS (for text that represents a greeting), SIGNATURE (for any text representing contact details of the sender), TABLE (for any information in table format), IMAGE (for images), DISCLAIMER (for text that represents any form of disclaimer), BODY (for text that does not belong to any of the previous types). If the content of a table semantically belongs to another type (different than BODY) then the other type has priority over TABLE. You should use only these types for the annotation and you should output only the annotation in the following format. The output should be a python list with a single dictionary, where the key 'Header' is always {'From': ' ', 'To': ' ', 'Subject': ' ', 'Sent': ' '}, and the key 'Body' contains the list of segments. Each segment is a dictionary with the keys 'Content', which contains the segment information stripped of ALL its HTML tags, and 'Type' which maps the segment to one of the valid types. Tables should be a dictionary of columns, where each column is a dictionary of cells where the row number in string format maps to the content of the cell. Do not output for any reason any message in plain text outside this format. I will now give you an example HTML and the corresponding annotation as an example of the desired output format. It is imperative that you respect this format when providing the annotation as output.
Email body: *<placeholder>*
Desired parsed output: *<placeholder>*
Remember not to add any text outside the python list!

Intent recognition Please categorize the following email into three categories according to the nature of the request. Return the answer that is strictly only the name of one of the categories as provided below. Even if unsure, do not return unknown, select a most likely category. Categories: *<List of answer options>*

Entity Extraction You are an entity extractor. You need to extract the following entities from an email given to you in parsed format. Do not produce any other verbiage. If you are not able to find an entity just write N/A infront of it. Split the entity types using ';' and the format should be Entity Type: All Entity Values. You need to make sure to print all entity types that have been defined even if they are not present. Entities that you need to extract: Client Identifier; Firm Identifier; CUSIP; SEDOL; ISIN; Other Trade Economics. Some clues about various entity types: *<Domain specific details about entities* [2]*>*

A.10 Sample Outputs

For an intuitive understanding of the intermediary outputs generated by AMP, we will walkthrough the pipeline with the example in Figure 1.

Section 3.1 already details the type of structured values that will be provided by the legacy message parser. The use case mapper will consume this output and produce an appropriate use case, say OPS TEAM 1.

The entity extractor determines the scope of entities that need to be extracted using the generated use case tag, OPS TEAM 1. It then applies the corresponding text and table based extractors to come up with the entities. In this case, the output would look like `{client_identifier:` "CIDTA12", `firm_identifier:` "F34GP5", `isin:` "US1234567892", `trade_date:` "16-07-24", `settlement_date:` "17-07-24", `account_number:` "A12345", `portfolio_id:` "P6763"`}`. This output along with the email and use case tag is

[2]Not shown here due to proprietary reasons.

then consumed by the Intent Recognition module, which determines the scope of applicable intents using the use case tag, OPS TEAM 1. in Figure 1, the text will be assigned to a cluster called INTENT CATEGORY 1 based on a trained clustering model. Alternatively, if the AMP-LM model is being used, then the input email is fed to the model with a classification head which would predict the class called INTENT CATEGORY 1. Finally, depending on the use case the appropriate action would be selected and in this case it would be Reply Generation. Once this action executes and data is returned from the database, the template would be composed and presented to the user. For instance, the generated reply would be rendered as:

```
Hi Jane,
Please find the status of your transactions below:
<custom table>
Thanks,
AMP
```

A.11 Comparison of AMP-LM and RoBERTa for Intent Recognition

Table 6 represents the F1 scores of AMP-LM and RoBERTa on the intent recognition task. We report the mean across three independent runs using different random seeds. It can be noticed here that AMP-LM outperforms RoBERTa by 0.2% for **Ops-X**, by 1.1% for **Ops-Y** and by 2.9% for **Ops-Z**. The higher margins in Ops-Z could be attributed to the fact that AMP-LM was further pretrained using data drawn from this team. In **Banking77** however, we notice that RoBERTa outperforms AMP-LM by 0.3%.

Model	Ops-X	Ops-Y	Ops-Z	Banking 77
AMP-LM	$71.0_{\pm 0.3}$	$69.0_{\pm 0.3}$	$97.1_{\pm 0.2}$	$93.2_{\pm 0.1}$
RoBERTa	$70.8_{\pm 0.7}$	$67.9_{\pm 0.4}$	$94.2_{\pm 0.8}$	$93.5_{\pm 0.1}$

Table 6: **AMP-LM vs RoBERTa**: F1 scores (in %) of AMP-LM and RoBERTa across three operational teams **(Ops-X, Ops-Y, Ops-Z)** and public data **Banking77**. Results represent the mean values obtained from three independent runs using different random seeds.

A.12 Discussion

When transitioning our pipeline from development to production, we encountered numerous challenges. These included managing dependencies on critical tools and technologies, addressing infrastructure complexities, adapting to evolving user needs, and upholding stringent security and quality standards to ensure a robust solution. A significant hurdle was our reliance on other tools and technologies. Effective UI design and seamless database management were essential for the pipeline's functionality. Meeting Service Level Agreements (SLAs) and ensuring scalable infrastructure were crucial to maintain reliability under varying workloads. Understanding user requirements posed another challenge, as initial automation needs were often unclear. Rigorous logging practices were implemented to monitor throughput, error rates, and latency, enabling timely adjustments and optimizations. Adherence to firmwide production release controls and rigorous code quality standards was mandatory throughout the deployment process. This included comprehensive security and vulnerability scans to protect sensitive data and uphold system integrity.